UNIVERSIDAD CATÓLICA DE SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"FRAMEWORK ORIENTADO A LA IMPLEMENTACIÓN DE APLICACIONES PARA LA ADQUISICIÓN DE DATOS EN DISPOSITIVOS MÓVILES SOBRE LA PLATAFORMA ANDROID UTILIZANDO CREACIÓN DINÁMICA DE COMPONENTES GRÁFICOS"

Tesis presentada por los Bachilleres:

Manrique Salas Brenda Stephanie
Marocho Cairo Fabricio

para obtener el Título Profesional en Ingeniería de Sistemas.

Arequipa – Perú
2013

A Dios, por darme la oportunidad de vivir y concederme una familia maravillosa.
A mis padres, por su total apoyo, consejos, espíritu alentador y sobre todo por su amor incondicional.
A mi hermana Nataly, por creer en mí, por ser mi mejor amiga y por cada enseñanza que siempre alimenta mi alma.
A mis amigos, por haber llegado a mi vida y por compartir momentos que nos hicieron crecer y emprender cada día con optimismo.

Brenda Stephanie Manrique Salas

A Dios, por permitirme ver el sol cada día.
A mi madre, por su apoyo incondicional, sus consejos y su amor.
A mi padre, por ser un gran ejemplo, el mejor amigo y por su incansable aliento.
A mi hermano, por sus palabras y su compañía.
A mi tío José Luis, que me cuida y guía desde el cielo.
A toda mi familia, por su interés y preocupación.
Y a todas aquellas personas que hicieron posible la realización de este proyecto.

Fabricio Marocho Cairo

AGRADECIMIENTOS

Gracias a Dios, a nuestros padres y hermanos por su paciencia, cariño y aliento.

A la Mgter. Karim Guevara Puente de la Vega y al Dr. Guillermo Calderón Ruiz, por su asesoría y conocimientos brindados en la realización del presente proyecto.

A todos nuestros profesores, por los conocimientos y consejos adquiridos en estos años.

A las personas que invirtieron parte de su tiempo e hicieron posible la realización de este proyecto.

A nuestros amigos.

ÍNDICE

CAPÍTULO 4: CASO DE ESTUDIO

CAPÍTULO 5: PRUEBAS Y RESULTADOS

ÍNDICE DE FIGURAS

CAPÍTULO 2: MARCO TEÓRICO CAPÍTULO

CAPÍTULO 3: DESARROLLO DE LA PROPUESTA

CAPÍTULO 4: CASO DE ESTUDIO

CAPÍTULO 5: PRUEBAS Y RESULTADOS

ANEXO A

ÍNDICE DE TABLAS

CAPÍTULO 2: MARCO TEÓRICO

CAPÍTULO 3: DESARROLLO DE LA PROPUESTA

CAPÍTULO 4: CASO DE ESTUDIO

CAPÍTULO 5: PRUEBAS Y RESULTADOS

ANEXO A

RESUMEN

En el ámbito del desarrollo de software cuando hablamos de Framework hacemos referencia a una estructura conceptual y tecnológica compuesta de artefactos o módulos de software concretos que sirvan de base para la organización y desarrollo de software, esto permite a las personas especializadas enfocarse en requerimientos más importantes y mantener un producto final de fácil mantenimiento; a razón de ello el presente proyecto "FRAMEWORK ORIENTADO A LA IMPLEMENTACIÓN DE APLICACIONES PARA LA ADQUISICIÓN DE DATOS EN DISPOSITIVOS MÓVILES SOBRE LA PLATAFORMA ANDROID UTILIZANDO CREACIÓN DINÁMICA DE COMPONENTES GRÁFICOS" establece un marco de trabajo para reducir el tiempo de implementación de aplicaciones móviles para adquisición de datos.

El Framework está compuesto por 2 módulos, un aplicativo móvil que se instalará en los dispositivos móviles, éste podrá sincronizar los formularios disponibles y enviar las respuestas al servidor, así mismo cuenta con la capacidad de trabajar de manera autónoma y sin conexión. Por otra parte el módulo web tiene un conjunto de páginas que permite la creación, edición y eliminación de formularios, ítems y estrategias, contando también con ficheros que permiten la interpretación de los datos enviados por el cliente móvil en formato JSON.

Debido a que el Framework propuesto permite la parametrización de componentes gráficos mediante una interfaz web, que finalmente serán visibles en la aplicación móvil, el presente trabajo se centra en la descripción de la arquitectura y desarrollo de

dicha colaboración entre el módulo web y el módulo móvil, proponiendo un caso de estudio que permita comprobar el correcto funcionamiento del Framework.

ABSTRACT

In the field of software development when we talk about Framework we refer to conceptual and technological structure composed of artifacts or specific software modules as a basis for the organization and software development, it allows specialized people to focus on major requirements and get an easy maintain of final product; because it, "FRAMEWORK FOR IMPLEMENTATION OF APPLICATIONS FOR DATA ACQUISITION ON MOBILE DEVICES ON ANDROID PLATFORM USING DYNAMIC CREATION OF GRAPHICAL COMPONENTS" establishes a framework for reducing implementation time of mobile applications for data acquisition.

The Framework consists of two modules, one mobile application to be installed on mobile devices, it can synchronize the available forms and send responses to the server , also has the ability to work independently , offline. On the other hand the web module has a set of web pages that allows creating, editing and deleting forms , items and strategies , also has files that allow the interpretation of the data sent by the mobile client in JSON format.

Because the proposed Framework allows parameterization of graphical components through a web interface, which eventually will be visible on the mobile application, this paper focuses on the description of the architecture and development of the collaboration between the web module and the mobile module proposing a study case to verify the correct operation of the Framework.

INTRODUCCIÓN

Actualmente el obtener información, almacenarla y analizarla sigue siendo, con bastante frecuencia, un proceso manual. La toma de datos y su recopilación en formatos impresos es bastante susceptible a errores humanos por lo cual realizar este proceso con ayuda de dispositivos móviles viene a ser una solución bastante viable para automatizar este proceso.

Se sabe que el uso de dispositivos móviles provee de diversas ventajas a personas y organizaciones, desde mejorar procesos hasta ofrecer servicios a clientes, siendo así un hecho cada vez más difundido el uso de aplicaciones móviles en ámbitos que tienen por interés principal la información recopilada para lograr una óptima toma de decisiones. Algo que se debe tomar en cuenta es que el costo y el conocimiento requerido en el desarrollo de este tipo de aplicaciones son elevados, así como la poca reusabilidad que estas aplicaciones ofrecen, ya que son diseñadas y desarrolladas para tareas específicas.

Es importante tomar en cuenta que el incremento en el uso de estos dispositivos inteligentes y el abandono de la utilización de formatos impresos para procesos como la recopilación de datos puede generar un ahorro significativo en los diversos materiales empleados en las labores manuales y a la vez contribuir significativamente con el desarrollo sostenible del medio ambiente al tomar en cuenta la materia prima utilizada para producir hojas de papel para dichos formatos impresos. Es importante tomar en cuenta también las ventajas económicas obtenidas al utilizar teléfonos inteligentes de plataforma abierta, como es Android y las características que este tipo de dispositivos

nos pueden brindar en cuanto a la riqueza visual de los componentes gráficos, es así que todas estas razones, entre otras que serán expuestas más adelante, justifican la realización del presente proyecto.

El Framework desarrollado en el presente documento, propone una estructura definida y organizada para la implementación de aplicaciones móviles para adquisición de datos, siendo el objetivo de este trabajo estudiar la arquitectura de dicho Framework y brindar las facilidades para colaborar con su desarrollo.

Para explicar cómo se desarrolló el Framework, el proyecto se ha dividido en 5 capítulos:

En el capítulo 1 titulado “Planteamiento Teórico” se identifica el problema a solucionar, se describen los objetivos a cumplirse, las variables, la hipótesis, la justificación, alcances y limitaciones del proyecto.

En el capítulo 2 titulado “Marco Teórico” se describen aplicaciones y proyectos relacionados al propuesto. Además se exponen conceptos fundamentales acerca de las tecnologías utilizadas en el proyecto.

En el capítulo 3 titulado “Desarrollo de la Propuesta” se realiza el análisis, diseño y desarrollo del proyecto bajo la metodología RUP, se explica la tecnología usada y se presentan las interfaces resultantes.

En el capítulo 4 titulado “Caso de Estudio” se parametrizará 2 formularios para demostrar el correcto funcionamiento del Framework, para demostrar la interacción del usuario con el Framework se utilizaron artefactos de UML.

En el capítulo 5 titulado “Pruebas y Resultados” se muestran los resultados de la evaluación hecha por los expertos.

Finalmente se presentan las conclusiones y recomendaciones, así como los anexos, los cuales describen el manual de usuario del Framework, el código fuente del Framework, tablas y procedimientos almacenados del Framework y el cuestionario elaborado para la evaluación de los expertos.

CAPÍTULO 1

PLANTEAMIENTO TEÓRICO

1.1. EL PROBLEMA

1.1.1. Título

Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android utilizando creación dinámica de componentes gráficos.

1.1.2. Descripción del problema

1.1.2.1. Definición del problema

El Perú viene experimentando un significativo crecimiento respecto al uso de tecnologías móviles en los sistemas de toma de datos para diversas labores generando así la necesidad de personal con experiencia y conocimiento en el desarrollo de aplicaciones móviles, hecho que genera un alto costo en distintos aspectos y la indecisión de dar un paso adelante en la innovación e inserción de este tipo de tecnologías.

Actualmente el desarrollo de aplicaciones móviles se ve afectado por muchas limitantes entre ellas el alto costo de desarrollo y el tiempo empleado en su implementación. Existen herramientas que permiten la creación de este tipo de aplicaciones ofreciendo soluciones particulares a problemas determinados, pero ninguna provee la flexibilidad hasta el punto de permitir la parametrización de componentes gráficos, persistencia de datos locales, la modificación de un formulario sin tener la necesidad de reinstalar la aplicación, la definición de valores y estrategias de toma de decisiones, entre otras, así mismo muchas de estas aplicaciones implican costos adicionales en mantenimiento y excesiva carga de red generando un incremento en el tiempo de respuesta causando una deficiente calidad de experiencia en los diversos niveles de usuarios.

Muchas veces la información requerida en la toma de datos no se limita a un conjunto único de posibilidades ni a una forma única de resolución de variables, es así que las demás herramientas carecen del alto grado de flexibilidad que algunas labores de toma de datos necesitan.

1.1.2.2. Área y línea de investigación

Área de Investigación

Ingeniería del software.

Línea de Investigación

Desarrollo de aplicaciones móviles.

1.1.2.3. Tipo y nivel de investigación

Tipo de Investigación

Investigación aplicada: Se aplicarán conceptos teóricos a problemas reales de manera práctica, abstrayendo las soluciones para desarrollar la aplicación.

Nivel de Investigación

Descriptiva: Con el desarrollo del prototipo y su aplicación, describiremos los resultados obtenidos según los indicadores señalados anteriormente.

1.2. OBJETIVOS

1.2.1. General

Proponer un Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android utilizando creación dinámica de componentes gráficos.

1.2.2. Específicos

- Definir una arquitectura para la implementación del Framework propuesto.
- Elaborar clases necesarias para el desarrollo de este tipo de aplicaciones.
- Proponer lineamientos generales para la utilización y aplicación práctica del Framework.
- Generar un prototipo que demuestre la utilidad y efectividad del uso del Framework propuesto.

1.3. HIPÓTESIS Y VARIABLES

1.3.1. Hipótesis

Es probable que el uso del Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android utilizando creación dinámica de componentes gráficos, permita establecer una línea base a partir de la cual se puedan generar una serie de aplicaciones con similitud en la adquisición de datos que sirvan para distintos propósitos.

1.3.2. Variables e indicadores

Variable Independiente: Creación dinámica de componentes gráficos**.**

Indicadores:

- Consistencia: Mantiene unidad y coherencia entre los componentes instanciados.

Variable Dependiente: Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android.

Indicadores:

- Facilidad de uso.
- Tiempos de implementación.

- Grado de reutilización de controles.

1.4. JUSTIFICACIÓN

En los últimos años, el hecho de poseer información se ha convertido en una ventaja para muchas organizaciones, esto les permite ponerse un paso más adelante y así poder tomar decisiones de manera anticipada. Es a razón de este hecho que se modelan, diseñan y desarrollan distintas aplicaciones móviles para recolección de datos que permiten ampliar el área de trabajo y quitar los límites de acción de las organizaciones.

Muchas veces el desarrollo de aplicaciones móviles para recolección de datos implican costos elevados así como conocimientos acerca de la plataforma que se utilice, motivos que desalientan y limitan el uso de esta tecnología, es por ello que se propone brindar una herramienta que permita simplificar el desarrollo de este tipo de aplicaciones, haciendo frente a las limitaciones y problemas mencionados, así mismo facilitar la inserción de este tipo de tecnologías en el ámbito nacional.

1.5. ALCANCES Y LIMITACIONES

1.5.1. ALCANCES

- Las aplicaciones construidas con ayuda del Framework permitirán la adquisición de datos utilizando la creación dinámica de componentes.
- El Framework será desarrollado para implementar aplicaciones sobre smartphones con Sistema Operativo Android a partir de la versión 2.2.

1.5.2. LIMITACIONES

- El Framework propuesto se orienta exclusivamente a la construcción de aplicaciones para la adquisición de datos.
- Orientado exclusivamente a sistema operativo Android.
- Como plataforma de implementación del Framework se concebirá software libre en el lado del cliente móvil y software propietario en el lado del servidor y base de datos, bajo la infraestructura tecnológica de Microsoft Windows.

CAPÍTULO 2

MARCO TEÓRICO

2.1. ESTADO DEL ARTE

En esta sección analizaremos algunas de las distintas herramientas existentes para la toma de datos, entre las cuales tenemos el Framework RoboQuiz, Quipper y Google App Inventor, los cuales permiten crear una interfaz gráfica sin necesidad de conocimientos en programación, pero que a la vez limitan al usuario al momento de querer modificar una aplicación ya creada. El proyecto “Testing” nos da luces sobre la parametrización flexible de cuestionarios orientado exclusivamente al ámbito educativo, este proyecto sólo está construido en la parte cliente considerando como futuros desarrollos la arquitectura completa, por su parte el software de Formiik permite la creación de formatos de ingreso de datos para trabajos en campo, pudiendo ser evaluados e interpretados desde una interfaz web, se limita únicamente a la recopilación de información. Por otra parte los proyectos más relacionados al Framework propuesto son Maritaca que permite configurar formularios generando un archivo ejecutable por cada formulario y Google Forms que permite crear formularios agregando controles, pero que necesita de conexión a internet para poder ser utilizado. Seguidamente describiremos algunos de estos proyectos:

2.1.1. Google App Inventor

Autor: Lifelong Kindergarten research group en el MIT Media Lab.

App Inventor es una aplicación escrita en java, libre y de código abierto que permite a las personas con cualquier experiencia en programación crear aplicaciones de software para el Sistema Operativo Android. Esta aplicación utiliza una interfaz gráfica de usuario que les permite jalar y situar bloques, que representan a objetos, para construir sus aplicaciones sin tener que escribir código de programación.

Esta aplicación permite modificar los eventos y propiedades de los controles, así mismo agregar imágenes y sonidos. Una vez generada alguna aplicación que sirva para la toma de datos no hay forma de modificar las preguntas, respuestas o estructura de este tipo de aplicación, creando la necesidad de generar nuevamente la aplicación con las modificaciones correspondientes y por lo tanto volver a instalar la aplicación en el dispositivo cliente. [MIT12]

2.1.2. Robo Quiz Framework

Autor: Tobias Sutor

Framework y conjunto de clases Java diseñadas para acelerar el desarrollo de aplicaciones de evaluación sobre dispositivos con Sistema Operativo Android.

Desde una interfaz de escritorio el usuario puede crear fases y preguntas con sus respectivas respuestas, al ser un test de respuestas cerradas se usan únicamente los controles “CheckBox” y “RadioButton”, al finalizar la parametrización debe seleccionarse la ruta del SDK de Android previamente instalado, así como una ruta de salida en la cual se generará la aplicación. Todo este proceso genera una aplicación única, poco flexible con controles definidos, en caso de querer modificar alguna pregunta o agregar una respuesta, debe generarse una aplicación nueva con los cambios indicados. [ROB13]

2.1.3. Quipper Quiz

Autor: Quipper

Esta aplicación móvil creada para el Sistema Operativo Android y iOS, permite acceder a una serie de cuestionarios acerca de temas definidos, requiere de una cuenta de Facebook o Gmail para usarla. En caso de que el usuario desee crear su propio cuestionario, debe acceder a la interfaz web de la aplicación, ingresar y crear sus preguntas con respuestas cerradas, una vez terminado el proceso de creación, cualquier usuario de Quipper puede acceder al cuestionario creado, únicamente buscando el título del mismo. Quipper Quiz no permite la toma de datos desde su aplicación, sirve exclusivamente para crear cuestionarios y resolverlos, pero no tiene la capacidad flexible de parametrizar los valores de las respuestas. [GOO13]

2.1.4. Proyecto: Testing, aplicación de cuestionarios para Android

Autor: Miguel Degayón Cortes

El objetivo de esta aplicación es que un alumno pueda realizar pruebas de autoevaluación que le permitan en cualquier momento evaluar sus conocimientos sobre las asignaturas que cursa. La aplicación fue concebida como la parte cliente de un sistema Cliente-Servidor, así mismo se elaboró una pequeña aplicación que hace las veces de servidor implementando el protocolo de red creado de forma expresa para la descarga de los cuestionarios. El alumno realiza el proceso de identificación en la aplicación y acto seguido se descarga en el teléfono la colección de cuestionarios disponibles, este es el único momento en que la aplicación hace uso de la conexión a la red móvil, lo que optimiza los inconvenientes que se pueden encontrar al necesitar una continua conexión a la red.

La aplicación utiliza la Base de Datos 'SQLlite Expert Personal', está disponible para la versión de Android 2.0 o superior, para solucionar el problema que tenían con la IP Dinámica del servidor situado en el equipo utilizaron la solución DynDNS Updater de la empresa Dynamic Network Services Inc, empresa dedicada a ofrecer la resolución de nombres DNS en soluciones IP dinámicas. Por último la aplicación utiliza ficheros XML para definir los esquemas y peticiones que utilizará el alumno. [DEG11]

2.1.5. Formiik

Autor: Formiik

Aplicación muy parametrizable, permite la generación de cualquier tipo de formato de ingreso de datos para labores en campo. Cuenta con una interfaz web desde la cual se administran los datos y visualizan reportes, es una aplicación exclusiva para toma de datos más no permite parametrizar resultados. Formiik es compatible con una amplia gama de plataformas móviles, está disponible bajo el modelo de distribución SaaS (Software as a Service), el costo de implementación fluctúa entre 4000 y 15000 dólares americanos. [FOR12]

2.1.6. Maritaca

Autor: Maritaca Team

Maritaca es un sistema desarrollado en el Instituto de Ciencia y Tecnología de la Universidad Federal de Sao Paulo, que sirve para soportar la recolección de datos mediante dispositivos móviles. Está compuesto por 3 partes: Aplicación de dispositivo móvil Android, servicio web de configuración de formularios y repositorio de datos. Maritaca permite configurar formularios en línea permitiendo utilizar diferentes componentes gráficos, una vez terminado un formulario podrá ser modificado y será guardado en el repositorio de datos, seguidamente se genera un archivo .apk por cada formulario configurado. Al ser instalada la aplicación en el

dispositivo móvil Android se tiene que sincronizar el formulario para luego poder ser respondido, finalmente es enviado al servidor donde será mostrado en grillas. Maritaca fue desarrollado en Eclipse y HTML5, utiliza comunicación vía archivos XML e implementa un modelo de servicio llamado SaaS (Software as a Service), que es un modelo asociado a aplicaciones basadas en computación en la nube que ponen servicios disponibles al usuario vía navegador web. [MAR13]

2.1.7. Google Forms

Autor: Google Inc.

Google Forms es una herramienta gratuita de Google Inc. que permite la creación y edición de formularios según las necesidades del usuario, permite la recolección de datos en tiempo real, tanto desde un browser mediante Google Drive como desde la aplicación para dispositivos móviles siempre y cuando se cuente con una conexión para transmisión de datos sea 3G o Wi-Fi, los datos enviados son almacenados en hojas de cálculo. [GOO06]

2.2. MARCO CONCEPTUAL

En este apartado, describiremos algunos conceptos que permitan establecer los fundamentos básicos a fin de comprender y desarrollar la propuesta.

2.2.1. Smartphone

Teléfono móvil que ofrece más funciones que otro teléfono móvil normal, tiene una gran variedad de prestaciones, permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad. Estas aplicaciones pueden ser desarrolladas por el fabricante, por operadores o por personas particulares [PEI10].

2.2.2. Aplicaciones móviles

Una aplicación móvil es un software que funciona en dispositivos móviles, son programas que se pueden descargar y que realizan una tarea específica, entre los tipos de aplicaciones tenemos a las aplicaciones nativas, son aquellas que están diseñadas para ejecutarse en el sistema operativo de un dispositivo, también en el firmware de la máquina, y que por lo general tiene que ser adaptada para distintos dispositivos, también tenemos aplicaciones Web o de navegador, son aquellas la totalidad o algunas partes de los programas se descargan de la Web cada vez que se ejecutan y que por

lo general se puede acceder desde todos los dispositivos móviles con capacidad Web.

Es necesario notar que en los últimos años muchas de las aplicaciones nativas utilizan la conectividad Web en tiempo real, y las aplicaciones Web ofrecen modos fuera de línea (offline) que se puede acceder sin conexión a la red. Como resultado, algunas de estas aplicaciones se conocen como aplicaciones híbridas [SCH11].

2.2.3. API Android

Generalmente hablando, una API (Application Programming Interface) es una biblioteca organizada de funciones, procedimientos o métodos que pueden ser utilizados software como una capa de abstracción, es así que [DEV13] nos dice que en Android existe un número entero, llamado "Nivel de API" que identifica como único al marco de trabajo de la API ofrecida por cada versión de Android.

La API de Android consiste en:

- Un conjunto básico de paquetes y clases.
- Un conjunto de elementos y atributos XML para la declaración de un archivo de manifiesto.

- Un conjunto de elementos y atributos XML para la declaración y el acceso a los recursos.
- Un conjunto de Intenciones.
- Un conjunto de permisos que pueden solicitar aplicaciones, así como refuerzos de permiso incluido en el sistema.

El Nivel API es muy importante a la hora de optimizar la experiencia de los usuarios y desarrolladores de aplicaciones, es así que:

- Permite que la plataforma Android se describe el marco máximo de la revisión de la API que soporta.
- Se permite a las aplicaciones describir la revisión de la API marco de trabajo que requieren.
- Permite que el sistema de negociar la instalación de aplicaciones en el dispositivo del usuario, de tal manera que la versión de las aplicaciones incompatibles no están instalados.

Se debe tener en cuenta que cada Nivel de API cuenta con soporte para todas las versiones anteriores a dicho Nivel.

2.2.4. Sistema Operativo Android

Sistema operativo y conjunto de software de código abierto, basado en Linux para teléfonos móviles y otros dispositivos, originado por un grupo de

empresas conocido como la Open Handset Alliance, liderada por Google y que actualmente está conformada por 84 compañías de diferentes ámbitos.

Android fue diseñado desde cero para permitir a los desarrolladores crear aplicaciones móviles que aprovechan al máximo todo lo que un dispositivo puede ofrecer, está basado en el kernel de Linux, además, utiliza una máquina virtual personalizada que se ha diseñado para optimizar los recursos de memoria y hardware en un entorno móvil. [OPH13]

a) Concepto

Android es una plataforma completa de código abierto diseñada para dispositivos móviles, promovida por Google y propiedad de Open Handset Alliance cuyo objetivo es "acelerar la innovación en los consumidores móviles y ofrecer una rica, económica y mejor experiencia móvil" teniendo a Android para hacerlo. Como tal, Android está revolucionando el espacio móvil, es una plataforma abierta que separa el hardware del software que se ejecuta en la misma, esto permite que un número mayor de dispositivos puedan ejecutar las mismas aplicaciones y crear un gran ecosistema para desarrolladores y consumidores.

b) Arquitectura

Android está compuesto por varias capas cada una con características propias y propósitos definidos.

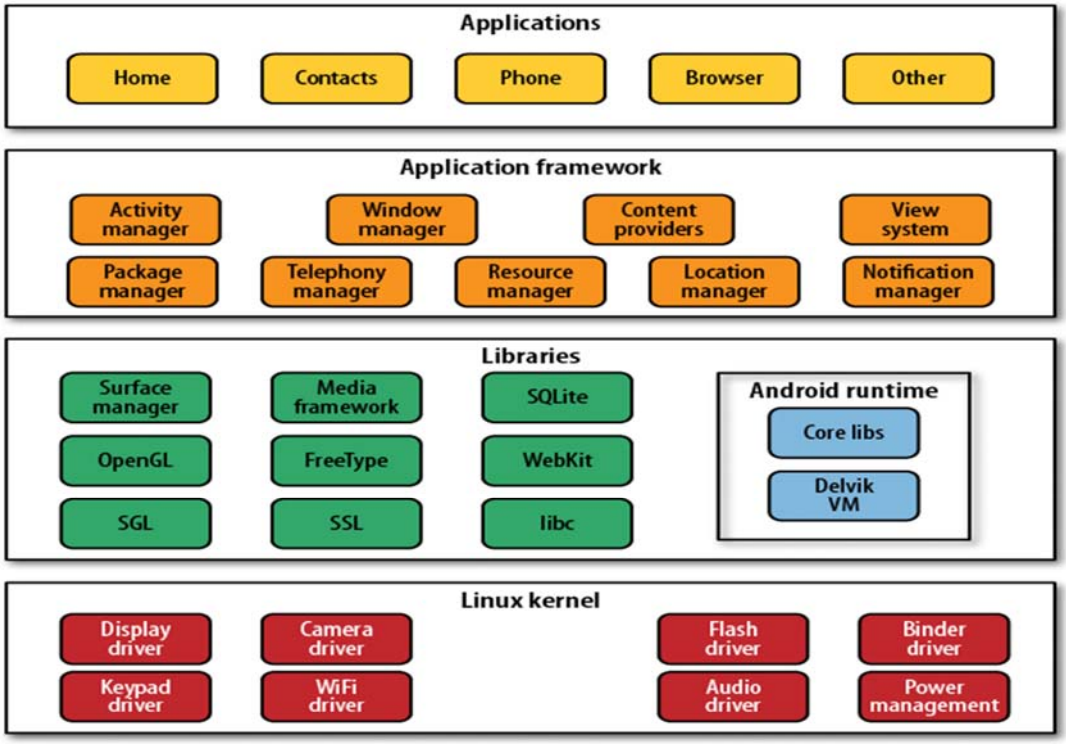

Figura 2.1. - Arquitectura Android.
Fuente: [GAR11]

Aplicaciones: Se incluyen tanto las aplicaciones nativas de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas utilizan los servicios, las API y librerías de los diferentes niveles. [JAI09]

Una aplicación es un paquete único, un archivo .apk que tiene, por lo general tiene tres componentes principales:

- Ejecutable Dalvik, este es todo el código fuente de Java compilado en la máquina virtual. Este es el código que ejecuta la aplicación.
- Recursos: Todo lo que no es código. Una aplicación puede contener imágenes, audio, vídeo, así como numerosos archivos XML que describen diseños, paquetes de idiomas, etc.
- Bibliotecas nativas: Opcionalmente, la aplicación pueden incluir algún código nativo, como bibliotecas C/C + +. Estas bibliotecas pueden ser empaquetados junto con el archivo APK.

Marco de trabajo de aplicaciones: Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. [JAI09] También proporciona una abstracción genérica para el acceso a hardware, gestiona la interfaz de usuario y los recursos de las aplicaciones.

Bibliotecas: La mayoría de ellas escritas en C/C++, proporcionan a Android la mayor parte de sus capacidades más características. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

Runtime de Android: Es un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su

propio proceso, con su propia instancia de la máquina virtual Dalvik. Esta máquina virtual ha sido diseñada para optimizar la memoria y los recursos de hardware en el entorno, a diferencia de la máquina virtual de Java, basada en el uso de pilas, la máquina virtual Dalvik está basada en registros. Interpreta archivos en el formato Dalvik Executable (*.dex) que fue optimizado para el almacenamiento eficiente y ejecución mapeable en memoria, al mismo tiempo permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar. No se ejecuta la máquina virtual de java puesto que se quiere optimizar al máximo los recursos y enfocar el funcionamiento de programas a un entorno con recursos escasos de memoria, procesador y almacenamiento como es el de los teléfonos.

Núcleo Linux: Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. La elección de Linux 2.6 se debe principalmente a dos razones: la primera, su naturaleza de código abierto y libre se ajusta al tipo de distribución que se buscaba para Android; la segunda es que este kernel de Linux incluye de por sí numerosos drivers, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistemas operativo. [JAI09]

c) Componentes de una aplicación.

Activity: Es el componente más habitual de las aplicaciones para Android. Una Activity refleja una determinada actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente Activity a través de vistas representadas por clases como View y sus derivadas. Este componente se implementa mediante la clase de mismo nombre Activity. [JAI09]

Una Activity es por lo general una sola pantalla que el usuario ve en el dispositivo. Una aplicación normalmente tiene múltiples Activities, el usuario va y viene entre ellas. Como tal, las Activities son la parte más visible de las aplicaciones. [GAR11]

Ciclo de Vida de una Activity

Iniciar una Activity puede resultar bastante caro en aspectos de memoria. Podría implicar la creación de un nuevo proceso, la asignación de memoria para todos los objetos de la interfaz, la parametrización de los objetos según la configuración XML, y la creación de toda la pantalla.

Ya que se realiza mucho trabajo para poner en marcha una Activity, sería una pérdida simplemente eliminarla una vez que el usuario abandona esa

pantalla. Para evitar esta situación, el ciclo de vida del Activity se gestiona a través del Activity Manager.

El Activity Manager se encarga de crear, destruir, y gestionar las Activities. Por ejemplo, cuando el usuario inicia una aplicación por primera vez, el Activity Manager creará su Activity y la muestra en la pantalla. Luego, cuando el usuario cambia de pantalla, el Activity Manager se moverá del Activity anterior al siguiente iniciado por el usuario. De esta manera, si el usuario quiere volver a un Activity anterior, se puede iniciar más rápidamente. Las Activities que el usuario no utiliza desde hace tiempo serán destruidos con el fin de liberar espacio para que la actual esté activa. Este mecanismo está diseñado para ayudar a mejorar la velocidad de la interfaz de usuario y así mejorar la experiencia general al usar una aplicación. [GAR11]

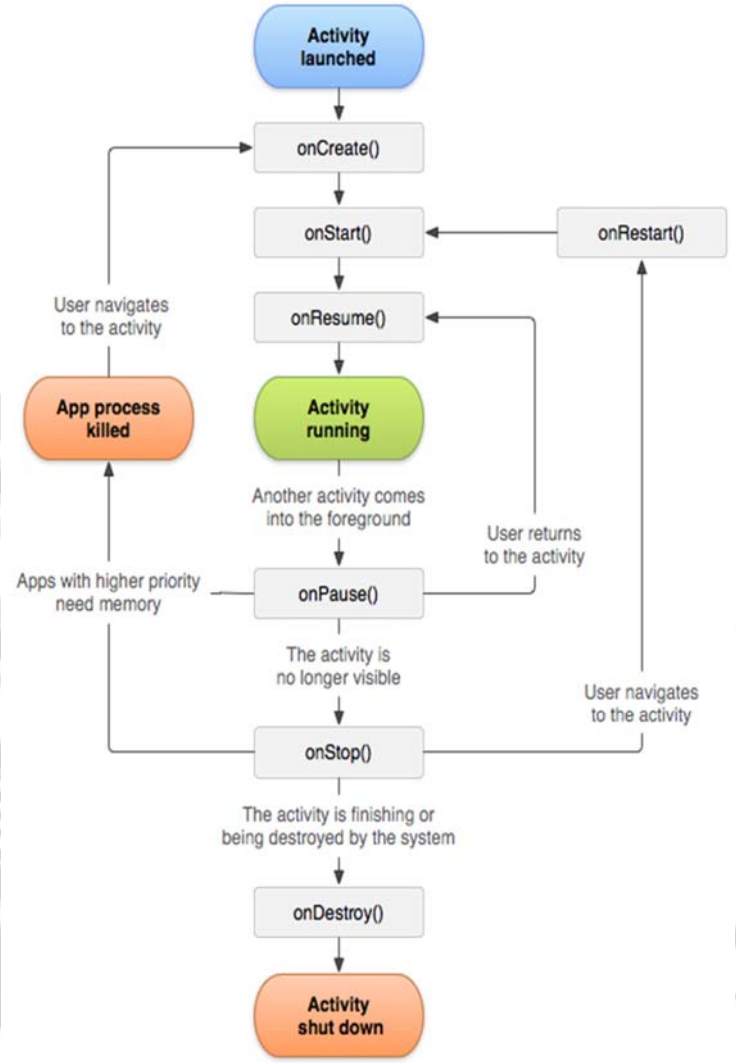

Figura 2.2. - Ciclo de Vida de un Activity.
Fuente: [DEV13]

A continuación describiremos las fases del ciclo de vida de un Activity:

- **Starting:** Cuando una Activity no existe en la memoria, se encuentra en un "estado de partida". Mientras se está iniciando, la Activity puede ejecutar un conjunto de métodos que los desarrolladores pueden sobrescribir. Finalmente, la Activity pasará a ejecutarse, esta

transición de un estado a otro a partir de una ejecución es una de las operaciones más costosas en términos de tiempo y uso de memoria lo que también afecta directamente a la vida de la batería del dispositivo. Esta es la razón exacta por la que las Activities que no están en uso no se destruyen automáticamente. [GAR11]

- **Running:** Indica que una Activity se encuentra actualmente en la pantalla y en interacción con el usuario. Todos los eventos (escribir, tocar la pantalla y presionar los botones) están a cargo de esta actividad, es por ello que sólo hay una actividad en funcionamiento en cualquier momento dado. La Activity que se ejecuta es la que tiene prioridad en términos de conseguir la memoria y los recursos que necesita para funcionar lo más rápido posible. Esto se debe a que Android quiere asegurarse de que la ejecución de la Activity sea enérgica y brinde una buena experiencia para el usuario. [GAR11]

- **Paused:** Cuando una Activity no interactúa con el usuario, pero sigue siendo visible en la pantalla, se dice que está en un estado de pausa. Esto no es un escenario típico, porque la pantalla del dispositivo es generalmente pequeña, y una Activity estaría ocupando toda la pantalla. A menudo se ve este caso con cuadros de diálogo que se presentan delante de una Activity, haciendo que ésta entre en pausa. [GAR11]

- **Stopped:** Cuando una Activity no es visible, pero todavía se encuentra en memoria, se dice que está detenida, pudiendo volver a ser utilizada o destruida y eliminada de la memoria. El sistema guarda las Activities detenidas, ya que es probable que el usuario vuelva a utilizarlas, es por ello que reiniciar una Activity detenida es mucho menos costoso que iniciar una nueva. [GAR11]

- **Destroyed:** Una Activity destruida ya no está en la memoria. El Activity Manager decide que ya no se necesita y la elimina de la memoria. Antes de adoptar este estado, se pueden realizar ciertas acciones, como guardar cualquier información no guardada, aunque se recomienda hacerlo cuando la Activity pasa al estado de pausa. [GAR11]

Intents: Son mensajes que se envían entre aplicaciones, pueden iniciar Activities, comunicar a un servicio cuando iniciarse o detenerse, o simplemente enviar datos. Los Intents pueden ser explícitos o implícitos. En un Intent explícito, el remitente específica claramente qué componente debe ser el receptor. En un Intent implícito, el remitente especifica únicamente el tipo de receptor. Cuando se tienen instaladas dos o más aplicaciones capaces de ejecutar una tarea, el sistema le pedirá que seleccione una de ellas para realizar una acción determinada. También puede configurar una aplicación por defecto. [GAR11]

Content Providers: Son interfaces relativamente simples que proveen un nivel de abstracción de los datos almacenados en el dispositivo, así, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que se considere. Además, los datos pueden ser compartidos entre distintas aplicaciones. [JAI09]

Services: Se ejecutan en segundo plano y no tienen ningún componente de interfaz de usuario, pueden realizar las mismas acciones que un Activity. Los servicios son útiles para las acciones que se quieren realizar durante un tiempo, con independencia de lo que está en la pantalla. Los servicios tienen un ciclo de vida mucho más simple que las Activities (Fig. 2.2). El desarrollador puede iniciar un servicio o detenerlo. Por lo tanto, se debe considerar el consumo de recursos compartidos, tales como la CPU y batería. [GAR11]

d) Entorno de Desarrollo (IDE)

Al momento de elegir el entorno de desarrollo adecuado para nuestro proyecto, se tomó en cuenta las diferentes características de diversos IDEs, es así que se extrajo las principales características según lo descrito en [TOR12]. Las características evaluadas fueron: costo, dificultad de utilización del IDE, si requiere aprendizaje de un nuevo lenguaje, flexibilidad del IDE y modificación del código, concluyendo que el entorno de desarrollo necesario para este proyecto es Eclipse, porque es

un IDE gratuito, de dificultad media de desarrollo, no requiere el aprendizaje de un nuevo lenguaje diferente al nativo de Android (Java), posee flexibilidad y modificación de código para futuros desarrollos. Se debe sumar a las ventajas del desarrollo en este entorno, la gran cantidad de información existente en la red, lo cual ha facilitará el diseño e implementación del proyecto.

e) Versiones de Android

Debido a que Android es un sistema operativo móvil que cuenta con un gran número de actualizaciones, es importante conocer cada versión desarrollada puesto que las actualizaciones comúnmente corrigen fallos de programa e implementan nuevas funcionalidades, datos importantes que enriquecen el proyecto. A continuación se describen las versiones conocidas de Android:

- Android 1.0 Nivel de API 1 (Septiembre 2008)

  Primera versión de Android. Nunca se utilizó comercialmente [POL11].

- Android 1.1 Nivel de API 2 (Febrero 2009)

  Se corrigieron algunos errores de la versión anterior, pero no se implementaron mayores funcionalidades [POL11].

- Android 1.5 Cupcake (Abril 2009)

  Esta versión cuenta con la posibilidad de proyectar teclado con predicción de texto en pantalla, grabación avanzada de audio y video dando la posibilidad de subir los videos a Youtube, los primeros widgets de escritorio (Mini aplicaciones diseñadas para proveer de información, servicios o interacción), además cuenta con transiciones de ventanas mediante animaciones y conexión bluetooth A2DP, que posibilitó la conexión de un auricular a distancia [POL11].

- Android 1.6 Donut (Septiembre 2009)

  Como nuevas habilidades, incluye la síntesis de texto a voz, el Android Market se convierte en Play Store, da soporte a VPN a dispositivos con pantallas WVGA (Wide Video Graphics Array). Aparece un nuevo atributo XML, onClick, que puede especificarse en una vista. [POL11]

- Android 2.0/2.1 Eclair (Octubre 2009 / Enero 2010)

  La versión 2.0 incorpora una API para manejar el Bluetooth 2.0, optimiza la velocidad de hardware, incluye una nueva interfaz del navegador y permite utilizar html5. La clase MotionEvent soporta eventos en pantallas multitáctil. Mientras que la versión de Android 2.1 realiza una actualización menor siendo incorporados mecanismos para administrar la configuración de la caché de aplicaciones, almacenamiento web, y modificar la resolución de la pantalla. Incluye

la capacidad de utilizar fondos animados de pantalla y mejora la velocidad del hardware. [POL11]

- Android 2.2 Froyo (Mayo 2010)

  Mejora de velocidad en ejecución de aplicaciones, reduciendo la fragmentación de la memoria y el rendimiento, esto se da gracias a el compilador JIT de la máquina Dalvik. Permite instalar las aplicaciones en un medio externo como la tarjeta de memoria SD, permite además realizar copias de seguridad de datos desde las aplicaciones y como una de las mejoras principales en esta versión se incluyen Wi-Fi hotspot y tethering por USB [POL11].

- Android 2.3 Gingerbread (Diciembre 2010)

  Ofrece soporte para la tecnología VoIP, NFC (Near field communication), decodificadores de audio tipo AAC, soporte para mayores tamaños de pantalla y resoluciones (WXGA y superiores), soporte nativo para sensores (como giroscopios y barómetros) , también se mejora la gestión de energía y control de aplicaciones; y se cambia el sistema de ficheros YAFFS a ext4 [POL11].

- Android 3.0/3.1/3.2 Honeycomb (Febrero 2011)

  Es la Primera versión de la plataforma que soporta procesadores multinúcleo. La máquina virtual Dalvik ha sido optimizada para permitir multiprocesamiento, lo que permite una ejecución más rápida

de las aplicaciones, incluso aquellas que son de hilo único [POL11]. Esta versión de Android fue realizada pensando sólo en el mercado de tabletas, es así que algunas aplicaciones son incompatibles con la versión 2, pero aún así incluye mejoras de sistema multitarea, permite manejar dispositivos conectados por USB (tanto host como dispositivo) además del Protocolo de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP).

- Android 4.0 / 4.0.3 Ice Cream (Octubre 2011 / Diciembre 2011)

  Se unifican las versiones anteriores (2.x para teléfonos y 3.x para tabletas) pudiendo ser instalado en cualquiera de estos dispositivos, se presenta una mejora del sistema multitarea y del gestor de tráfico de datos de Internet, pudiendo ver el consumo de forma gráfica pudiendo definir los límites de consumo que utilizaremos, esto para evitar cobros no presupuestados por parte del servicio de datos. Se presenta la posibilidad de realizar aceleración por hardware, pudiendo ser la interfaz dibujada por el GPU (Graphics Processing Unit) [TOR12].

- Android 4.1 / 4.2 Jelly Bean (Julio 2012 / Noviembre 2012)

  Se incorporan varias técnicas para mejorar la fluidez de la interfaz de usuario, como: sincronismo vertical, triple búfer y aumentar la velocidad del procesador al tocar la pantalla. Se introducen varias mejoras en Google Search. Se incorporan nuevo soporte para usuarios internacionales: como texto bidireccional y teclados instalables.

También se permite actualizaciones parciales de aplicaciones [POL11]. En la versión 4.2 se cuenta con la posibilidad de crear varias cuentas de usuario para las tabletas, la función PhotoSphere para realizar fotos esféricas además de un control fluido de la cámara del dispositivo móvil.

- Android 4.3 Jelly Bean (Julio 2013)

  Google lanzó Jelly Bean 4.3, bajo el lema "Una forma aún más dulce Jelly Bean" el 24 de julio 2013, entre las mejoras más importantes está el soporte para Bluetooth de Baja Energía, OpenGL ES 3.0, locación de Wi-Fi en segundo plano, opciones para creadores de Apps, y mejoras en la seguridad.

  Se incorporan varias técnicas para mejorar la fluidez de la interfaz de usuario, como: sincronismo vertical, triple búfer y aumentar la velocidad del procesador al tocar la pantalla. Se introducen varias mejoras en Google Search. Se incorporan nuevo soporte para usuarios internacionales: como texto bidireccional y teclados instalables. También se permite actualizaciones parciales de aplicaciones [POL11]. En la versión 4.2 se cuenta con la posibilidad de crear varias cuentas de usuario para las tabletas, la función PhotoSphere para realizar fotos esféricas además de un control fluido de la cámara del dispositivo móvil.

f) Actualidad de Android.

Se muestran los principales factores que motivaron la elección del sistema operativo Android para el desarrollo del proyecto:

- **Sistemas Operativos más utilizados:** En los últimos años, las compañías están luchando por liderar el ámbito de los diversos sistemas operativos móviles que encontramos en el mercado, es así que podemos apreciar la evolución de las ventas a nivel mundial de las plataformas móviles más importantes:

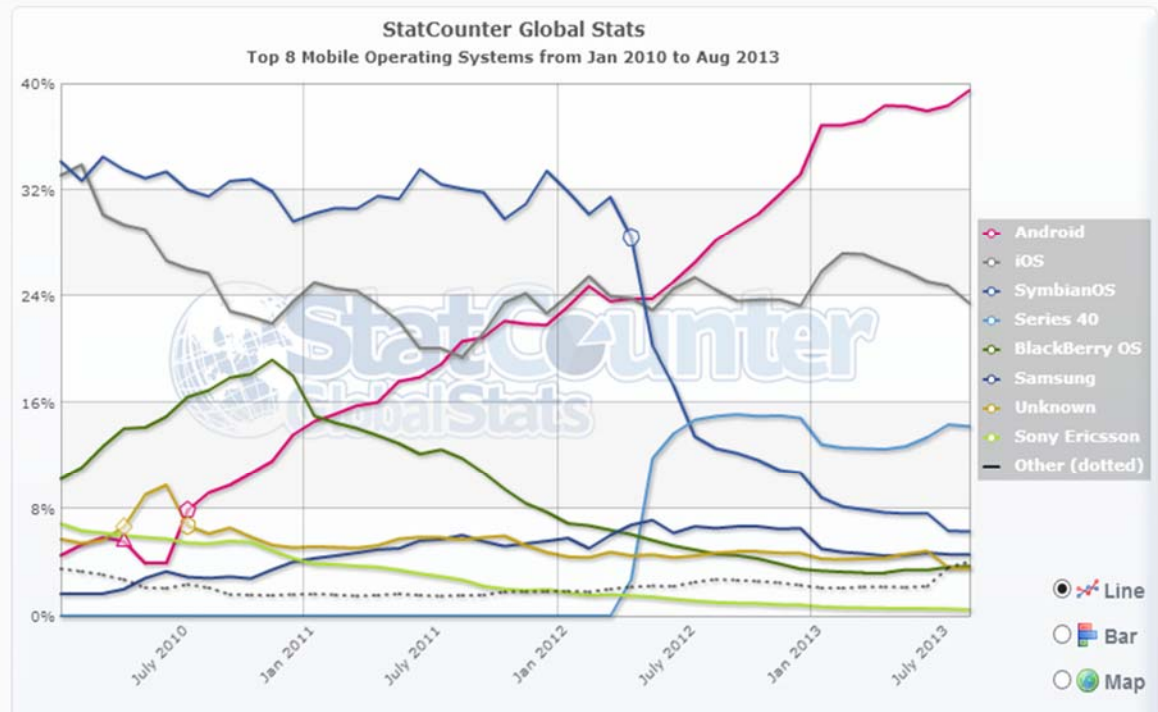

Figura 2.3. - Porcentaje de S.O. móviles utilizados a nivel mundial.
Fuente: [STA13]

En la figura se observa cómo la plataforma con mayor cuota de mercado actualmente es Android, pese a que su principal desventaja

es su juventud, ha conseguido aumentar rápidamente su cuota de mercado, pasando de contar con aproximadamente 6% de ventas en el 2010 a poseer cerca de un 39.49% a agosto del 2013 y pasar a ser el competidor más fuerte de iOS.

En Perú el escenario se muestra con Android a la cabeza de los S.O. de dispositivos móviles inteligentes, pudiendo observar en la siguiente figura la gran acogida de este S.O. en el mercado peruano:

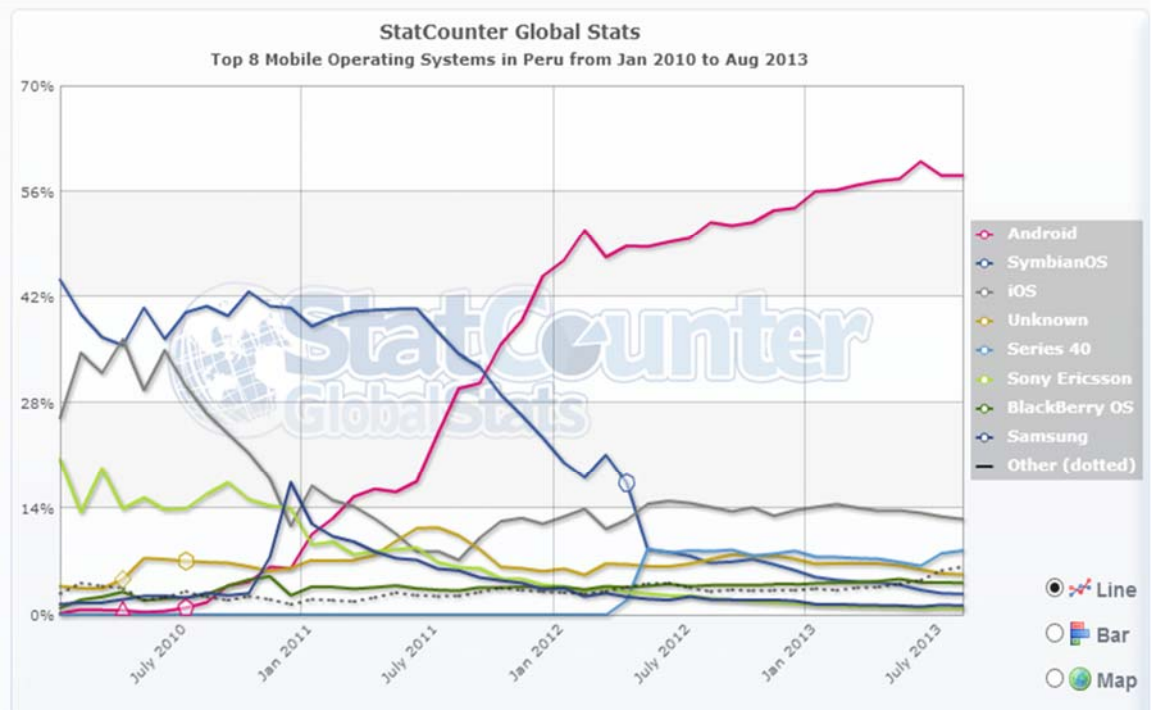

Figura 2.4. - Porcentaje de S.O. móviles utilizados en Perú.
Fuente: [STA13]

Ahora que observamos que actualmente el S.O. Android cubre una importante cuota de mercado en nuestro país, podemos inferir que la elaboración de un proyecto en base a este sistema operativo no sería de difícil implementación en nuestro medio.

- **Popularidad de los lenguajes de programación:** Para medir el índice de popularidad de los lenguajes de programación consultamos el índice de la compañía TIOBE, este se actualiza una vez al mes. La compañía TIOBE se especializa en asesorar y hacer seguimiento a la calidad del software, ellos poseen este índice que asigna ratings a cada lenguaje de programación basándose en distintos datos como son el número de programadores experimentados a nivel mundial, la cantidad de cursos online, motores de búsqueda y sitios más importantes con los cuales se calculan los ratings para cada lenguaje [TIB13].

Position Sep 2013	Position Sep 2012	Delta in Position	Programming Language	Ratings Sep 2013	Delta Sep 2012	Status
1	1	=	C	16.975%	-2.32%	A
2	2	=	Java	16.154%	-0.11%	A
3	4	↑	C++	8.664%	-0.48%	A
4	3	↓	Objective-C	8.561%	-1.21%	A
5	6	↑	PHP	6.430%	+0.82%	A
6	5	↓	C#	5.564%	-1.03%	A

Figura 2.5. - Índice TIOBE de popularidad de lenguajes de programación.
Fuente: [TIB13]

Analizando la figura anterior correspondiente al índice de popularidad de los lenguajes de programación del mes de septiembre del 2013 a nivel mundial, podemos concluir que casi el doble de programadores conocen y consultan el lenguaje de programación Java a comparación

de el lenguaje Objective-C con el cual se programan aplicaciones en el sistema operativo iOS de la empresa Apple. Vemos un escenario parecido al comparar el lenguaje de programación C++ con Java, sabiendo que el sistema operativo SymbianOS utiliza aplicaciones desarrolladas en C++ y un poco de Java, encontramos que el lenguaje de programación Java es uno de los más conocidos actualmente por programadores en la red y por lo tanto podemos llegar a la conclusión que hacer un Framework que utilice este lenguaje de programación ayudará a que mayor parte de programadores pueda entender, modificar y adaptar el presente proyecto a sus necesidades.

- **Costo de adquisición del los (SDK) Kit de desarrollo de software para las plataformas móviles y costo de suscripción como desarrollador:** Debemos saber que una vez que se adquiere la suscripción como desarrollador de cualquiera de estos sistemas operativos, se puede acceder a publicar aplicaciones en sus tiendas virtuales.

Tabla 2.1. - Costo de adquisición de SDKs para plataformas móviles y costo de suscripción como desarrollador.
Fuente: [DAP13] [DAN13] [DNO13]

Periodo	Costo suscripción como desarrollador	Costo adquisición SDK	Tienda	Sistema Operativo
Suscripción Anual	US$99	✓Gratuito ✓Suscripción para probar aplicaciones en disp. móviles. ✓Se debe contar con un AppleID.	Apple Store	iOS
Al registrarse	US$25	✓Gratuito	Android Market	Android
Al registrarse	1€	✓Gratuito	Ovi Store (SymbianOS)	SymbianOS

Al observar la tabla anterior nos damos cuenta que al adquirir los SDK para Android y SymbianOS podremos gozar de todas las facilidades que nos brinda la versión gratuita, pero al adquirir el SDK para iOS tendremos posibilidades limitadas.

Es por esto que al observar los gráficos de Sistemas Operativos más utilizados, popularidad de lenguajes de programación y la tabla de costo de adquisición de Kits de desarrollo de aplicaciones móviles, decidimos utilizar el Sistema Operativo Android para realizar el presente proyecto.

- **Fragmentación de Android y su solución:** Uno de los aspectos importantes que debemos tomar en cuenta al momento de desarrollar

aplicaciones en este S.O. es el nivel de API que debemos utilizar. Esto se debe a la fragmentación, es decir que cada versión de API, cuenta con diversas librerías que pueden cambiar o incluso ser descontinuadas, lo cual genera que nuestra aplicación pueda funcionar solo en algunas versiones, para esto se muestra la tabla de API según las versiones de Android, en la cual las versiones menores a 0.1% no se muestran.

Tabla 2.2. - Porcentaje uso de las versiones Android.
Fuente: [DEV13]

Distribución	Nivel de API	Nombre	Versión Plataforma
0.1%	4	Donut	Android 1.6
1.2%	7	Eclair	Android 2.1
2.5%	8	Froyo	Android 2.2
0.1%	9	Gingerbread	Android 2.3 - 2.3.2
33.0%	10	Gingerbread	Android 2.3.3 - 2.3.7
0.1%	13	Honeycomb	Android 3.2
22.5%	15	Ice Cream Sandwich	Android 4.0.3 - 4.0.4
34.0%	16	Jelly Bean	Android 4.1.x
6.5%	17	Jelly Bean MR1	Android 4.2.x

Es así que el uso de las diferentes versiones de Android, ocasiona el mayor desafío para los desarrolladores al momento de construir una aplicación compatible con la mayoría de dispositivos, este problema es llamado “fragmentación”, actualmente las principales versiones pueden ser graficadas proporcionalmente en la siguiente figura:

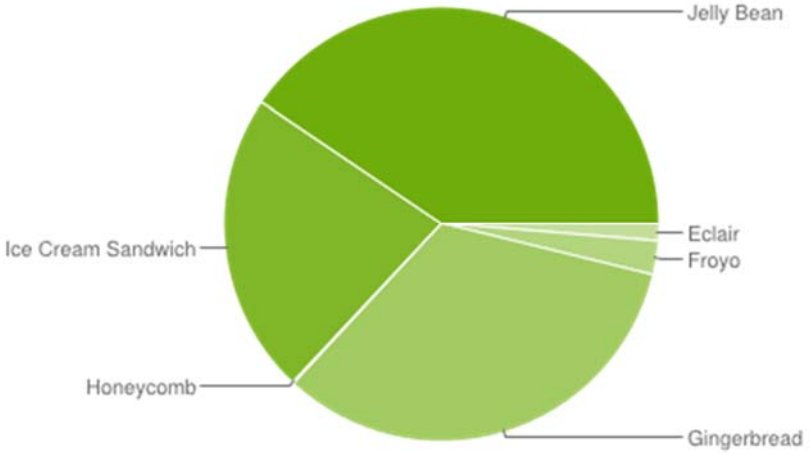

Figura 2.6. - Uso de las versiones de Android.
Fuente: [DEV13]

Tras estudiar la gráfica podemos destacar el reducido número de usuarios que utilizan las versiones 1.x (0.1%) y 2.1 (1.2%). Por lo tanto, utilizaremos como versión mínima la 2.2 para desarrollar nuestro proyecto, pues daríamos cobertura al 98,7% de los terminales y podríamos combatir así el continuo problema de fragmentación.

2.2.5. Web Service

Los servicios Web son componentes de software que se comunican utilizando tecnologías Web basadas en estándares como HTTP y XML.

Los Servicios Web están diseñados para ser "consumidos" por otras aplicaciones, variando en la complejidad de sus operaciones, desde la comprobación de una cuenta bancaria, saldo en línea, hasta los procesos más

complejos como sistemas de planificación de recursos empresariales (ERP), CRM (Customer Relationship Management).

El hecho de basarse en estándares abiertos hace que los Web Services puedan interoperar con sistemas con hardware, lenguaje de programación y sistema operativo distinto. Esto significa que las aplicaciones escritas en diferentes lenguajes de programación y que se ejecuten en diferentes plataformas pueden intercambiar datos de manera transparente a través de intranets o Internet utilizando Web Services. [ECW13]

2.2.6. JSON

JavaScript Object Notation, es un subconjunto de la notación de objetos de Javascript, básicamente, es un formato ligero para el intercambio de datos. Es completamente independiente del lenguaje de programación pero fácilmente interpretado en lenguaje y estructura por desarrolladores de lenguajes C, C++, C#, Java, Javascript, etc.

Al momento de comparar JSON con XML, la representación de datos en XML es inherentemente pesada, pues usa bastantes etiquetas de apertura y cierre que son innecesarias aumentando el peso de carga de la aplicación. En el mundo de las aplicaciones, recortar algunos bytes del proceso de carga puede mejorar dramáticamente el rendimiento de las aplicaciones y reducir el uso de plan de datos para los usuarios que utilizan alguna operadora.

Debemos tener en cuenta que los datos XML son difíciles de convertir, es así que se usa DOM (Document Object Model) y SAX (Simple APIs for XML) en computadoras desktop, pero para plataformas móviles usar estos recursos sería muy caro computacionalmente y en términos de capacidad de memoria. Es así que al comparar XML con JSON, concluimos que este último es un formato ideal para el intercambio de datos en nuestro proyecto. [MSI10]

2.2.7. Framework

Es una estructura de soporte definida, una colección organizada de clases que constituyen un diseño reutilizable para que otro proyecto de software puede ser organizado y desarrollado. Contiene un conjunto de librerías, componentes de software y directrices arquitectónicas que provee al implementador las herramientas necesarias para la creación de una aplicación de principio a fin.

Según [GUT05], un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

2.2.8. GUI

Conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando imágenes y formas gráficas (botones, íconos, ventanas, etc. que representan funciones, acciones e información). [INT06]

Para el caso particular de Android, estos controles son propios del sistema operativo y no descienden del lenguaje nativo Java, a medida de su evolución, Android ha mejorado y definido un conjunto de buenas prácticas para el diseño de aplicaciones basándose en la aparición de nuevos dispositivos y características que se pueden aprovechar para ofrecer una mejor experiencia al usuario. [DEV13] Cabe destacar que aunque la documentación de Android referente al tema es abundante, existen muy pocas investigaciones respecto a la creación de controles de manera dinámica, es decir en tiempo de ejecución.

CAPÍTULO 3

DESARROLLO DE LA PROPUESTA

En el presente capítulo desarrollaremos la propuesta del proyecto presentando sus principales características así como la descripción del análisis realizado.

3.1. ANÁLISIS

Para obtener un proceso de desarrollo exitoso, será necesario utilizar un marco de trabajo adecuado, en el cual se podrá definir la estructura y planificación del proceso de desarrollo. Este marco de trabajo nos debe brindar la oportunidad de lograr un desarrollo iterativo, obtener un modelado visual del software y por lo tanto verificar su calidad en un posterior proceso, es por ello que decidimos utilizar RUP (Proceso Unificado de Rational).

En RUP los cinco flujos de trabajo: requerimientos, análisis, diseño, implementación y prueba, tienen lugar sobre las cuatro fases: concepción, elaboración, construcción y transición.

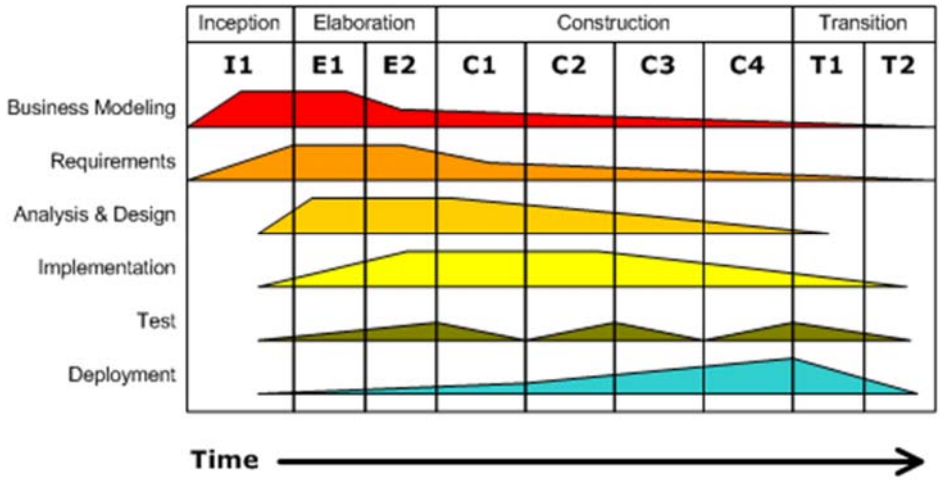

Figura 3.1. - Desarrollo Iterativo.
Fuente: [DUT07]

Para este proyecto se aplican las cuatro fases de RUP, las cuales consisten en:

- **Fase Concepción:** Identificaremos y definiremos los requerimientos utilizados en el proyecto.
- **Fase de Elaboración:** Se realizará el análisis especificando los casos de uso, diagramas de clases, diagramas de secuencia, etc. y así mismo se expondrá el diseño de la arquitectura del proyecto.
- **Fase de Construcción:** Desarrollaremos el software basándonos en la arquitectura diseñada y especificada.
- **Fase de transición:** Realizaremos las pruebas del software.

Para especificar el marco de trabajo utilizado en este proyecto, mostraremos la actividad fundamental a la que pertenece cada artefacto que será desarrollado.

Tabla 3.1. - Marco de trabajo organizado en actividades aplicadas al proyecto.
Fuente: Elaboración propia.

Artefacto	Actividad
Diagrama de casos de uso y especificación	Requerimientos
Diagrama de paquetes Diagrama de clases	Análisis
Diagrama de componentes Modelo entidad relación Diagramas de secuencia	Diseño
Diagrama de despliegue Interfaces	Implementación
Pruebas del Framework	Pruebas

3.1.1. Diagramas de casos de uso y especificación

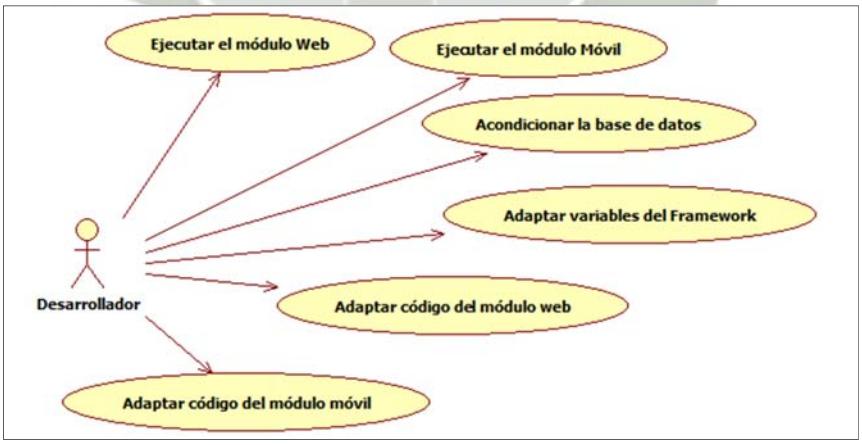

Figura 3.2. - Diagrama de Casos de Uso del Framework propuesto.
Fuente: Elaboración propia.

Tabla 3.2. - Descripción de Caso de Uso: Ejecutar el módulo Web
Fuente: Elaboración propia.

Caso de Uso	**Ejecutar el módulo Web**
Descripción:	El desarrollador copia los archivos que constituyen el Framework en el servidor donde vaya a situar la parte web del sistema a desarrollar, luego deberá ejecutar y probar que tenga completo acceso a la pantalla principal de este módulo.
Actores:	Desarrollador
Precondición:	1. Identificar la carpeta que contiene el módulo web del Framework. 2. Configurar el servidor IIS. 3. Verificar que el servicio se encuentre operativo.
Flujo:	1. El desarrollador copia la carpeta que contiene el módulo web del Framework en la ruta correcta del Servidor a utilizar.
Postcondición:	El desarrollador ingresa a la URL: http://localhost:80/datileapp, si está utilizando el puerto por defecto del servidor utilice el 80, en caso contrario utilice el configurado para este módulo.

Tabla 3.3. - Descripción de Caso de Uso: Ejecutar el módulo Móvil
Fuente: Elaboración propia.

Caso de Uso	**Ejecutar el módulo móvil**
Descripción:	El desarrollador copia la carpeta del módulo móvil del Framework, luego deberá importar, ejecutar y probar la aplicación móvil.
Actores:	Desarrollador
Precondición:	1. Identificar la carpeta que contiene el módulo móvil del Framework. 2. Tener instalado el entorno de desarrollo Eclipse. 3. Verificar que se tenga instalada una versión de API aceptada por el proyecto ingresando al “Android SDK Manager” del entorno de desarrollo Eclipse. 4. Importar el proyecto Android.
Flujo:	1. El desarrollador ingresa al entorno de desarrollo eclipse. 2. Selecciona la opción File en el menú de herramientas, luego la opción Import y seguidamente “Existing Projects into Workspace”. 3. Selecciona la carpeta donde se encuentra el módulo móvil del Framework y presiona aceptar. 4. Ejecutar el proyecto en un dispositivo móvil Android.
Postcondición:	Probar la aplicación utilizando un dispositivo con API compatible al proyecto.

Tabla 3.4. - Descripción de Caso de Uso: Acondicionar la base de datos
Fuente: Elaboración propia.

Caso de Uso	**Acondicionar la base de datos**
Descripción:	El propósito de este caso de uso es crear y acondicionar la base de datos para luego ejecutar los procedimientos almacenados pertenecientes al Framework.
Actores:	Desarrollador
Precondición:	1. Instalar el sistema para la gestión de bases de datos SQL SERVER Express. 2. Reconocer el archivo que contiene los procedimientos almacenados, este archivo se encuentra dentro de la carpeta de archivos del Framework.
Flujo:	1. Crear la base de datos. 2. Crear tablas de la base de datos. 3. Ejecutar los procedimientos almacenados.
Postcondición:	Es recomendable que se analice la estructura de la base de datos del Framework para poder seguidamente agregar alguna tabla, campo o procedimiento.

Tabla 3.5. - Descripción de Caso de Uso: Adaptar variables del Framework
Fuente: Elaboración propia.

Caso de Uso	**Adaptar variables del Framework**
Descripción:	El desarrollador necesita adaptar a su entorno las variables del Framework, por ejemplo direcciones donde se referencia al servidor, el Framework contiene por defecto la del localhost pero necesitará ser modificada, usuario y contraseña de base de datos, etc.
Actores:	Desarrollador
Precondición:	1. Se necesita tener instalado el sistema para la gestión de bases de datos SQL SERVER Express. 2. Tener el módulo web de Framework correctamente instalado. 3. Tener el módulo móvil del Framework correctamente instalado.
Flujo:	1. Seguir los pasos conforme al tutorial para cambiar las variables del módulo web. 2. Seguir los pasos conforme al tutorial para cambiar las variables del módulo móvil.
Postcondición:	Se debe probar que funcionen correctamente las conexiones.

Tabla 3.6. - Descripción de Caso de Uso: Adaptar código del módulo web
Fuente: Elaboración propia.

Caso de Uso	**Adaptar código del Framework**
Descripción:	El desarrollador adapta el código del módulo web a sus necesidades si es necesario.
Actores:	Desarrollador
Precondición:	1. Ejecutar correctamente el módulo web. 2. Acondicionar la base de datos. 3. Adaptar las variables del Framework.
Flujo:	1. Para mayor entendimiento de los archivos que componen esta parte del Framework consultar el Manual del Usuario. 2. Implementar o cambiar archivos adaptando el código a las necesidades de la aplicación.
Postcondición:	Se adaptará el código del módulo web y el sistema podrá entrar en funcionamiento.

Tabla 3.7. - Descripción de Caso de Uso: Adaptar código del módulo móvil
Fuente: Elaboración propia.

Caso de Uso	**Adaptar código del Framework**
Descripción:	El desarrollador adapta el código del módulo móvil a sus necesidades si es necesario.
Actores:	Desarrollador
Precondición:	1.Ejecutar correctamente la aplicación móvil. 2.Adaptar las variables del Framework.
Flujo:	1.Para mayor entendimiento de los archivos que componen esta parte del Framework consultar el Manual del Usuario. 2.Implementar o cambiar archivos adaptando el código a las necesidades de la aplicación.
Postcondición:	Se adaptará el código del módulo móvil y el sistema podrá entrar en funcionamiento.

3.1.2. Diagrama de Paquetes

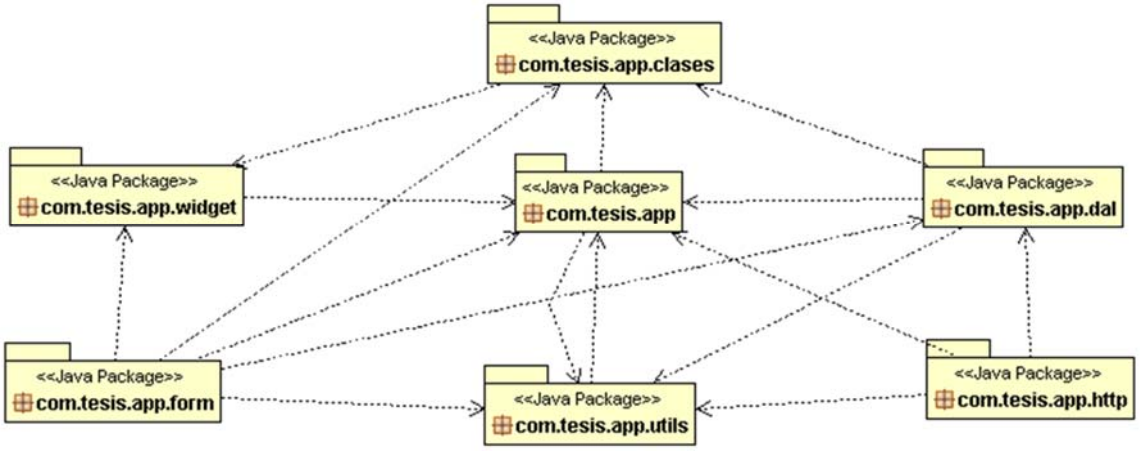

Figura 3.3. - Diagrama de Paquetes del Framework propuesto.
Fuente: Elaboración propia.

3.1.3. Diagrama de clases Módulo web

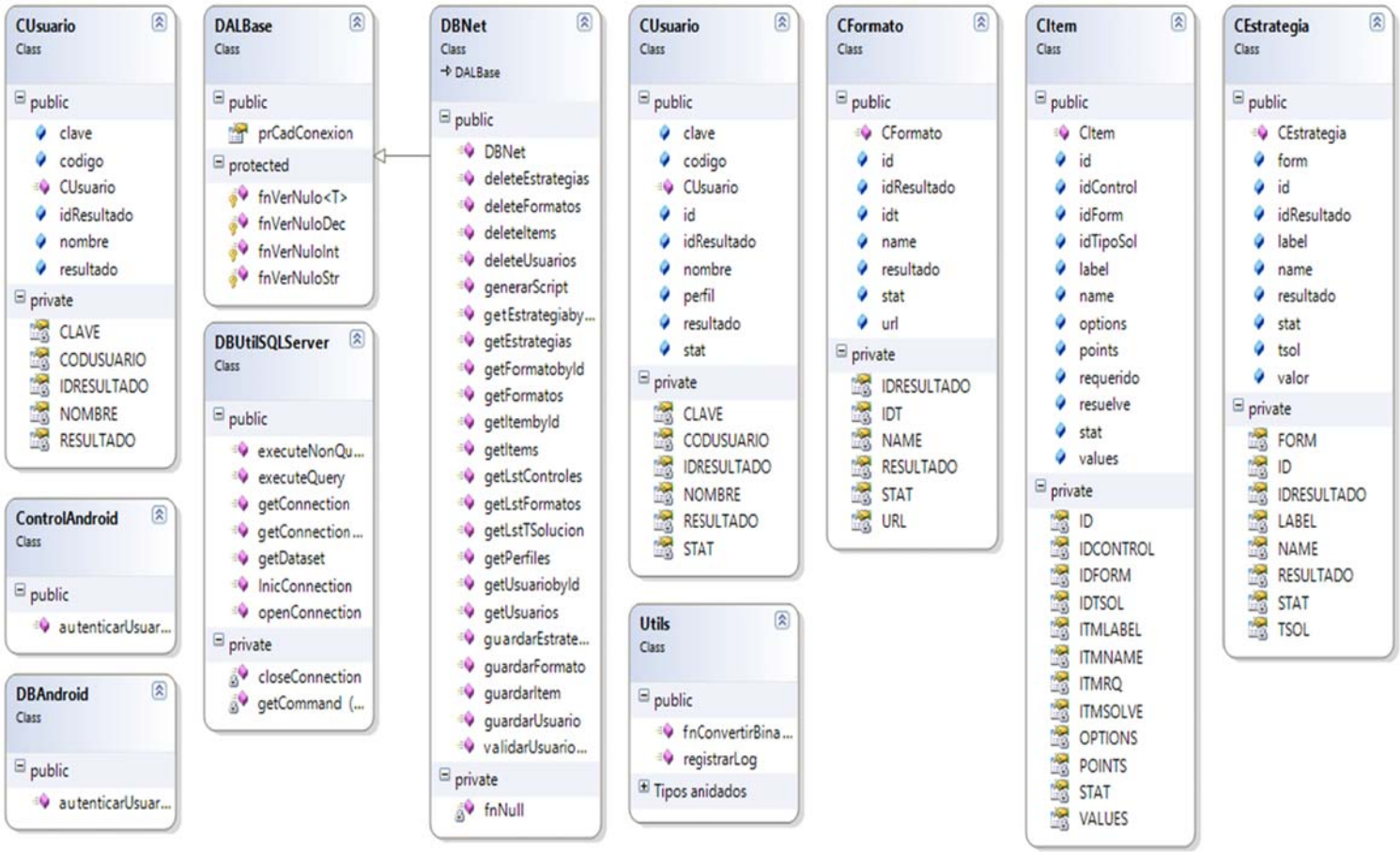

Figura 3.4. - Diagrama de Clases del Framework propuesto (Módulo Web).
Fuente: Elaboración propia.

3.1.4. Diagrama de Clases Módulo móvil: Se muestran sólo las principales clases de este módulo.

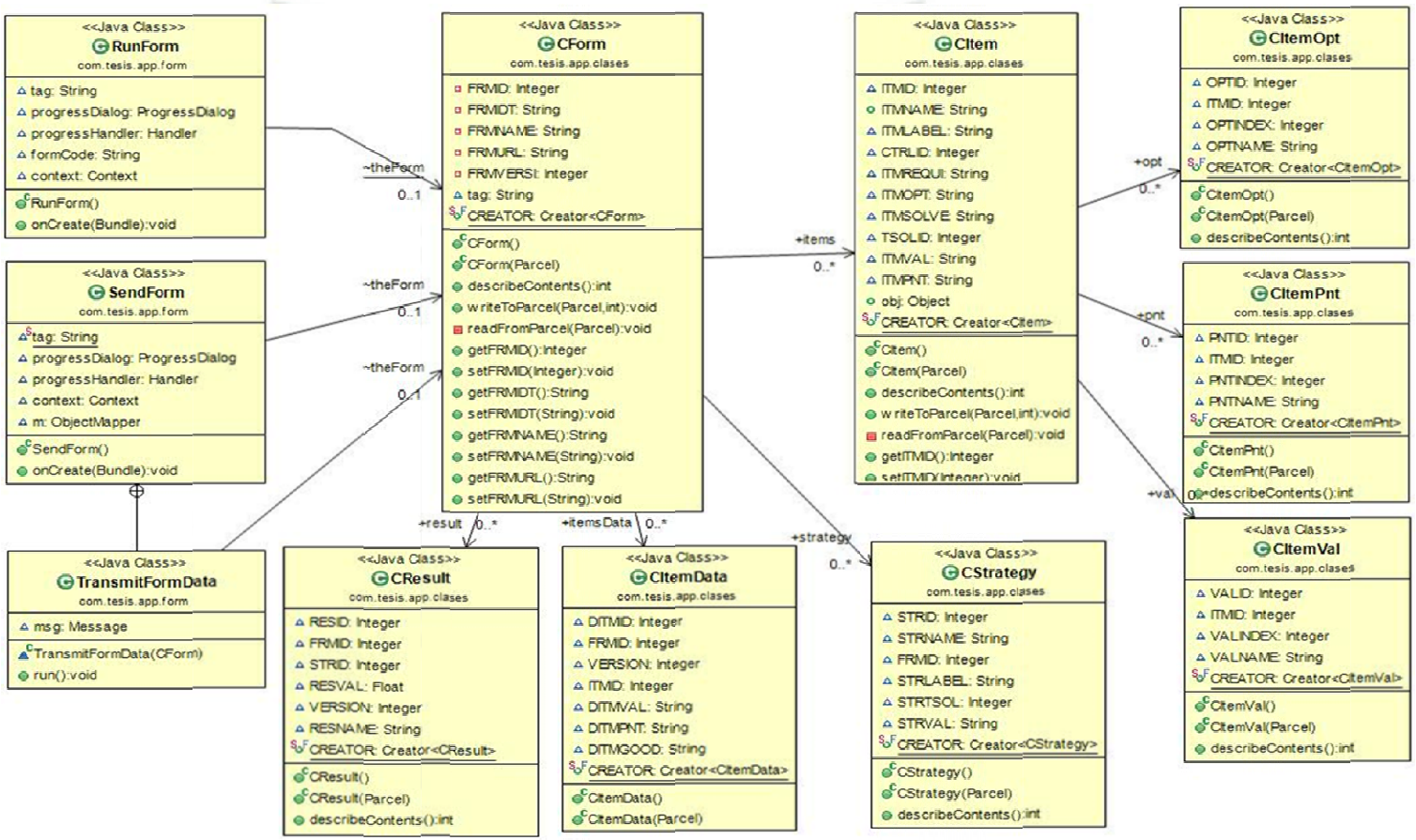

Figura 3.5. - Diagrama de Clases del Framework propuesto (Módulo móvil).
Fuente: Elaboración propia.

3.2. DISEÑO

Se describirán aspectos relacionados a la arquitectura y prototipos.

3.2.1. Diseño de la arquitectura

El Framework consta de 3 componentes: el cliente, el servidor y la base de datos, cada uno de ellos con tareas específicas que se detallarán a continuación.

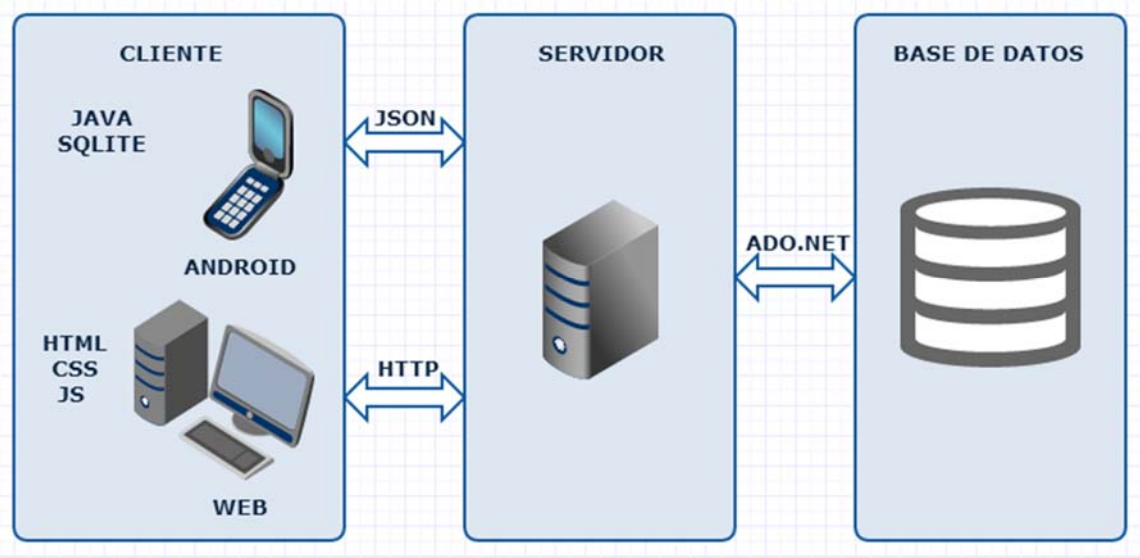

Figura 3.6. - Arquitectura del Framework / Diagrama de componentes.
Fuente: Elaboración propia.

- **Cliente móvil:** Es el dispositivo móvil con sistema operativo Android, donde está instalada la aplicación Datile, perteneciente al Framework, en ella se ejecuta el algoritmo de creación dinámica de componentes gráficos así como el proceso de sincronización y envío de datos mediante la

conexión Wi-Fi o 3G. Utiliza SQLite para el almacenamiento de datos localmente y así permitir el trabajo fuera de cobertura.

- **Cliente web:** En este se lleva a cabo la creación y parametrización de formularios y componentes gráficos. Ingresa a la parte web del Framework mediante un browser conectado a internet.

- **Servidor:** Está compuesto por procedimientos para la comunicación con el dispositivo móvil y a la vez un conjunto de páginas web que permiten la comunicación con usuarios web.

- **Base de Datos:** Contiene las tablas y procedimientos almacenados necesarios para lograr la persistencia y consulta de datos adecuada.

3.2.2. Modelo entidad relación

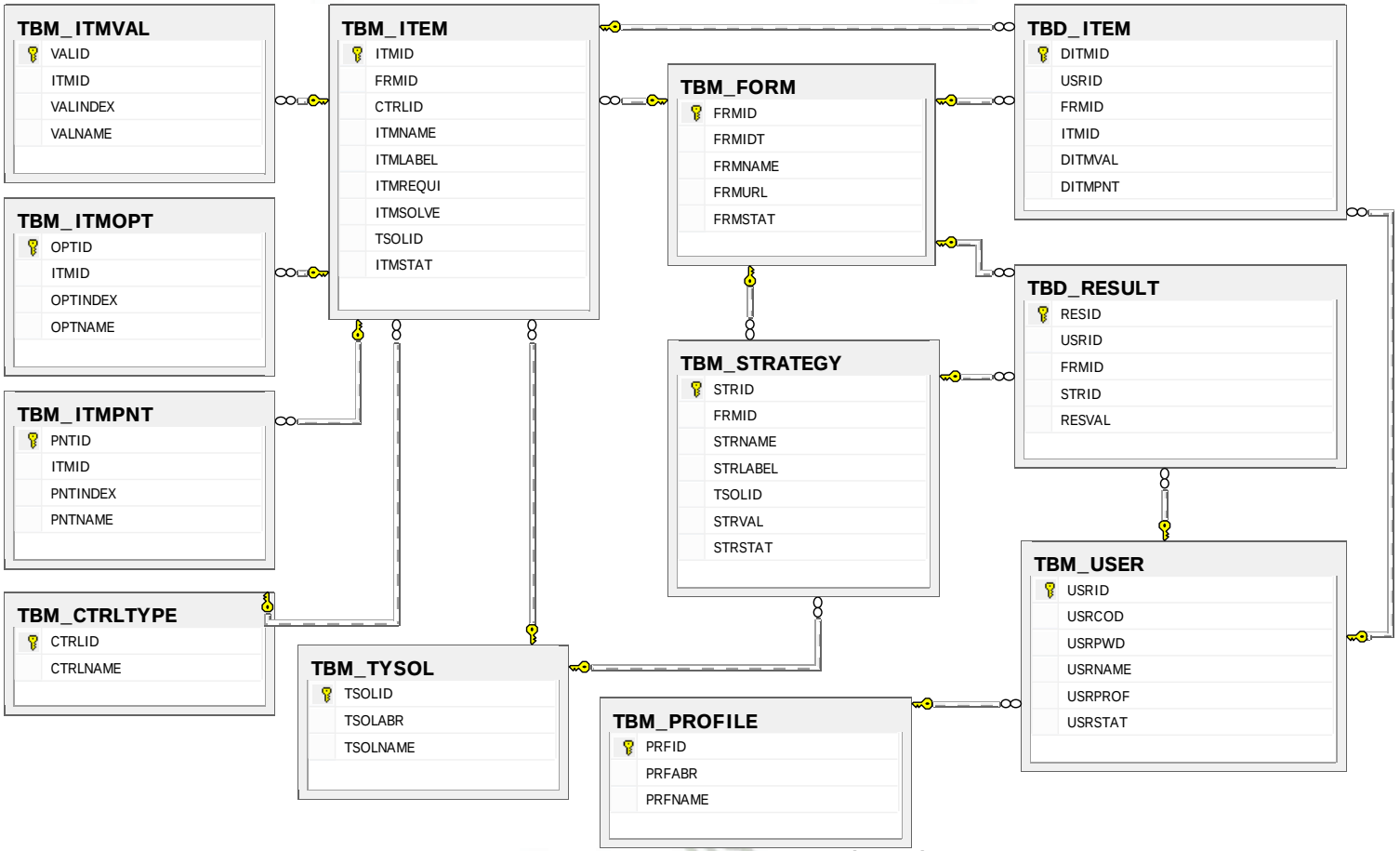

Figura 3.7. - Diagrama Entidad Relación (Módulo web).
Fuente: Elaboración propia.

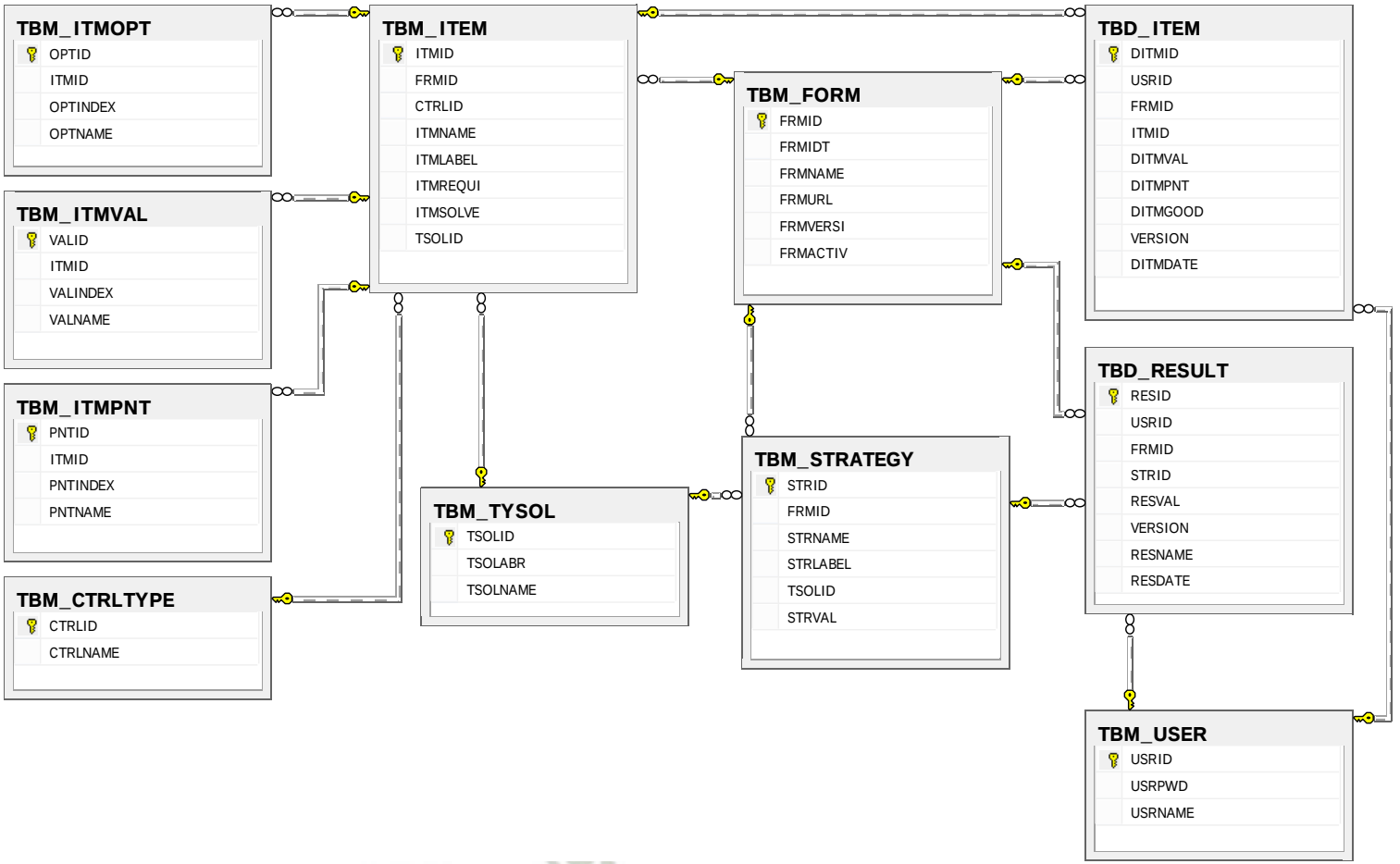

Figura 3.8. - Diagrama Entidad Relación (Módulo móvil).
Fuente: Elaboración propia.

3.2.3. Diagramas de Secuencia

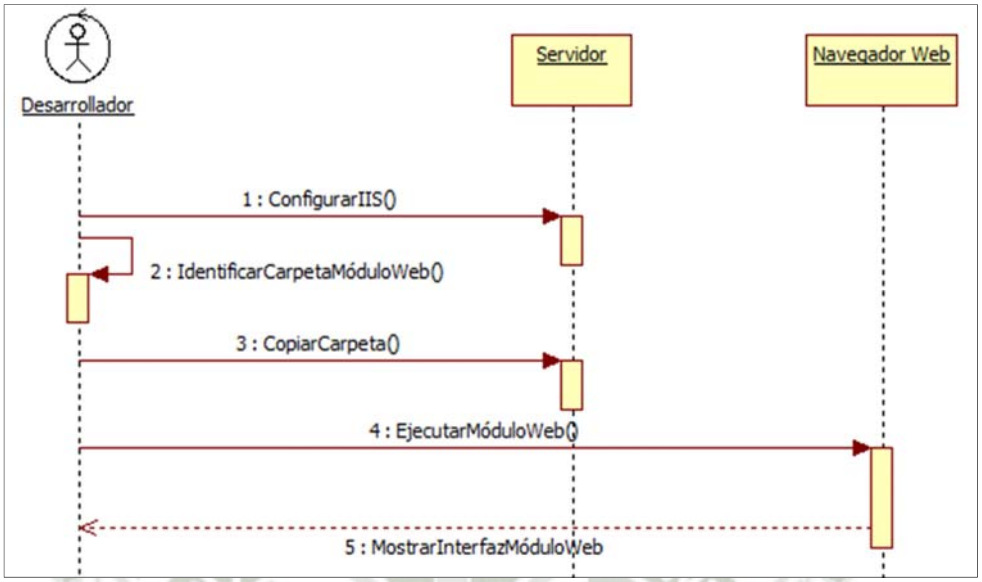

Figura 3.9. - Diagrama de Secuencia - Ejecutar módulo Web.
Fuente: Elaboración propia.

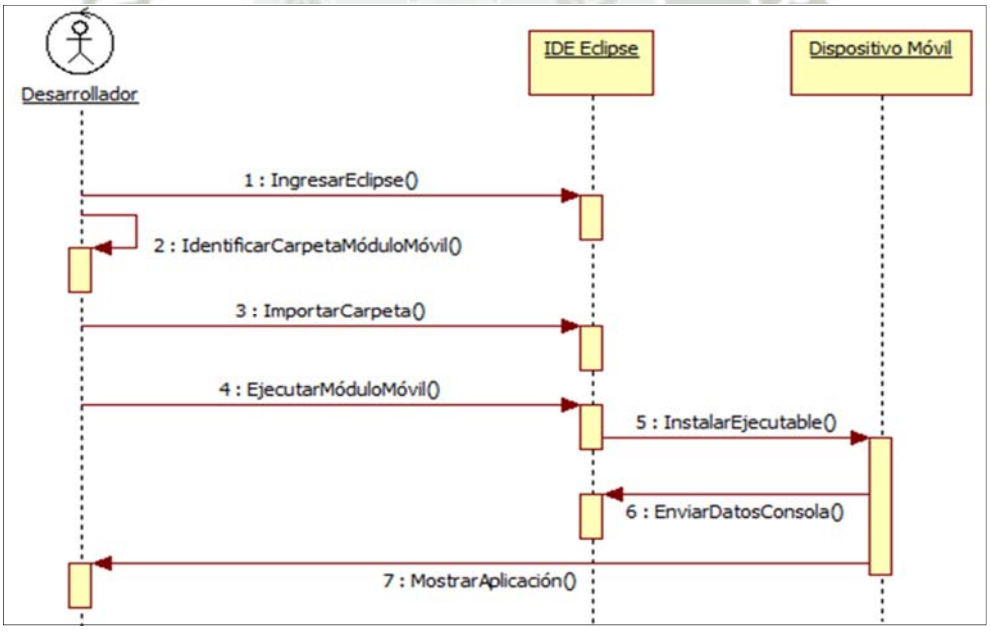

Figura 3.10. - Diagrama de Secuencia - Ejecutar módulo Móvil.
Fuente: Elaboración propia.

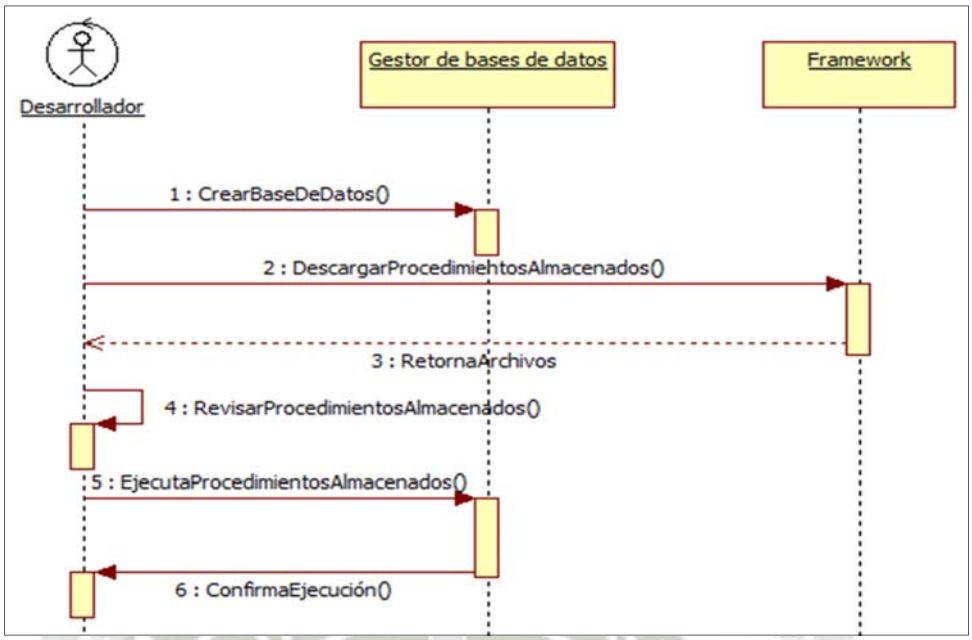

Figura 3.11. - Diagrama de Secuencia - Acondicionar base de datos.
Fuente: Elaboración propia.

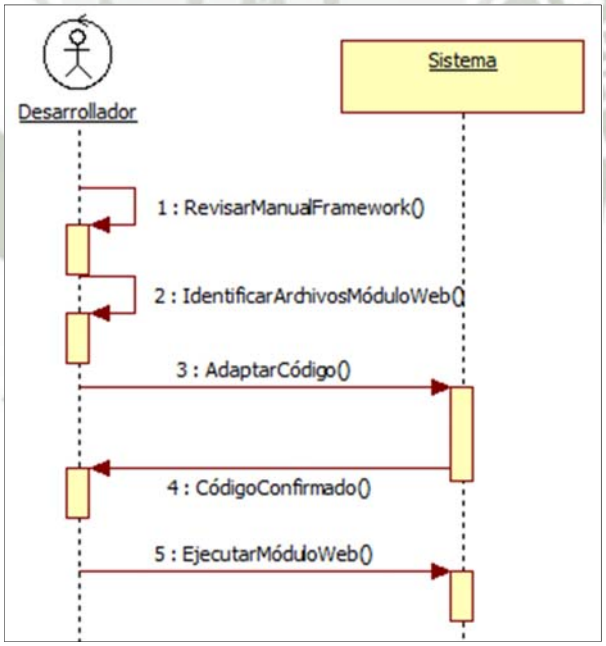

Figura 3.12. - Diagrama de Secuencia - Adaptar código del módulo web.
Fuente: Elaboración propia.

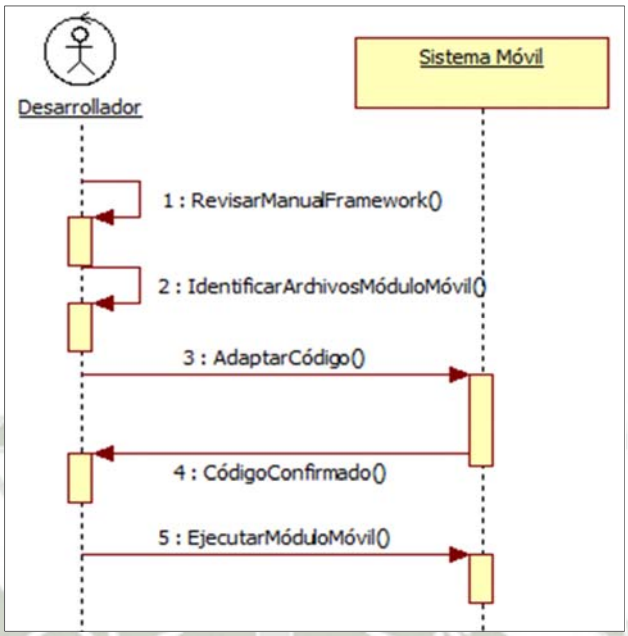

Figura 3.13. - Diagrama de Secuencia - Adaptar código del módulo móvil.
Fuente: Elaboración propia.

3.3. DESARROLLO E IMPLEMENTACIÓN

Una vez descritos los objetivos y características principales de nuestra aplicación, explicaremos los procedimientos necesarios para su elaboración, así como la tecnología y componentes necesarios para llevar a cabo su implementación.

3.3.1. Diagrama de despliegue

El diagrama de despliegue modela los nodos y sus relaciones. Por cada nodo podemos especificar los componentes de software que se despliegan en él. Es así que demostramos el diagrama de despliegue.

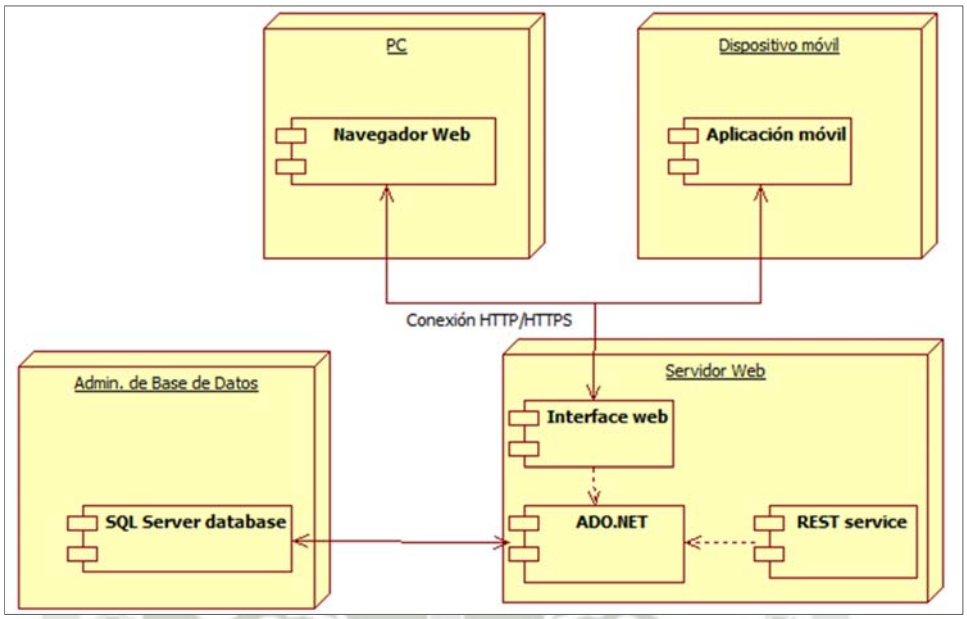

Figura 3.14. - Diagrama de despliegue.
Fuente: Elaboración propia.

3.3.2. Tecnología utilizada

Para el presente proyecto elegimos utilizar la arquitectura Cliente/Servidor, este es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. [SOM05], es así que utilizaremos las siguientes tecnologías:

a) Lenguaje de Programación: Para el caso del cliente (aplicación móvil) utilizaremos Java, siendo éste el lenguaje nativo para el desarrollo de aplicaciones para dispositivos con sistema operativo Android, y para la parte del servidor utilizaremos C# y ASP.NET con el IDE Visual Studio 2005, por ofrecer bastante flexibilidad en conexión con el gestor de base de datos

elegido, poseer una baja curva de aprendizaje, ofrecer una interfaz agradable para el desarrollo y facilitar la programación y depuración.

b) Tecnología Web Service: La comunicación entre cliente y servidor se realizará mediante JSON, se utilizará esta tecnología por ser mucho más liviana que XML por lo tanto optimiza los tiempos de comunicación. Adicionalmente en el cliente y en el servidor se utilizarán librerías que permitirán la elaboración de los mensajes JSON.

c) Sistema Administrador de Base de Datos: Utilizaremos SQL Server 2005 en su versión Express, la comunicación con el servidor se realizará mediante ADO.NET, para el almacenamiento, modificación y extracción de datos según las peticiones del cliente utilizaremos procedimientos almacenados, evitando así el uso de codificación SQL en el servidor, y protegiendo la aplicación de posibles ataques externos. SQL Server provee herramientas intuitivas para la administración de las bases de datos que reducen la complejidad y soporte de las aplicaciones.

En el caso de usar otros manejadores de base de datos, se debe modificar el archivo Web.config del servidor, debe utilizarse la cadena de conexión adecuada según el manejador que se utilice, se debe añadir los controladores que permitan la conexión de .NET con el manejador utilizado, y finalmente reproducir las tablas y procedimientos almacenados en el manejador.

3.3.3. Descripción de Archivos - Código fuente del Framework

Para mayor especificación acerca de los archivos y sus funciones, ver Anexo A.

Para mayor especificación acerca del código fuente, ver Anexo B.

3.3.4. Tablas y procedimientos almacenados del Framework

En esta sección se describen las tablas y la utilización de los procedimientos almacenados necesarios para el funcionamiento del Framework, para consultar la especificación, ver Anexo C.

3.3.5. Estructura de la aplicación generada como resultado del Framework

Como el Framework trabaja sobre la arquitectura Cliente - Servidor, la aplicación generada trabajará bajo las mismas características, sin embargo, debe tenerse en cuenta las necesidades y el funcionamiento de la aplicación que se desee parametrizar.

3.3.6. Interfaces.

- **Web: Autenticación.**

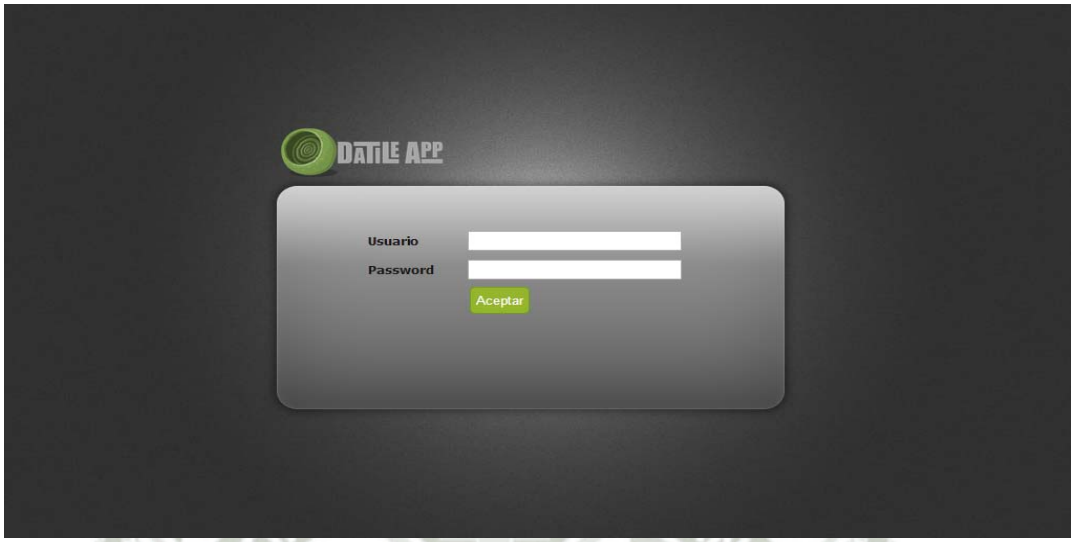

Figura 3.15. - Módulo Web: Pantalla de Logueo.
Fuente: Elaboración propia.

- **Web: Mantenimiento de Usuarios**

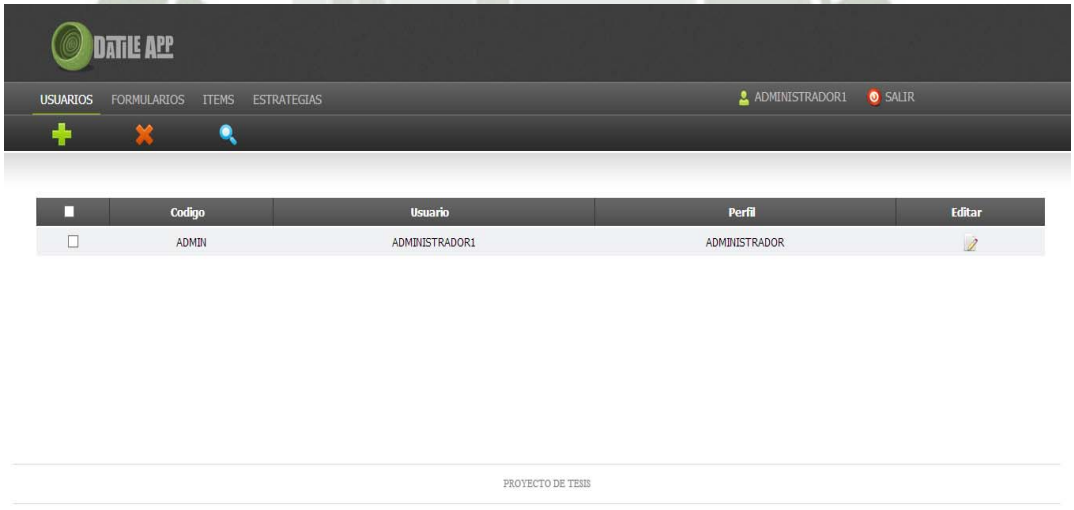

Figura 3.16. - Módulo Web: Mantenimiento de Usuarios.
Fuente: Elaboración propia.

- **Web: Mantenimiento de Formularios**

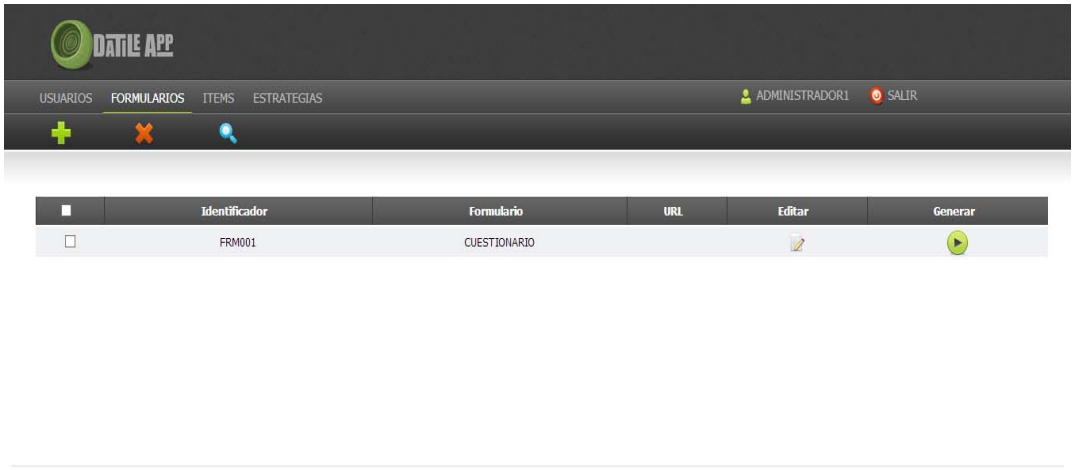

Figura 3.17. - Módulo Web: Mantenimiento de Formularios.
Fuente: Elaboración propia.

- **Web: Mantenimiento de Items**

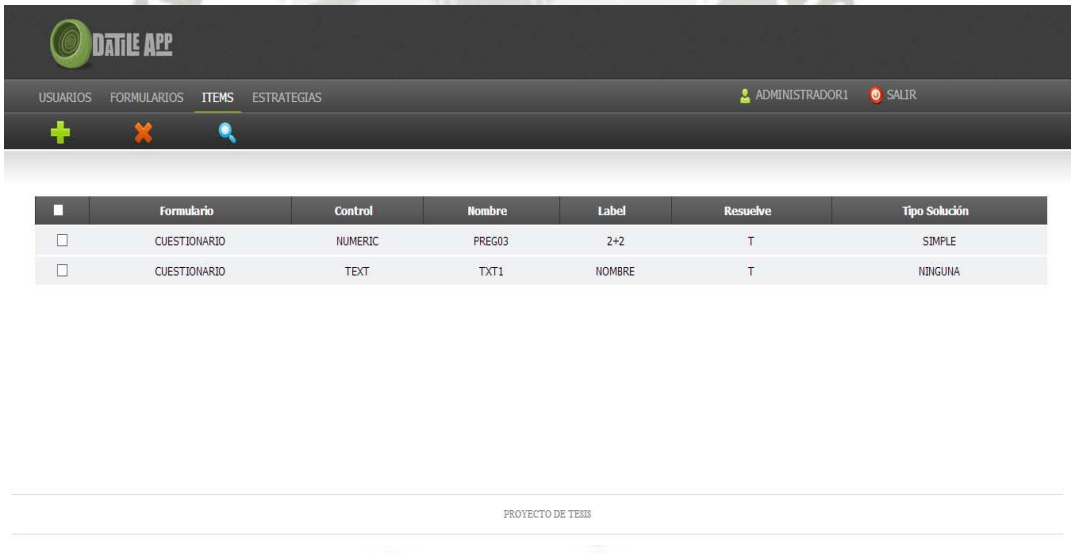

Figura 3.18. - Módulo Web: Mantenimiento de Items.
Fuente: Elaboración propia.

- **Web: Mantenimiento de Estrategias.**

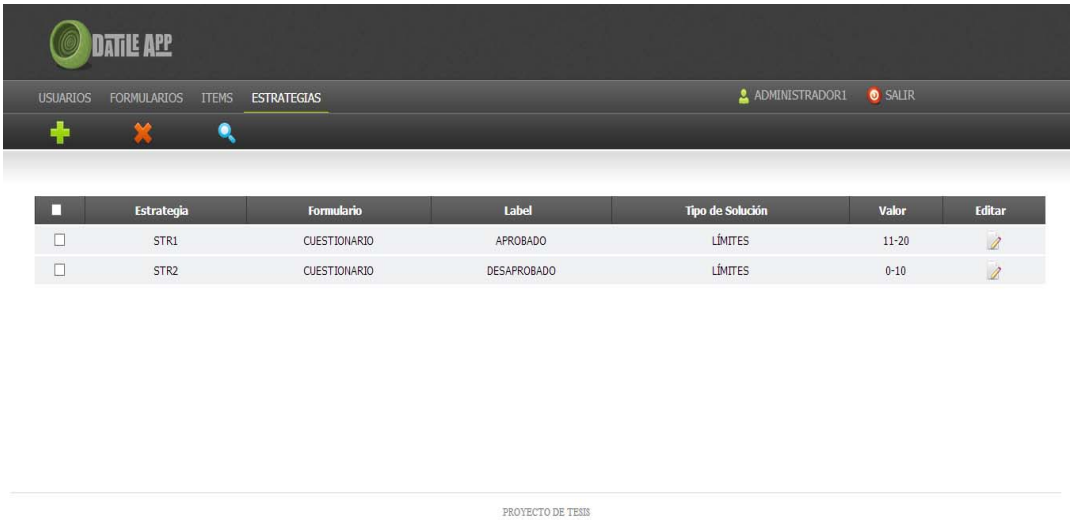

Figura 3.19. - Módulo Web: Mantenimiento de Estrategias
Fuente: Elaboración propia.

- **Móvil: Pantalla de carga:** Esta pantalla se ejecuta al iniciar la aplicación.

Figura 3.20. - Módulo Móvil: Pantalla de Inicio.
Fuente: Elaboración propia.

- **Móvil: Menú principal:** Esta pantalla contiene elementos parametrizados en la clase “Configuración” y agregados al momento de la ejecución, lo cual hace de la construcción de un menú sumamente flexible. Si no se cuenta con señal Wi-Fi o 3g al momento de sincronizar, se le notificará.

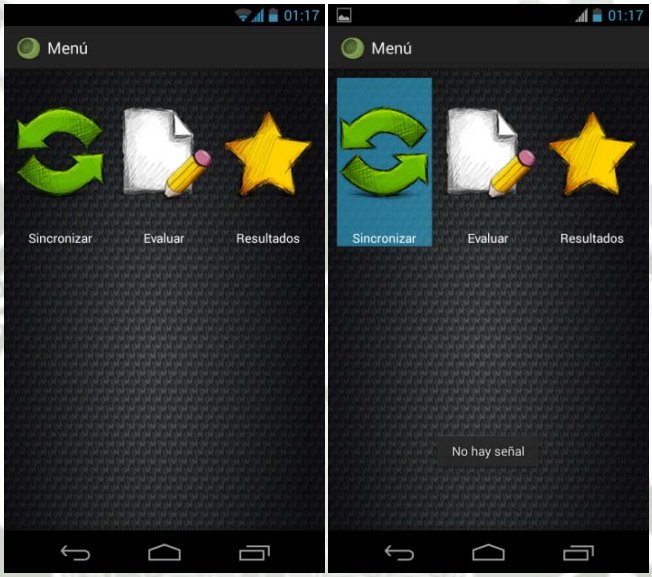

Figura 3.21. - Módulo Móvil: Menú principal.
Fuente: Elaboración propia.

- **Móvil: Selección sincronizar:** Al presionar el botón “Sincronizar”, debemos ingresar el código único del formulario a sincronizar y luego presionar “Obtener Formulario”, lo primero que se hace es una validación para determinar si el campo es nulo y si no, se procede a realizar la sincronización. Como complemento tenemos la posibilidad de leer códigos QR.

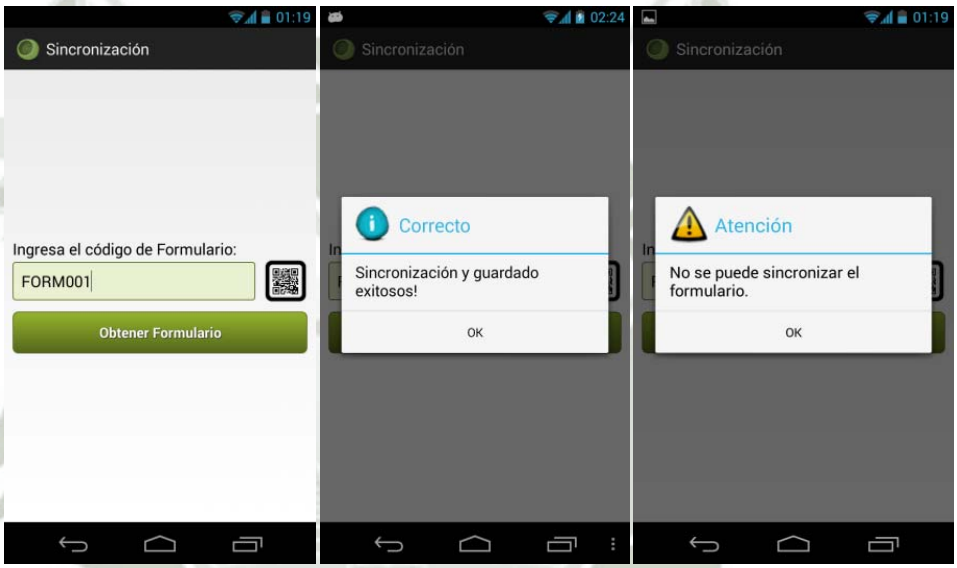

Figura 3.22. - Módulo Móvil: Sincronizar.
Fuente: Elaboración propia.

- **Móvil: Selección evaluar:** Al presionar el botón “Evaluar”, debemos seleccionar el nombre del formulario a responder y luego presionar “Cargar Formulario” para proceder a la creación dinámica de componentes gráficos, y así poder visualizar el formulario creado en la parte web.

Figura 3.23. - Módulo Móvil: Evaluar.
Fuente: Elaboración propia.

- **Móvil: Selección resultados:** Al presionar el botón “Resultados”, debemos seleccionar el nombre del conjunto de respuestas de los diversos formularios respondidos.

Figura 3.24. - Módulo Móvil: Resultados.
Fuente: Elaboración propia.

- **Móvil: Evaluación:** Se encarga de la creación dinámica de componentes gráficos y mostrar el formulario creado en la parte web.

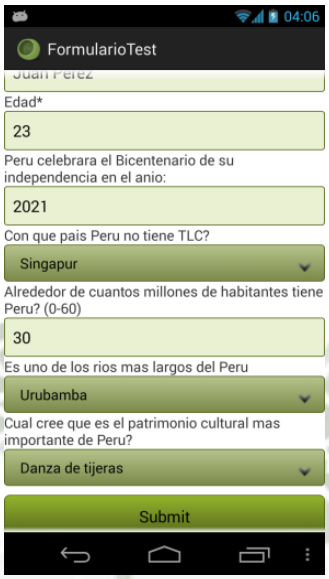

Figura 3.25. - Módulo Móvil: Evaluación.
Fuente: Elaboración propia.

- **Móvil: Mostrar resultados:** En una lista se muestran los resultados gráficamente. Se tiene la oportunidad de cambiar el nombre a un formulario ya resuelto, de forma que podrá ser consultado posteriormente.

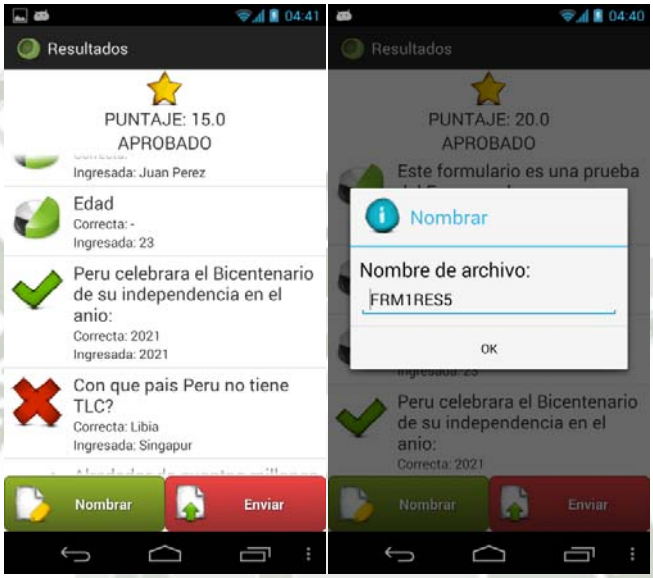

Figura 3.26. - Módulo Móvil: Mostrar Resultados.
Fuente: Elaboración propia.

- **Móvil: Envío de datos:** Envía los resultados al servidor y nos muestra su respuesta.

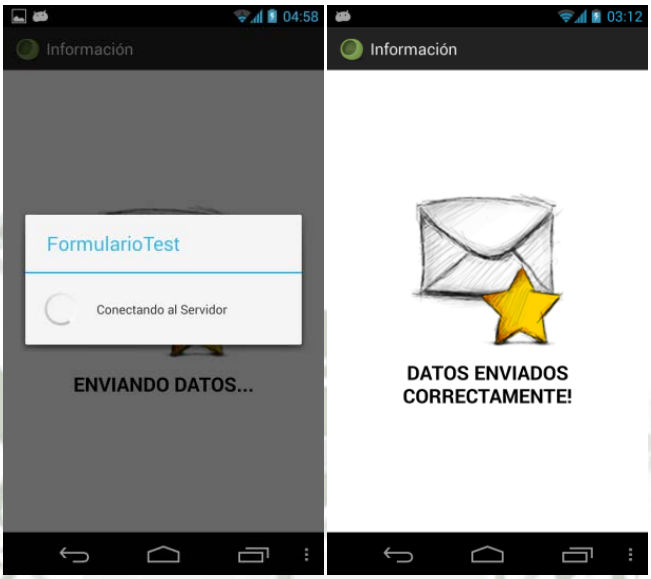

Figura 3.27. - Módulo Móvil: Envío de datos.
Fuente: Elaboración propia.

CAPÍTULO 4

CASO DE ESTUDIO

En el presente capítulo se proponen dos casos de estudio para ello desarrollaremos la aplicación de la propuesta del proyecto presentando sus principales características, esto demostrará la capacidad de parametrizar varios formularios sin necesidad de reinstalar la aplicación y generando un único instalador.

4.1. INTRODUCCIÓN

Para demostrar el funcionamiento del Framework propuesto, se instanciará la aplicación APP DATILE mediante la cual se generará dos formularios, uno para el caso de sólo toma de datos y el otro para ejemplificar un formulario con resolución de ítems y aplicación de estrategias.

4.2. MARCO GENERAL DE LA APP DATILE

Para ejemplificar los dos casos mencionados previamente se realiza un análisis para identificar los ítems necesarios y que serán parametrizados desde la interfaz web.

Se utilizan artefactos UML para poder explicar los pasos necesarios para poder utilizar el Framework y generar una aplicación en base a la parametrización.

Formulario 1: Solicitud de crédito.

Tabla 4.1. - Ítems Formulario 1: Solicitud de Crédito
Fuente: Elaboración propia.

Nombre	Label	Opciones	Val.	Pnt.	Control	Tip. Sol.	Req.	Res.
INFO	Formulario para solicitud de créditos.	-	-	-	Label	-	-	-
NOMRZ	Nombre/Razón Social	-	-	-	Text	-	Si	-
DNIRUC	DNI/RUC	-	-	-	Text	-	Si	-
EDAD	Edad	-	-	-	Numeric	-	Si	No
DIRECCION	Dirección	-	-	-	Text	-	Si	-
UBICACION	Ubicación	-	-	-	GPS	-	No	-
TLFCEL	Telf/Cel	-	-	-	Text	-	Si	-
ACTIVIDAD	Actividad	Abarrotes Ferreteria Agricultura	-	-	Choice	-	Si	No
TIPOCRED	Tipo de crédito	Pequeña empresa Personal Vivienda	-	-	Choice	-	Si	No
MONTO	Monto	-	-	-	Numeric	-	Si	No
PLAZO	Plazo(Meses)	-	-	-	Numeric	-	Si	No
OBSERV	Observaciones	-	-	-	Text	-	Si	-

Formulario 2: Cuestionario.

Tabla 4.2. - Ítems Formulario 2: Cuestionario
Fuente: Elaboración propia.

Nombre	Label	Opciones	Valor	Pnt.	Control	Tip. Sol.	Req.	Res.
INFO	Formulario Cultura General Perú	-	-	-	Label	-	-	-
NOMBRE	Nombres y Apellidos	-	-	-	Text	-	Si	-
EDAD	Edad	-	-	-	Numeric	-	No	No
BICENT	Perú celebrará el Bicentenario de su independencia en el año:	-	2021	5	Numeric	Simple	Si	Si
TLC	¿Con que país Perú no tiene TLC?	Singapur Libia China USA	-	0 5 0 0	Choice	Simple	Si	Si
HABIT	¿Alrededor de cuántos millones de habitantes tiene Perú? (0-60)	-	0-20 21-40 41-60	0 5 0	Numeric	Límite	Si	Si
RIOS	Es uno de los ríos más largos del Perú	Urubamba Amazonas Chili Majes	-	5 5 0 0	Choice	Simple	Si	Si
PATR	¿Cuál cree que es el patrimonio cultural más importante de Perú?	Danza de tijeras Señor de los Milagros Caballo de paso	-	-	Choice	-	No	No

4.3. DESARROLLO

4.3.1. Diagrama de paquetes

Para cubrir todas las áreas de la aplicación la dividiremos en paquetes destinados a agrupar clases por medio de sus funcionalidades, los paquetes utilizados serán: Usuario, Parametrización, Sincronización y Resolutor.

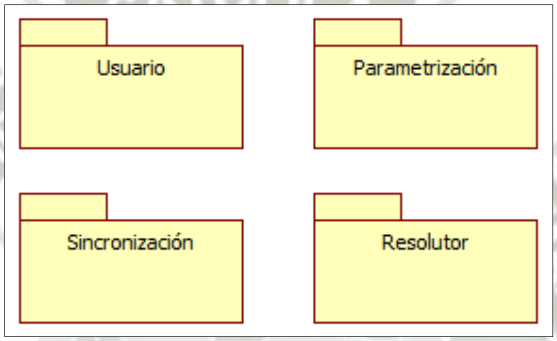

Figura 4.1. - Diagrama de paquetes.
Fuente: Elaboración propia.

4.3.2. Requerimientos de la aplicación

Paquete: Usuario

Tabla 4.3. - Requerimientos para el Paquete: Usuario
Fuente: Elaboración propia.

Código	Descripción
R1.W1	El módulo web permitirá validar los datos del usuario Administrador, antes de permitir el ingreso a la interfaz web.
R1.W2	El módulo web permitirá crear, modificar y anular usuarios.

Paquete: Parametrización

Tabla 4.4. - Requerimientos para el Paquete: Parametrización
Fuente: Elaboración propia.

Código	Descripción
R2.W1	El módulo web permitirá la creación, modificación y eliminación de formularios de toma de datos.
R2.W2	Permitir la creación, modificación y eliminación de ítems.
R2.W3	El módulo web permitirá la asignación de valores y límites asociados a cada ítem de evaluación.
R2.W4	El módulo web permitirá la parametrización de controles de interfaz gráfica asociados a cada ítem de evaluación.
R2.W5	El módulo web permitirá la creación de estrategias asociadas a los resultados de los formularios creados.

Paquete: Sincronización

Tabla 4.5. - Requerimientos para el Paquete: Sincronización
Fuente: Elaboración propia.

Código	Descripción
R3.M1	La aplicación móvil permitirá la sincronización de los datos correspondientes a formatos, ítems y estrategias.
R3.M2	La aplicación móvil permitirá la creación dinámica de los controles de interfaz gráfica en el entorno de trabajo del usuario.

Paquete: Resolutor

Tabla 4.6. - Requerimientos para el Paquete: Resolutor
Fuente: Elaboración propia.

Código	Descripción
R4.M1	La aplicación permitirá el grabado de los datos ingresados por el usuario móvil.
R4.M2	La aplicación permitirá la resolución de cada ítem parametrizado, guardando el resultado localmente.
R4.M3	La aplicación permitirá el envío de los resultados en formato JSON hacia el servidor.

4.3.3. Diagramas de casos de uso y especificación

Paquete: Usuario

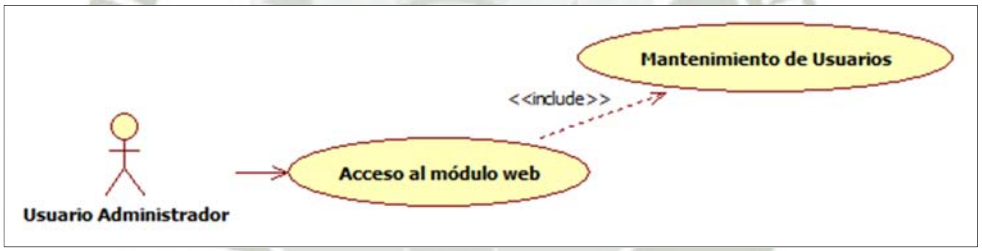

Figura 4.2. - Diagrama de Casos de Uso: Usuario
Fuente: Elaboración propia.

Tabla 4.7. - Descripción de Caso de Uso: Acceso al módulo web
Fuente: Elaboración propia.

Caso de uso	**Acceso al módulo Web**
Descripción:	El propósito de este caso de uso es permitir el acceso al módulo web, evaluando el alias de identificación y password ingresados por el usuario.
Actores:	Usuario Administrador
Precondición:	Existencia de al menos un Usuario Administrador en la Base de Datos.
Flujo:	1. El Usuario Administrador ingresa su alias de identificación y contraseña. 2. El módulo web evaluará los datos ingresados, comparándolos con los guardados en la base de datos. 3. Si el usuario está registrado y la contraseña es válida, el módulo permitirá el acceso, en caso contrario se denegará el mismo.
Postcondición:	Ingreso al módulo web.

Tabla 4.8. - Descripción de Caso de Uso: Mantenimiento de Usuarios
Fuente: Elaboración propia.

Caso de uso	**Mantenimiento de usuarios**
Descripción:	El propósito de este caso de uso es realizar mantenimiento a los usuarios existentes.
Actores:	Usuario Administrador
Precondición:	Ingreso válido del usuario al módulo web.
Flujo:	1. Inicia cuando el usuario autorizado selecciona la opción “Usuarios”. 2. Se mostrará la lista de usuarios existentes. 3. Para crear un usuario se presionará el botón “Nuevo” y se ingresarán los datos solicitados, para almacenar estos datos se presionará el botón “Guardar”. 4. Para modificar los datos de un usuario, se seleccionará el botón “Editar” perteneciente a cada registro. 5. Para eliminar los datos de un usuario, se seleccionará el registro o los registros y presionará el botón “Eliminar”.
Postcondición:	Creación, modificación y/o anulación de usuarios.

Paquete: Parametrización

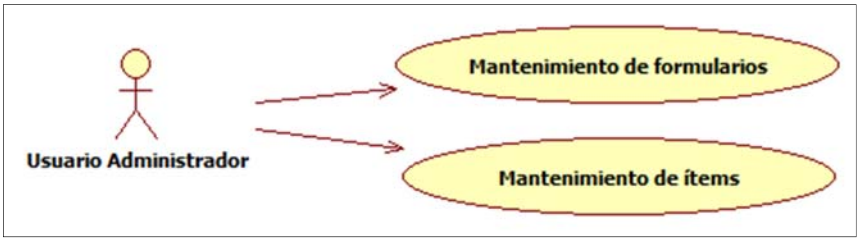

Figura 4.3. - Diagrama de Casos de Uso: Parametrización
Fuente: Elaboración propia.

Tabla 4.9. - Descripción de Caso de Uso: Mantenimiento de formularios
Fuente: Elaboración propia.

Caso de uso	**Mantenimiento de formularios**
Descripción:	El propósito de este caso de uso es realizar mantenimiento a los formularios.
Actores:	Usuario Administrador
Precondición:	Ingreso válido del usuario al módulo web.
Flujo:	1. Inicia cuando el usuario autorizado selecciona la opción "Formulario". 2. Se mostrará la lista de formularios existentes. 3. Para crear un formulario se presionará el botón "Nuevo" y se ingresarán los datos solicitados, para almacenar estos datos se presionará el botón "Guardar". 4. Para modificar los datos de un formulario, se seleccionará el botón "Editar" perteneciente a cada registro. 5. Para eliminar los datos de un formulario, se seleccionará el registro o los registros y presionará el botón "Eliminar".
Postcondición:	Creación, modificación y/o anulación de formularios.

Tabla 4.10. - Descripción de Caso de Uso: Mantenimiento de Ítems
Fuente: Elaboración propia.

Caso de uso	**Mantenimiento de ítems**
Descripción:	El propósito de este caso de uso es realizar mantenimiento a los ítems.
Actores:	Usuario Administrador
Precondición:	Ingreso válido del usuario al módulo web.
Flujo:	1. Inicia cuando el usuario autorizado selecciona la opción “Ítems”. 2. Se mostrará la lista de ítems existentes. 3. Para crear un ítem se presionará el botón “Nuevo” y se ingresarán los datos solicitados, para almacenar estos datos se presionará el botón “Guardar”. 4. Para eliminar los datos de un ítem, se seleccionará el registro o los registros y presionará el botón “Eliminar”.
Postcondición:	Creación y/o anulación de ítems de evaluación.

Paquete: Sincronización

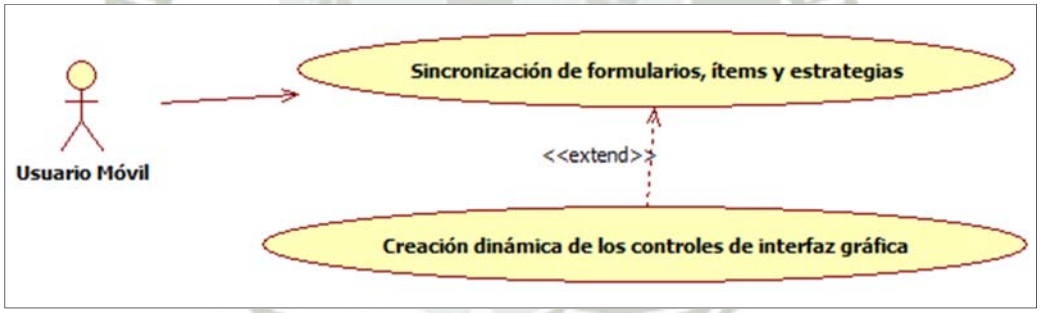

Figura 4.4. - Diagrama de Casos de Uso: Sincronización
Fuente: Elaboración propia.

Tabla 4.11. - Descripción de Caso de Uso: Sincronización de formularios, ítems y estrategias
Fuente: Elaboración propia.

Caso de uso	**Sincronización de formularios, ítems y estrategias**
Descripción:	El propósito de este caso de uso es sincronizar los datos de los formularios, ítems y estrategias que se hayan creado previamente.
Actores:	Usuario móvil
Precondición:	Deberá existir al menos un formularios e ítem parametrizado.
Flujo:	1. El usuario deberá seleccionar la opción “Sincronizar” del menú principal.
Postcondición:	Los formularios, ítems y estrategias se guardarán en el dispositivo móvil.

Tabla 4.12. - Descripción de Caso de Uso: Creación dinámica de controles
Fuente: Elaboración propia.

Caso de uso	**Creación dinámica de los controles de interfaz gráfica**
Descripción:	El propósito de este caso de uso es la creación dinámica de la interfaz gráfica de la aplicación basada en la sincronización de formularios.
Actores:	Usuario Móvil
Precondición:	El usuario deberá haber sincronizado al menos un formulario previamente.
Flujo:	1. El usuario seleccionará la opción “Iniciar”. 2. Se ejecutará el algoritmo de creación dinámica, obteniendo los ítems sincronizados y creando los respectivos controles.
Postcondición:	Se crearán dinámicamente los controles de interfaz gráfica.

Paquete: Resolutor

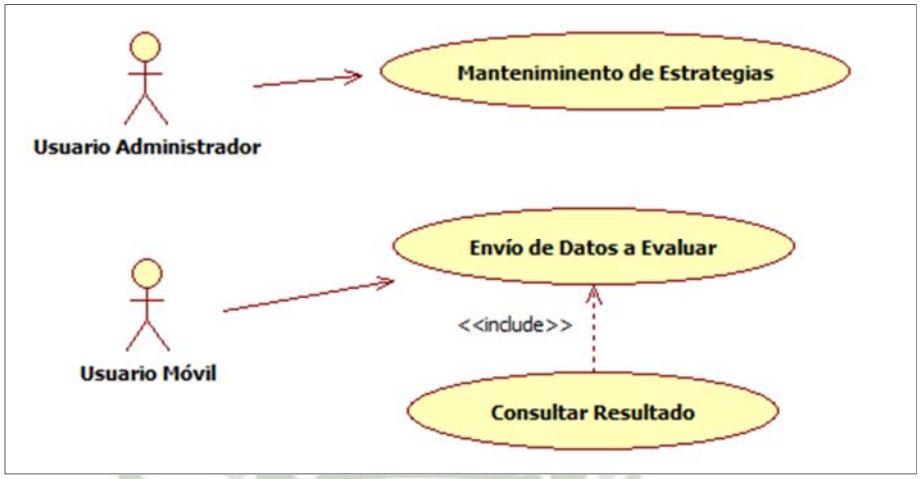

Figura 4.5. - Diagrama de Casos de Uso: Resolutor
Fuente: Elaboración propia.

Tabla 4.13. - Descripción de Caso de Uso: Mantenimiento de estrategias
Fuente: Elaboración propia.

Caso de uso	**Mantenimiento de estrategias**
Descripción:	El propósito de este caso de uso es la creación, modificación y eliminación de estrategias, asociadas a cada formulario.
Actores:	Usuario Administrador
Precondición:	El formulario asociado debe haber sido creado previamente.
Flujo:	1. Inicia cuando el usuario autorizado selecciona la opción "Estrategias" del menú principal. 2. Se podrá parametrizar puntajes máximos y mínimos para ser asignados a una decisión en concreto, es decir, se podrá crear una estrategia para cada rango de puntajes. 3. Se podrán crear, modificar y eliminar tantas estrategias como el usuario vea por conveniente.
Postcondición:	Las estrategias para los puntajes resultantes de la evaluación serán creadas.

Tabla 4.14. - Descripción de Caso de Uso: Envío de datos a evaluar
Fuente: Elaboración propia.

Caso de uso	**Envío de los datos a evaluar**
Descripción:	El propósito de este caso de uso es el envío de datos ingresados por el usuario móvil para que el Resolutor los evalúe.
Actores:	Usuario Móvil
Precondición:	Deben haberse ingresado todos los datos solicitados.
Flujo:	1. Al seleccionar el botón Enviar se pedirán enviarán los datos para completar la evaluación o toma de datos. 2. Al finalizar se enviarán los datos al servidor.
Postcondición:	Datos enviados y registrados.

Tabla 4.15. - Descripción de Caso de Uso: Consultar resultado
Fuente: Elaboración propia.

Caso de uso	**Consultar resultado.**
Descripción:	El propósito de este caso de uso es la recepción del puntaje resultante de la evaluación de riesgo crediticio y así mismo la visualización de la estrategia recomendada.
Actores:	Usuario Móvil.
Precondición:	Existencia de datos analizados por el Resolutor.
Flujo:	1. Una vez enviados los datos desde la aplicación móvil al Resolutor, éste resolverá los valores de los ítems a evaluar emitiendo un puntaje. 2. Según el puntaje emitido para el cliente evaluado, se mostrará también en la pantalla la estrategia recomendada.
Postcondición:	Puntaje y estrategia resultante emitidos por la aplicación móvil.

4.3.4. Diagrama de clases: Se consideran las principales clases del módulo móvil.

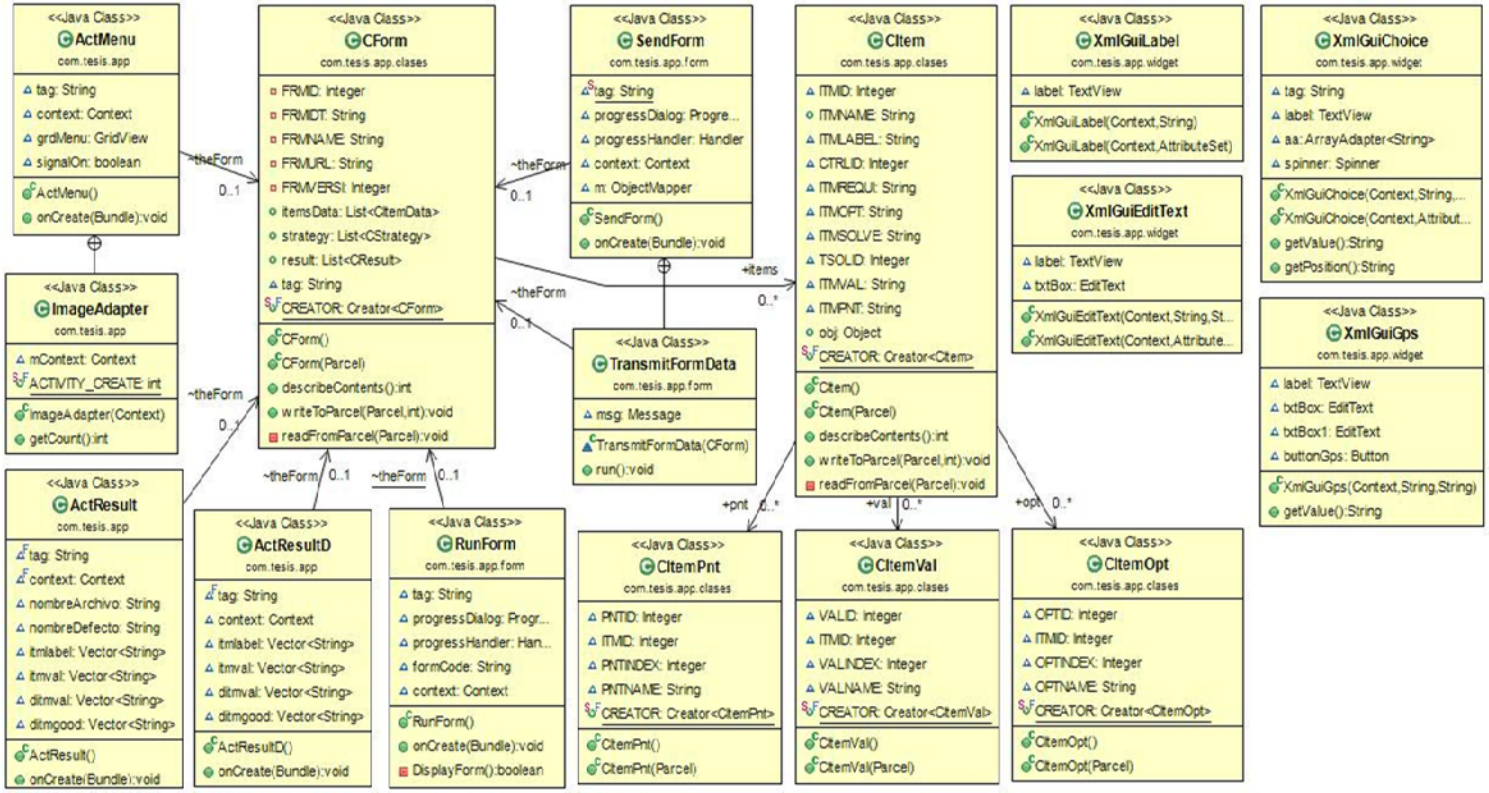

Figura 4.6. - Diagrama de Clases: Módulo móvil
Fuente: Elaboración propia.

4.3.5. Diagrama de estados

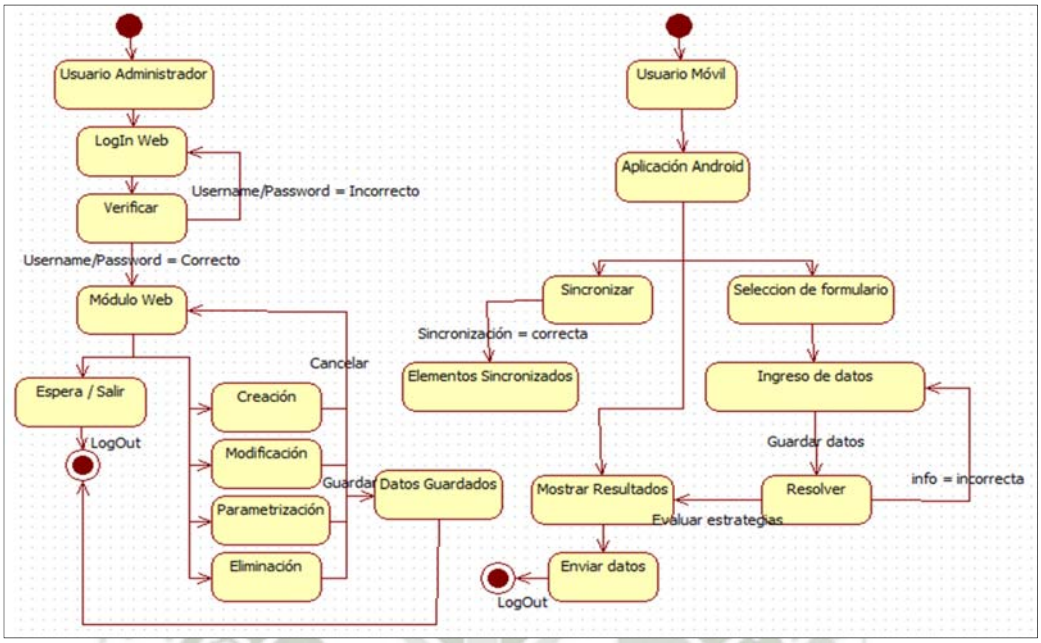

Figura 4.7. - Diagrama de Estados
Fuente: Elaboración propia.

4.3.6. Diagramas de Secuencia

4.3.6.1. Diagramas de Secuencia - Web

- Diagrama de Secuencia - Autenticación

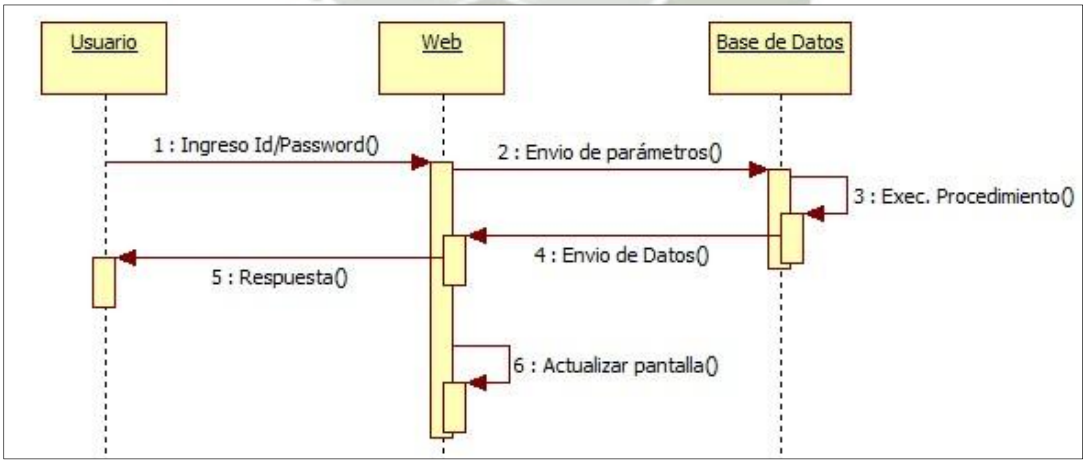

Figura 4.8. - Diagrama de secuencia: Autenticación
Fuente: Elaboración propia.

- Diagrama de Secuencia - Mantenimiento de usuarios

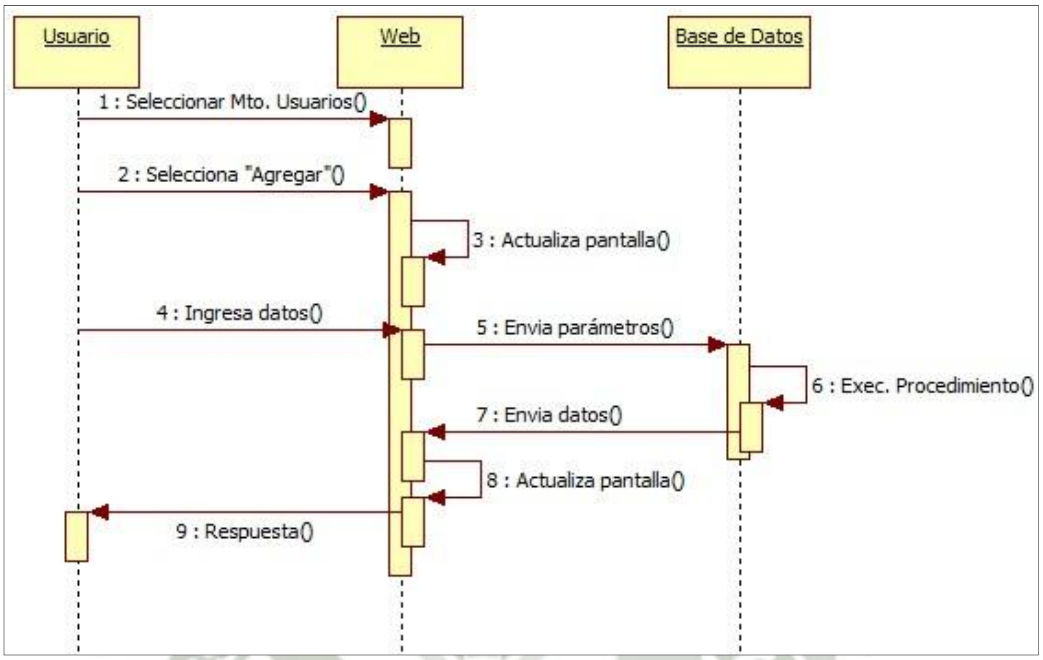

Figura 4.9. - Diagrama de Secuencia: Mantenimiento de Usuarios - Alta
Fuente: Elaboración propia.

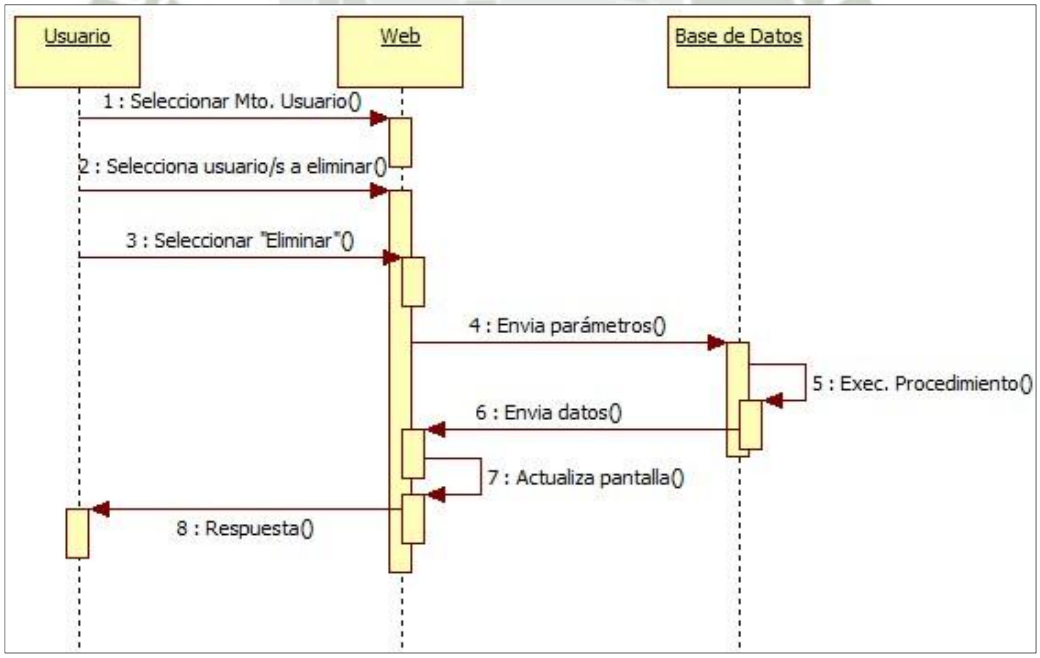

Figura 4.10. - Diagrama de Secuencia: Mantenimiento de Usuarios - Baja
Fuente: Elaboración propia.

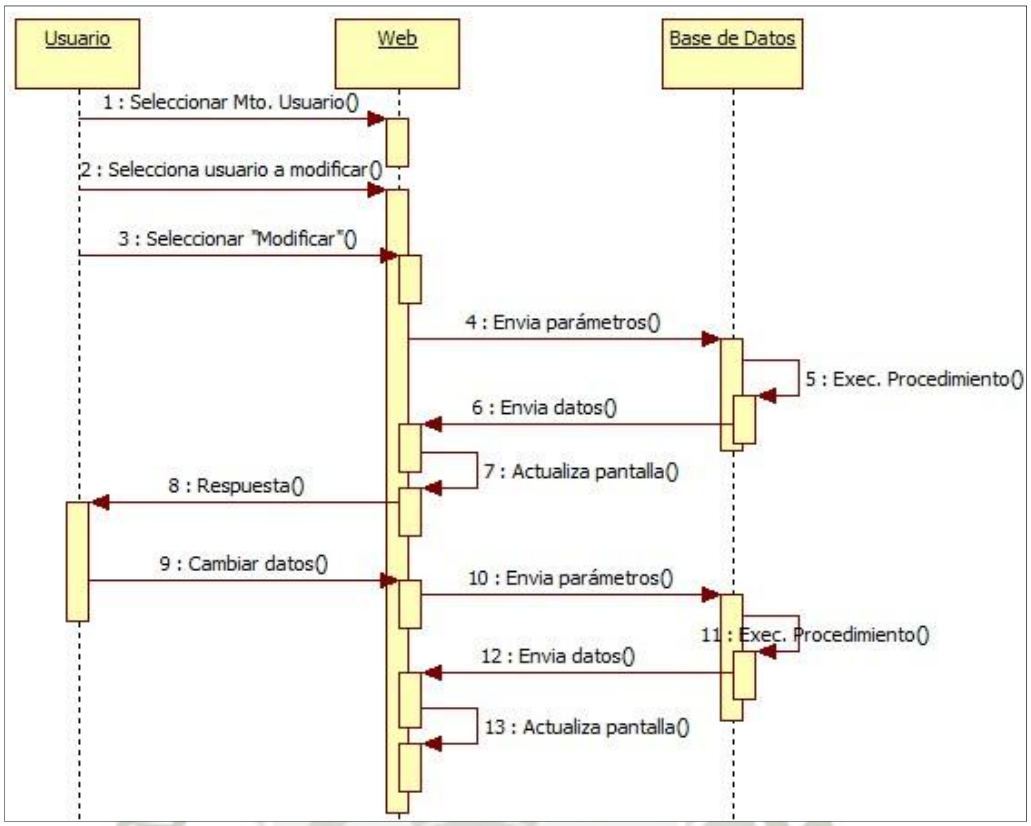

Figura 4.11. - Diagrama de Secuencia: Mantenimiento de Usuarios - Cambio
Fuente: Elaboración propia.

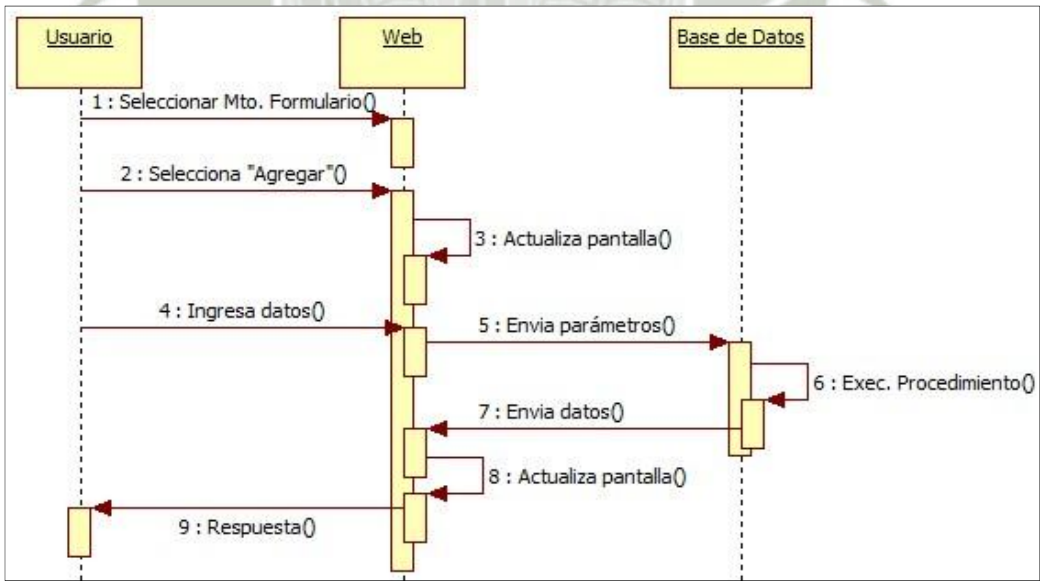

Figura 4.12. - Diagrama de Secuencia: Mantenimiento de Formularios - Alta
Fuente: Elaboración propia.

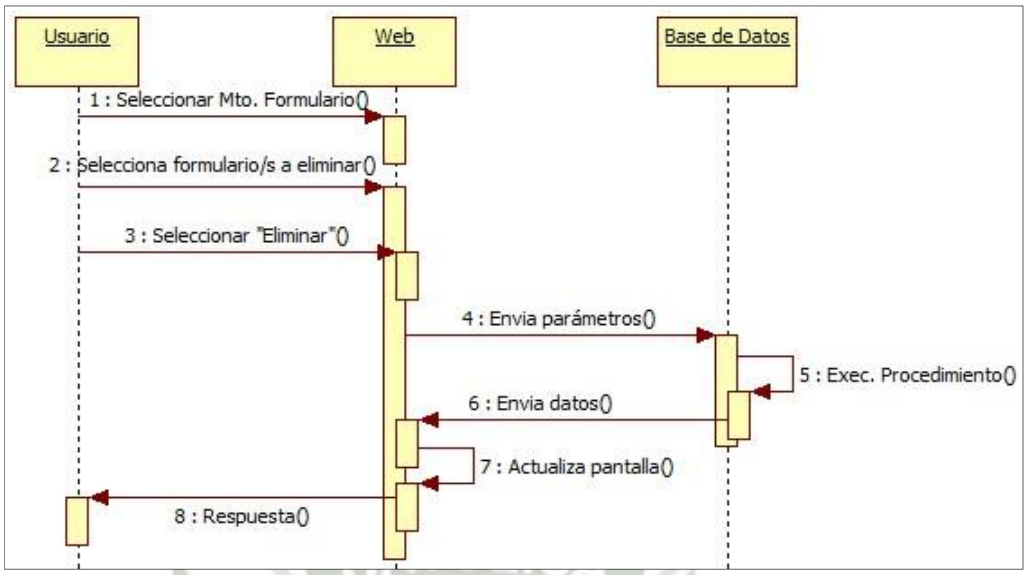

Figura 4.13. - Diagrama de Secuencia: Mantenimiento de Formularios - Baja
Fuente: Elaboración propia.

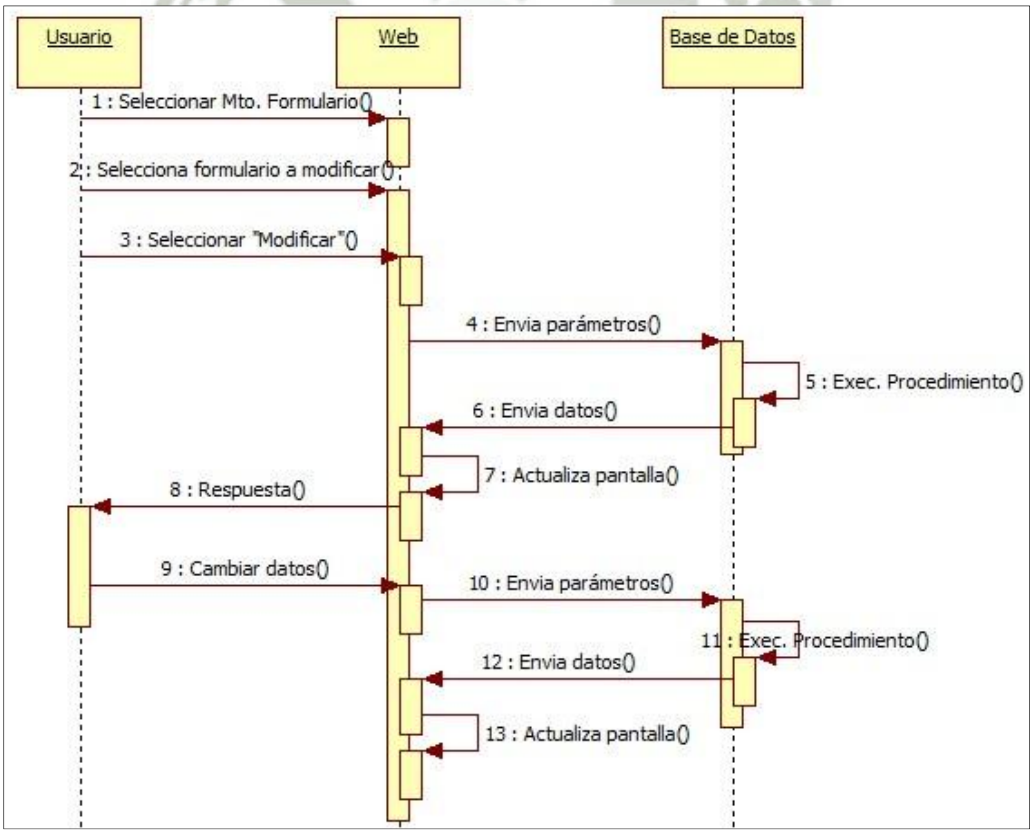

Figura 4.14. - Diagrama de Secuencia: Mantenimiento de Formularios - Cambio
Fuente: Elaboración propia.

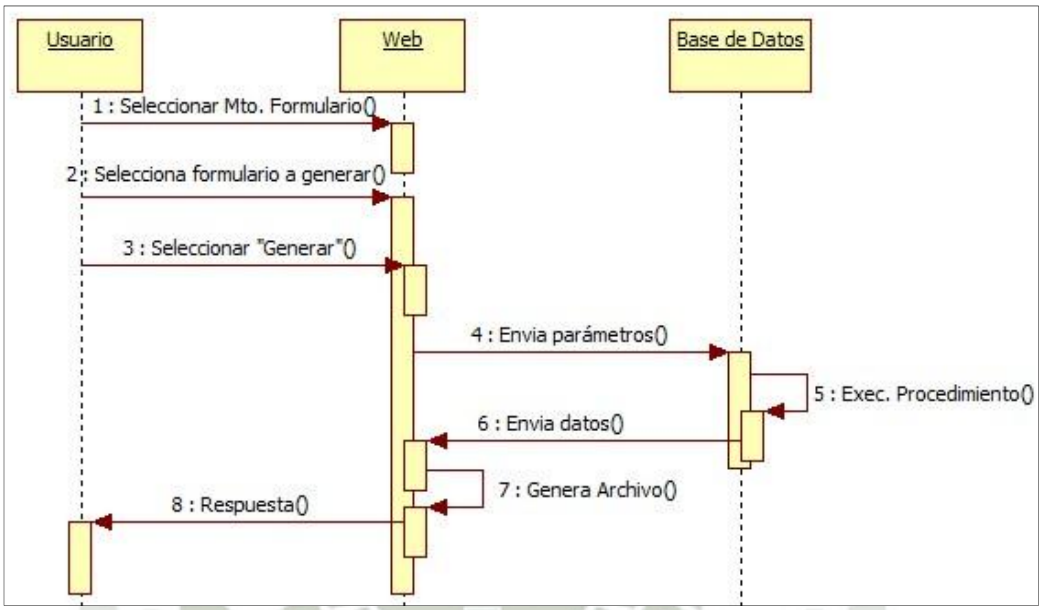

Figura 4.15. - Diagrama de Secuencia: Mantenimiento de Formularios - Generar
Fuente: Elaboración propia.

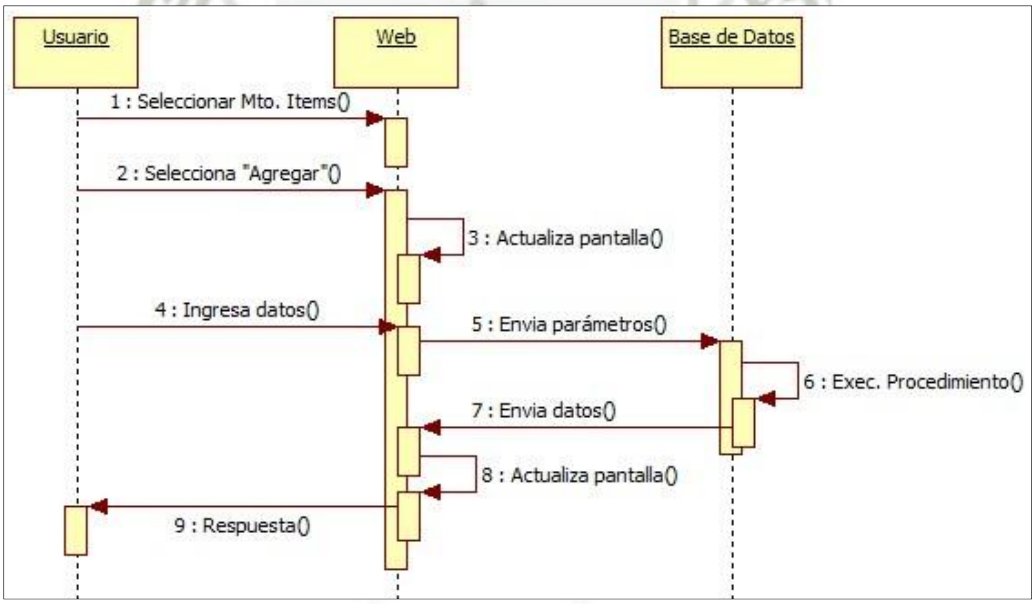

Figura 4.16. - Diagrama de Secuencia: Mantenimiento de Ítems - Alta
Fuente: Elaboración propia.

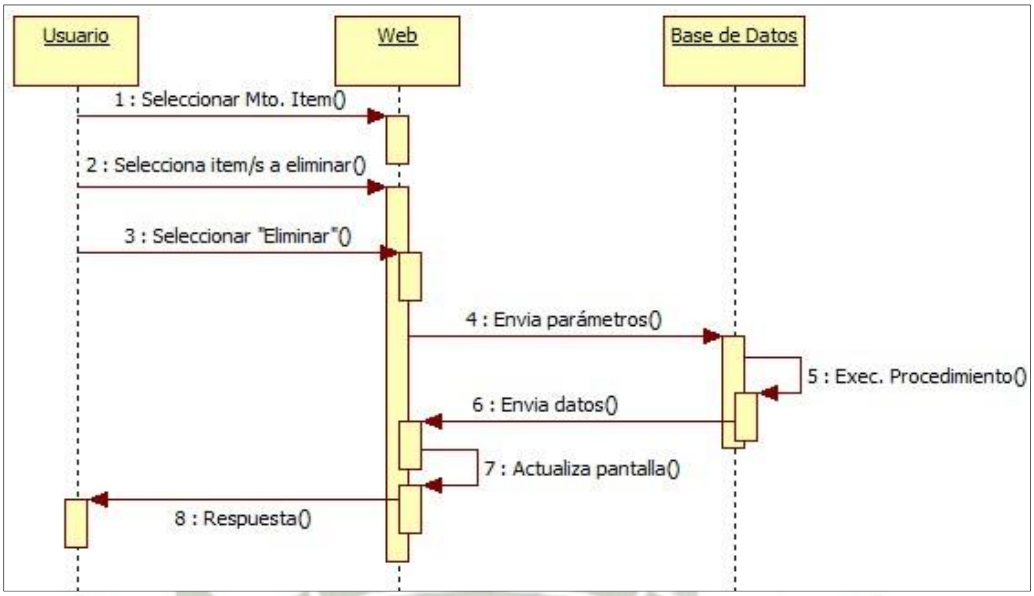

Figura 4.17. - Diagrama de Secuencia: Mantenimiento de Ítems - Baja
Fuente: Elaboración propia.

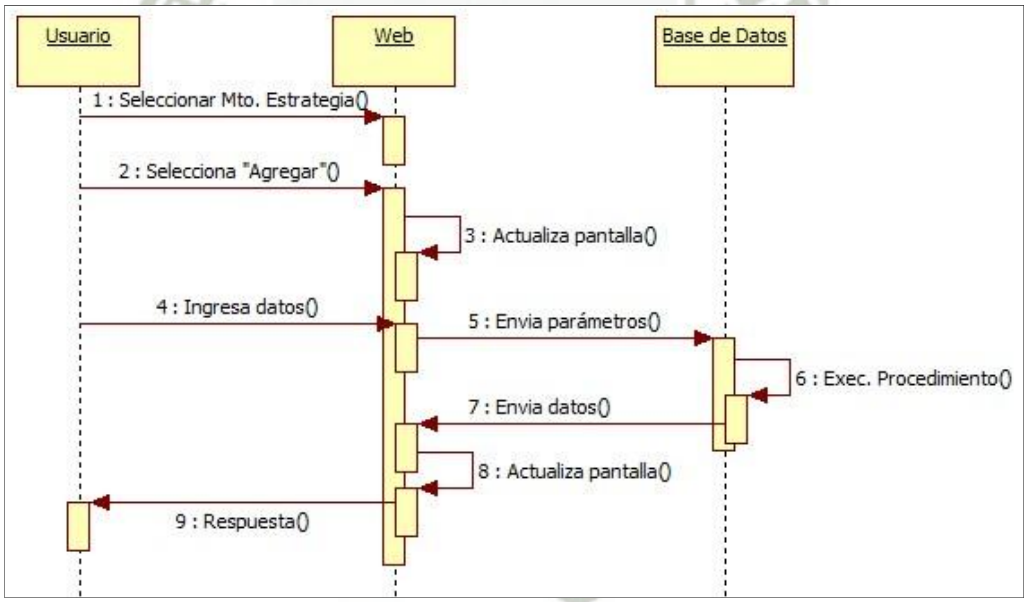

Figura 4.18. - Diagrama de Secuencia: Mantenimiento de Estrategias - Alta
Fuente: Elaboración propia.

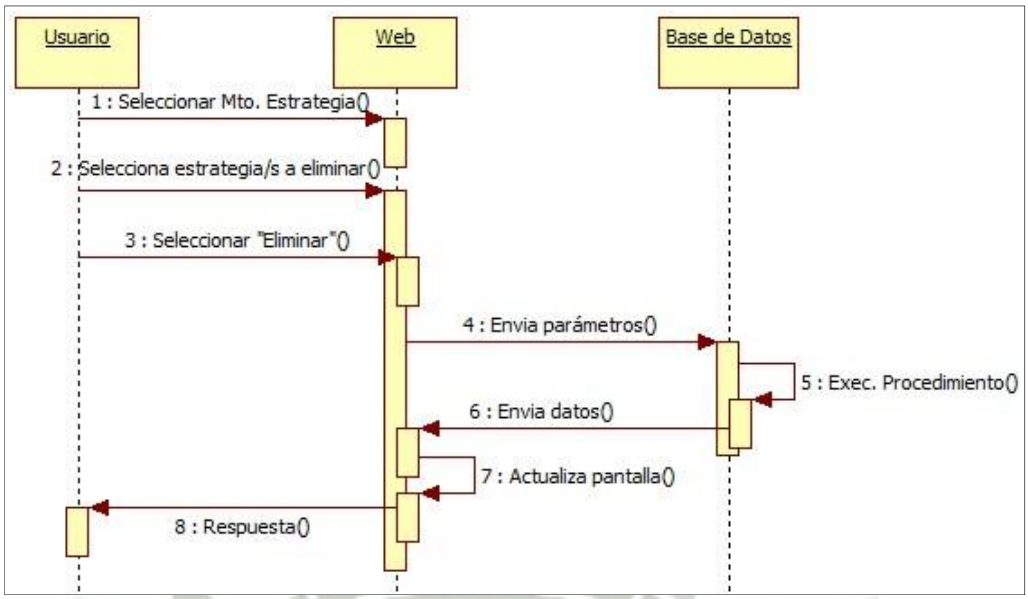

Figura 4.19. - Diagrama de Secuencia: Mantenimiento de Estrategias - Baja
Fuente: Elaboración propia.

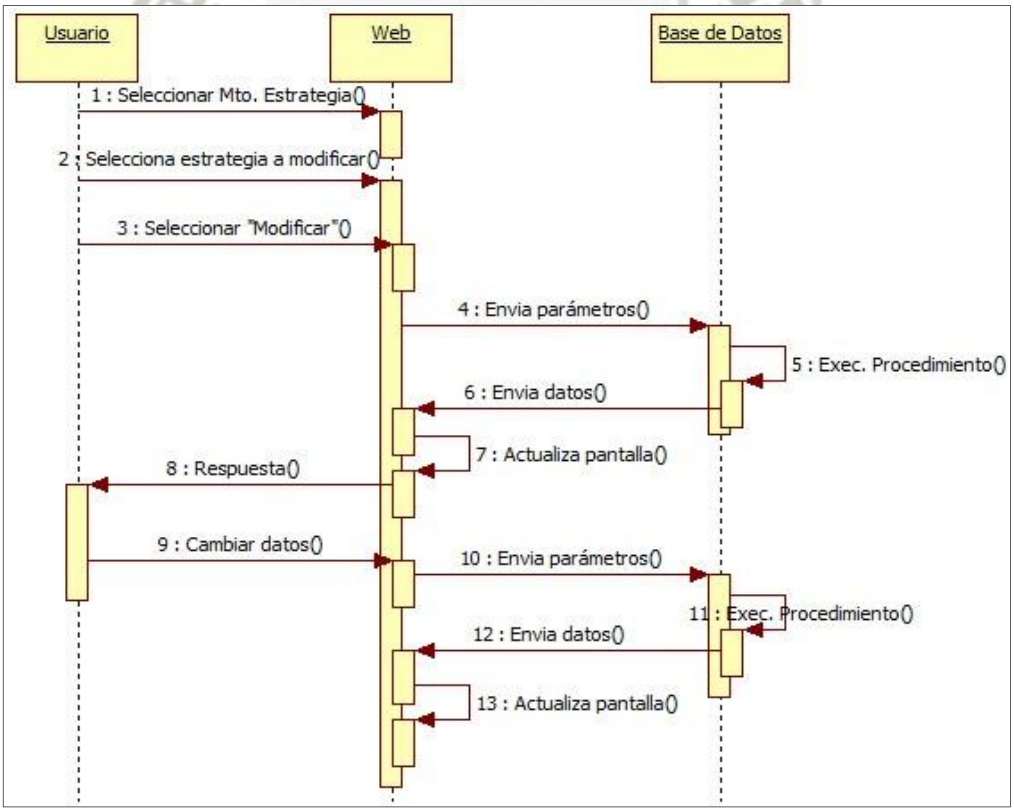

Figura 4.20. - Diagrama de Secuencia: Mantenimiento de Estrategias - Cambio
Fuente: Elaboración propia.

4.3.6.2. Diagramas de Secuencia - Móvil

- Diagrama de Secuencia - Sincronización

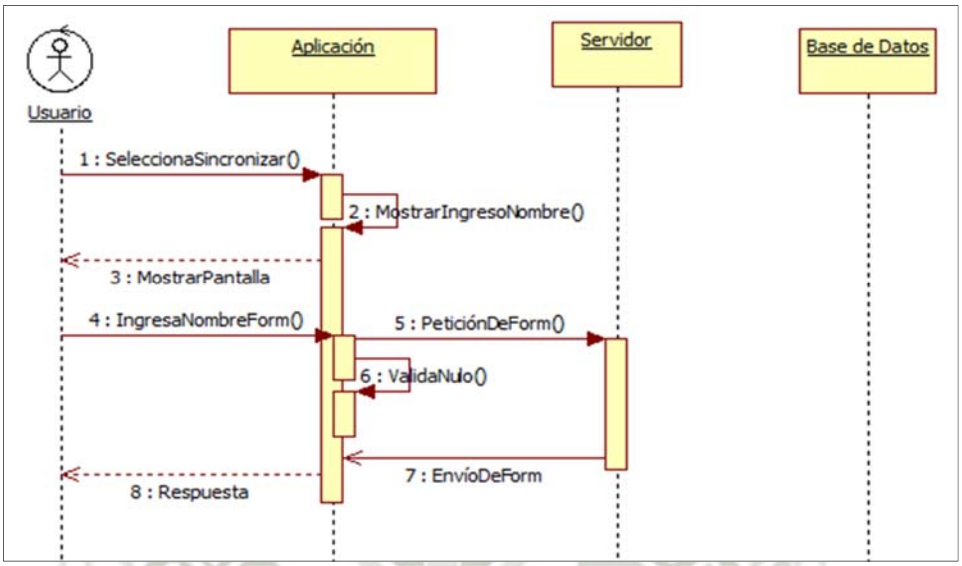

Figura 4.21. - Diagrama de Secuencia: Sincronización
Fuente: Elaboración propia.

- Diagrama de Secuencia - Visualización de formulario

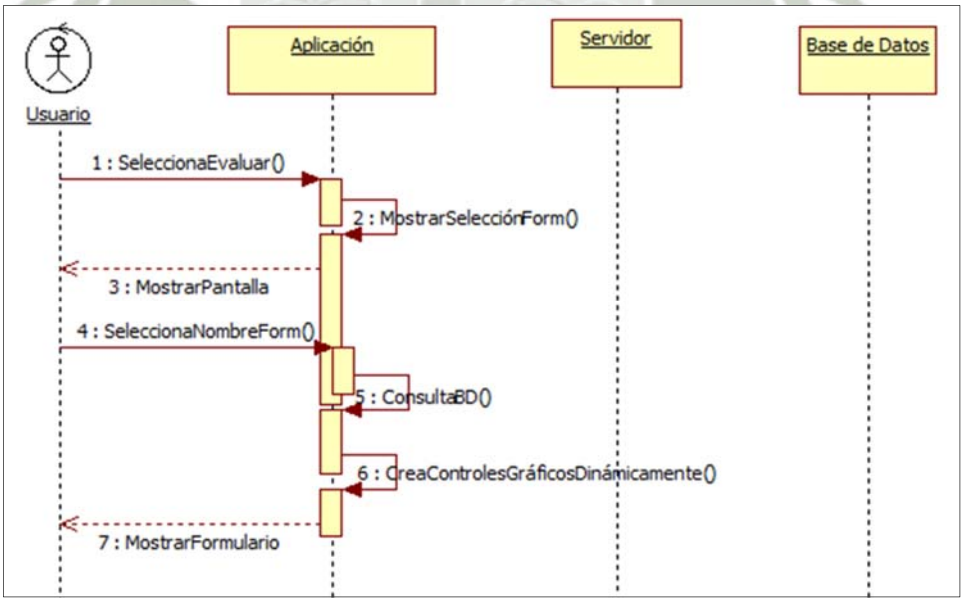

Figura 4.22. - Diagrama de Secuencia: Visualización de formulario
Fuente: Elaboración propia.

- Diagrama de Secuencia - Ingreso de datos y Evaluación

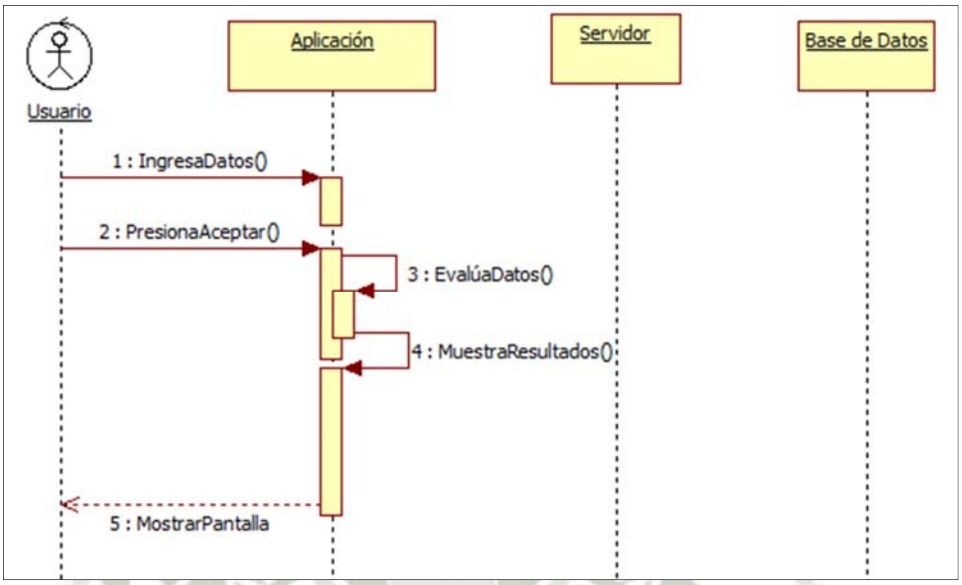

Figura 4.23. - Diagrama de Secuencia: Ingreso de datos y Evaluación
Fuente: Elaboración propia.

- Diagrama de Secuencia - Envío de resultados

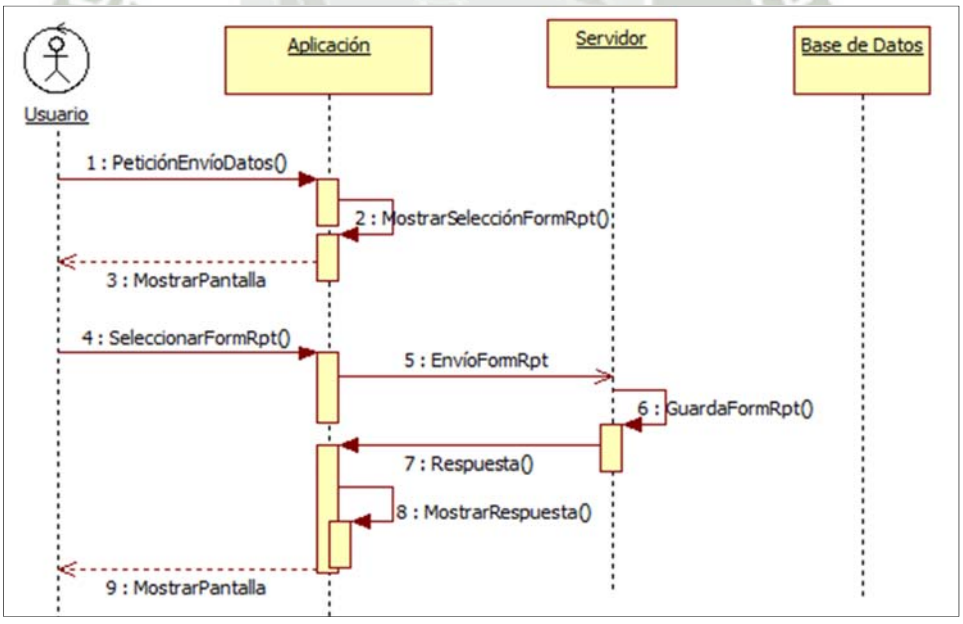

Figura 4.24. - Diagrama de Secuencia: Envío de resultados
Fuente: Elaboración propia.

4.4. IMPLEMENTACIÓN

Una vez realizado el análisis de los formularios y de la aplicación, se procede a la instanciación de la misma en base al Framework (Ver Anexo A).

CAPÍTULO 5

PRUEBAS Y RESULTADOS

5.1. EVALUACIÓN DE EXPERTOS

Nielsen [NIE94] sostiene que la evaluación basada en criterios debe ser realizada por evaluadores especializados en los temas que componen la investigación, al menos 3 deben ser los expertos, esto determina el costo mínimo de este método frente a otros.

Los estudios de Nielsen muestran que un número de entre 3 y 5 evaluadores es suficiente para la evaluación por criterios, así mismo permite la interacción con los expertos logrando así profundizar en cuestiones de interés que ayuden a mejorar el producto.

5.2. MARCO DE LA EVALUACIÓN DE EXPERTOS

Con la finalidad de presentar y evaluar el Framework propuesto se realizó la siguiente dinámica:

- Presentación del Framework.
- Pasos para instanciar el Framework.

- Explicación de uso del Framework.
- Definición de recursos.
- Presentación de la aplicación generada.

Al finalizar se realizó una encuesta con 14 preguntas a 4 personas expertas.

5.3. PERFIL DE EXPERTOS

- Estudios y/o experiencia en análisis, diseño e implementación de aplicaciones móviles y sistemas web.
- Experiencia en la utilización de Frameworks.
- Conocimientos en tecnologías web y móviles.

5.4. TÉCNICA E INSTRUMENTOS DE EVALUACIÓN

La encuesta realizada a los expertos (Ver ANEXO D) fue construida utilizando la técnica de evaluación formal: Observación sistemática, que sirve para evaluar y recoger información relevante sobre la conducta de los sujetos evaluados, además de características de objetos y hechos de manera grupal o personal sin discriminar el lugar en que se llevan a cabo.

Se utilizan los siguientes instrumentos de evaluación comprendidos dentro de la técnica de Observación Sistemática: [TEC06]

- **Escala de Apreciación:** Este instrumento permite apreciar o estimar la intensidad de la conducta a lo menos en tres categorías. Exactamente se utilizó escalas de apreciación de tipo Graficas.

  - ✓ **Las Escalas gráficas:** Se representan mediante una línea o casilleros con conceptos opuestos en sus extremos. Son apropiadas para representar aspectos afectivos y de sociabilidad como las actitudes, intereses y sentimientos. Considera los siguientes elementos:

    - Los tramos de la escala son impares.
    - El centro representa un punto neutro o indiferencia.
    - El lado izquierdo es negativo y el derecho positivo.

- **Lista de Cotejo:** Permite estimar la presencia o ausencia de una serie de características o atributos relevantes en las actividades o productos realizados por los evaluados. Se puede emplear tanto para la evaluación de capacidades como de actitudes. Tienen siempre dos posibilidades de respuesta, como por ejemplo: sí o no; logrado o no logrado; etc.

5.5. INTERPRETACIÓN DE RESULTADOS

Tabla 5.1. - Relación: Variables, Indicadores, Ítems de Encuesta.
Fuente: Elaboración propia.

Variable	Indicadores	Ítem de la Encuesta
Variable Independiente Creación dinámica de componentes gráficos.	Consistencia	Pregunta N° 4
Variable Dependiente Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android.	Facilidad de uso.	Pregunta N° 1 Pregunta N° 2 Pregunta N° 3 Pregunta N° 13
	Tiempos de implementación.	Pregunta N° 1 Pregunta N° 5 Pregunta N° 8 Pregunta N° 10
	Grado de reutilización de controles.	Pregunta N° 6 Pregunta N° 7

1. **¿El Framework propuesto facilita y agiliza la implementación de aplicaciones para adquisición de datos?**

Tabla 5.2. - Pregunta 1: Ayuda y agilización de la implementación.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	4	100%
No	0	0%
Total	4	100%

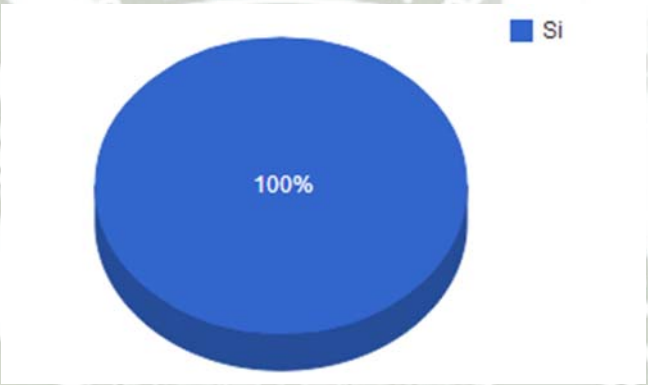

Figura 5.1. - Pregunta 1: Ayuda y agilización de la implementación.
Fuente: Elaboración propia.

Interpretación: Todos los usuarios afirman que el Framework propuesto agiliza la implementación de aplicaciones para toma de datos, resaltando la facilidad de uso y la reducción en tiempos de implementación debido a que funciona como una óptima base adaptable para el desarrollo de aplicaciones móviles de este tipo.

2. ¿Cómo calificaría la organización de los paquetes y archivos del Framework?

Tabla 5.3. - Pregunta 2: Organización de paquetes y archivos.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	3	75%
Regular	1	25%
Mala	0	0%
Total	4	100%

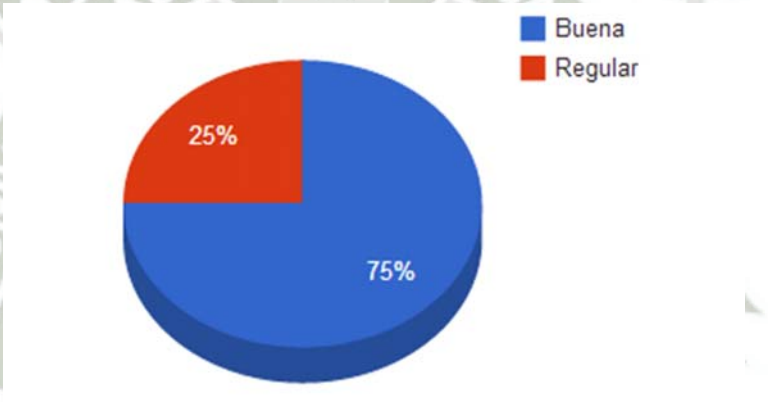

Figura 5.2. - Pregunta 2: Organización de paquetes y archivos.
Fuente: Elaboración propia.

Interpretación: De los 4 expertos, 3 de ellos sostienen que la organización de los archivos y paquetes ayuda mucho en el orden, comprensión y simplicidad de la propuesta, 1 persona recomienda incluir la carpeta de imágenes del servidor en la carpeta CSS con el fin de reducir los accesos. Resaltaron que la organización de los archivos se hizo de forma muy descriptiva, lo que facilita su comprensión y ubicación para futuras modificaciones.

3. ¿Considera que es difícil el manejo, entendimiento estructural y funcional del Framework?

Tabla 5.4. - Pregunta 3: Manejo, entendimiento estructural y funcional.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	0	0%
No	4	100%
Total	4	100%

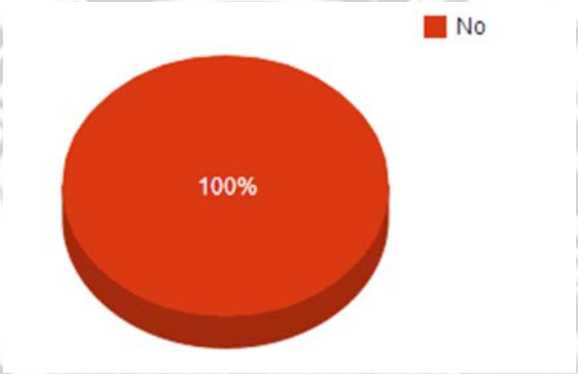

Figura 5.3. - Pregunta 3: Manejo, entendimiento estructural y funcional.
Fuente: Elaboración propia.

Interpretación: El 100% de los expertos coincidió en que no es difícil el manejo y el entendimiento estructural y funcional del Framework resaltando la claridad de la documentación y la intuitiva parametrización de controles y formularios que posee el presente proyecto.

4. ¿Considera que los controles vistos en la aplicación móvil demuestran correctamente las propiedades de los controles parametrizados desde el módulo web del Framework?

Tabla 5.5. - Pregunta 4: Consistencia de controles parametrizados.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	4	100%
No	0	0%
Total	4	100%

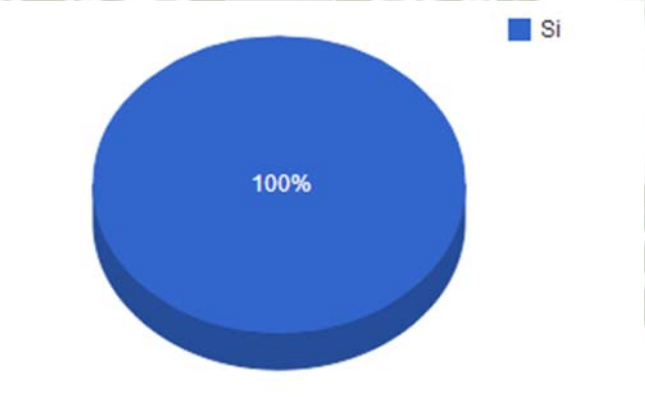

Figura 5.4. - Pregunta 4: Consistencia de controles parametrizados.
Fuente: Elaboración propia.

Interpretación: Todos los expertos coinciden en la consistencia de los controles parametrizados desde la interfaz web, tienen exactamente el comportamiento que se esperaba en el dispositivo móvil.

5. Considera que integrar un nuevo control para la toma de datos es:

Tabla 5.6. - Pregunta 5: Integración de nuevos controles.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Fácil	2	50%
Regular	2	50%
Difícil	0	0%
Total	4	100%

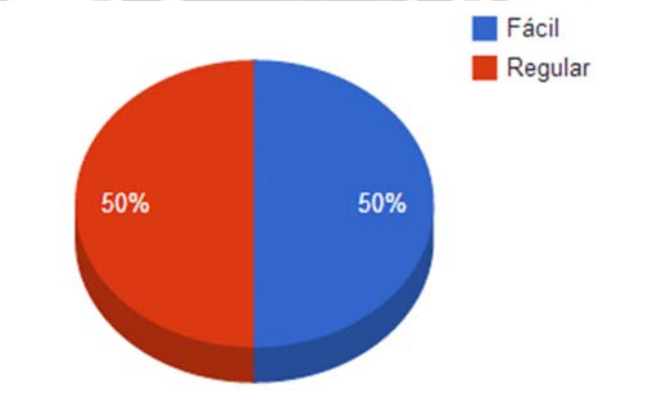

Figura 5.5. - Pregunta 5: Integración de nuevos controles.
Fuente: Elaboración propia.

Interpretación: Del total de expertos, 2 sostienen que integrar un nuevo control para toma de datos es un proceso fácil, por otra parte 2 expertos sostienen que este proceso puede tomar cierto nivel de dificultad al necesitar conocimientos en Android o en caso de ser necesaria la modificación de los archivos principales del Framework.

6. Considera que el grado de reutilización de los controles existentes en el Framework es:

Tabla 5.7. - Pregunta 6: Grado de reutilización.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	2	50%
Regular	2	50%
Mala	0	0%
Total	4	100%

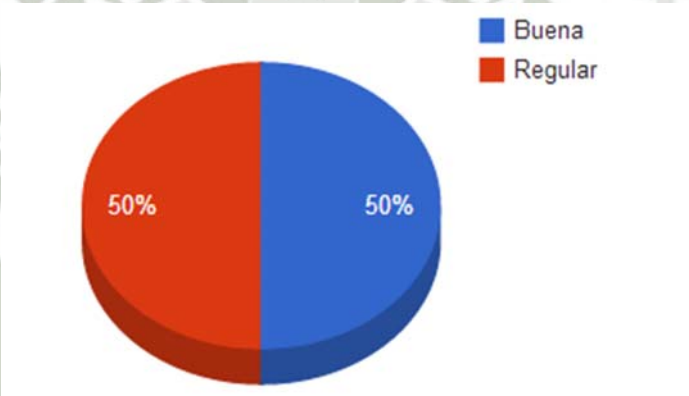

Figura 5.6. - Pregunta 6: Grado de reutilización.
Fuente: Elaboración propia.

Interpretación: De los 4 expertos encuestados, 2 manifiestan que el grado de reutilización de los controles es bueno porque pueden aplicarse en distintos propósitos, por otra parte 2 expertos califican la reutilización de controles como regular, ya que al implementar nuevos controles para tareas muy específicas, estos podrían perder su nivel de reutilización.

7. ¿Cómo calificaría la creación de nuevos controles en base al modelo de los controles existentes?

Tabla 5.8. - Pregunta 7: Creación de nuevos controles.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	2	50%
Regular	2	50%
Mala	0	0%
Total	4	100%

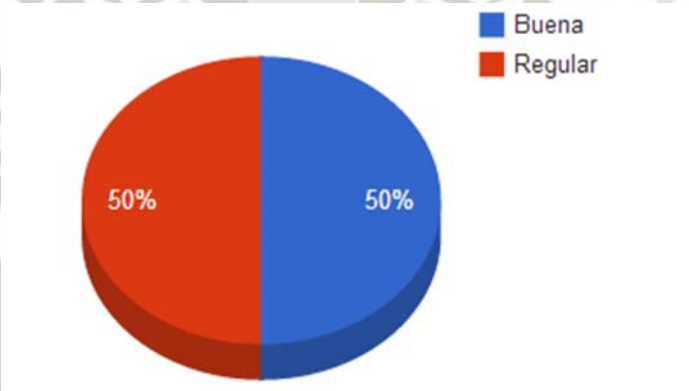

Figura 5.7.- Pregunta 7 - Creación de nuevos controles.
Fuente: Elaboración propia.

Interpretación: Del total de expertos encuestados, 2 expertos calificaron como buena la creación de nuevos controles en base a los ya existentes, por otra parte 2 expertos calificaron como regular la creación de nuevos controles argumentando que mientras menos congruencia tenga el nuevo control con alguno ya existente, aumentará la dificultad de implementación del nuevo control y que al crear controles con propósitos específicos los ya existentes sólo servirán como un modelo básico.

8. Sabiendo que la creación dinámica de un componente gráfico, sin utilizar algún Framework, conlleva un determinado tiempo de implementación, ¿Cree usted que el tiempo invertido en crear un nuevo control utilizando el Framework propuesto, es óptimo?

Tabla 5.9. - Pregunta 8: Tiempo de creación de un nuevo control.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	3	75%
No	1	25%
Total	4	100%

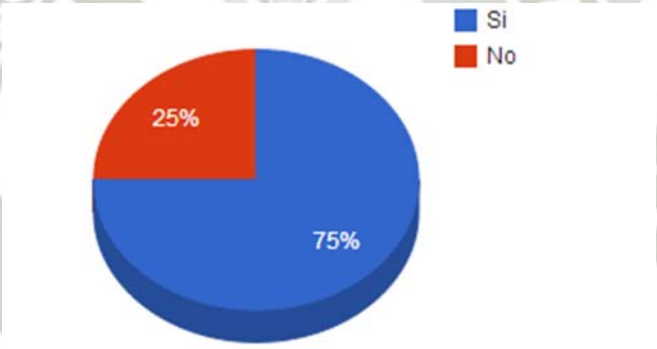

Figura 5.8. - Pregunta 8: Tiempo de creación de un nuevo control.
Fuente: Elaboración propia.

Interpretación: El 75% de los expertos coinciden en que el tiempo invertido en crear un nuevo control es óptimo debido a que gran parte del proceso está automatizado y que los archivos que se deben modificar están plenamente identificados. Mientras tanto 1 experto que representa el 25% del total de expertos piensa que el tiempo invertido dependerá de los conocimientos del desarrollador y que el proyecto debería necesitar la modificación de menor número de archivos.

9. ¿Cómo calificaría usted el proceso de generación (creación) dinámica de componentes gráficos al momento de ejecución?

Tabla 5.10. - Pregunta 9: Proceso de generación dinámica de componentes.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	3	75%
Regular	1	25%
Mala	0	0%
Total	4	100%

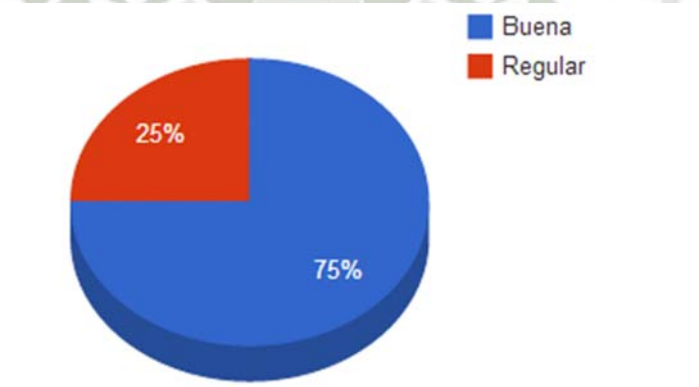

Figura 5.9. - Pregunta 9: Proceso de generación dinámica de componentes.
Fuente: Elaboración propia.

Interpretación: Para el 75% de los expertos, representado por 3 personas, el proceso de generación dinámica de componentes es bueno siendo un proceso óptimo y transparente para el usuario, 1 experto recomienda el uso de ayudas visuales durante la parametrización para ayudar al usuario a comprender el uso de los controles.

10. ¿Cree usted que el uso de Procedimientos Almacenados para las operaciones que realiza el Framework reducen el tiempo de mantenimiento y acceso a datos?

Tabla 5.11. - Pregunta 10: Uso de Procedimientos Almacenados.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	4	100%
No	0	0%
Total	4	100%

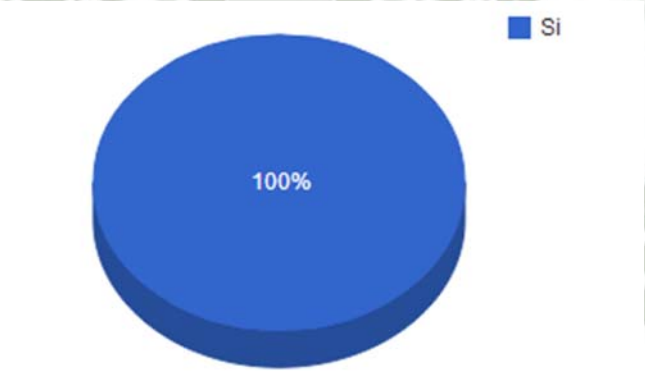

Figura 5.10. - Pregunta 10: Uso de Procedimientos Almacenados.
Fuente: Elaboración propia.

Interpretación: Los 4 expertos encuestados que representan el 100% consideran que el uso de procedimientos almacenados reduce el tiempo de mantenimiento y acceso a datos al encapsular toda la lógica en el manejador de base de datos, así mismo permite la reusabilidad de procedimientos y reduce la carga de red en caso de aumentar el número de formularios o parametrizaciones.

11. Le parece que la conexión y acceso a datos es:

Tabla 5.12. - Pregunta 11: Conexión y acceso a datos.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	2	50%
Regular	2	50%
Mala	0	0%
Total	4	100%

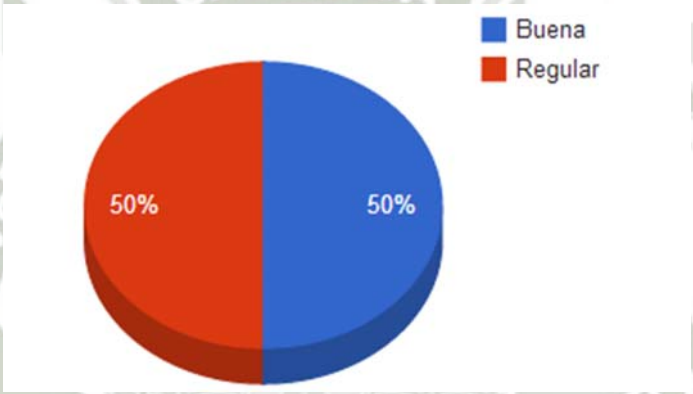

Figura 5.11. - Pregunta 11: Conexión y acceso a datos.
Fuente: Elaboración propia.

Interpretación: Del total de los expertos encuestados, 2 que representan el 50% califican como buena la conexión y acceso a datos ya que no requiere una conexión continua al servidor para su funcionamiento, consideran que guardar los datos localmente en los dispositivos dota de autonomía a la aplicación. Mientras tanto, 2 expertos que representan el 50% restante valoran como regular a la conexión y acceso a datos, argumentan que les parece excelente el uso del formato de intercambio JSON debido a que el intercambio de datos se hace más ligero, pero que sería bueno incluir mayor seguridad y utilizar encriptación.

12. ¿Cómo calificaría la aplicación instanciada en base al Framework?

Tabla 5.13. - Pregunta 12: Calificación de la aplicación instanciada.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Buena	3	75%
Regular	1	25%
Mala	0	0%
Total	4	100%

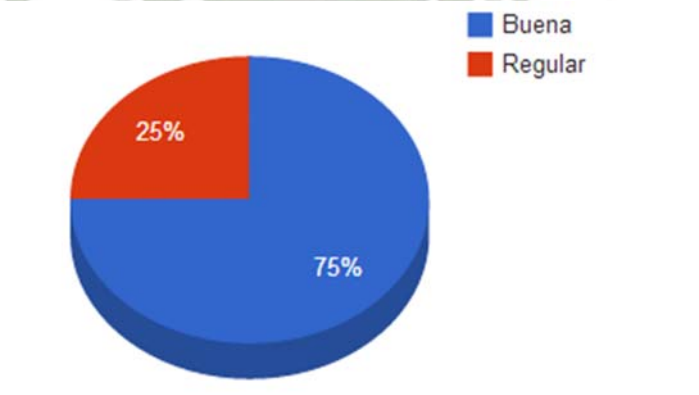

Figura 5.12. - Pregunta 12: Calificación de la aplicación instanciada.
Fuente: Elaboración propia.

Interpretación: 3 expertos califican la aplicación instanciada como buena ya que el funcionamiento es correcto y la interfaz agradable, además de acelerar el proceso de desarrollo, por otra parte, el experto que representa el 25% del total califica como regular la aplicación instanciada y opina que sería óptimo comprimir algunas clases dentro de un archivo .jar para que los desarrolladores no modifiquen las partes de código del Framework que son elementales para conservar su correcto funcionamiento.

13. Considera que dar mantenimiento a una aplicación instanciada por el Framework es:

Tabla 5.14. - Pregunta 13: Mantenimiento de la aplicación instanciada.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Fácil	3	75%
Regular	1	25%
Difícil	0	0%
Total	4	100%

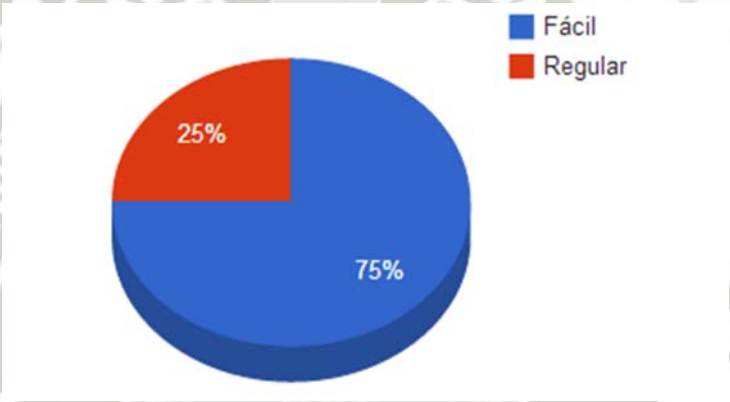

Figura 5.13. - Pregunta 13: Mantenimiento de la aplicación instanciada.
Fuente: Elaboración propia.

Interpretación: 3 expertos que conforman el 75% consideran que dar mantenimiento a la aplicación instanciada es fácil ya que la organización de los paquetes y carpetas ayudan al programador, por otra parte 1 persona manifiesta que la complejidad del mantenimiento dependerá del crecimiento de la aplicación y de la inclusión de lógica de negocio para la satisfacción de requerimientos.

14. ¿Considera que el Framework propuesto podría modificarse fácilmente con el fin de proveer nuevas funcionalidades?

Tabla 5.15. - Pregunta 14: Facilidad para integrar nuevas funcionalidades.
Fuente: Elaboración propia.

Respuesta	Periodicidad	Porcentaje
Si	4	100%
No	0	0%
Total	4	100%

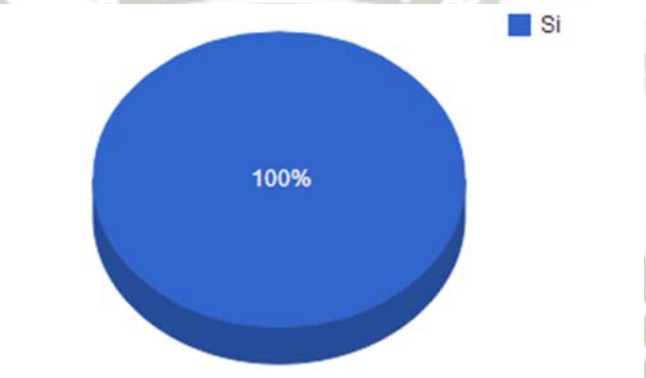

Figura 5.14. - Pregunta 14: Facilidad para integrar nuevas funcionalidades.
Fuente: Elaboración propia.

Interpretación: El total de expertos encuestados manifiesta que si se puede integrar nuevas funcionalidades al Framework propuesto ya que la organización, estructura y lenguajes utilizados lo permite.

CONCLUSIONES

1. A través de la presente propuesta, se logró construir un Framework orientado a la implementación de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android utilizando creación dinámica de componentes gráficos.

2. Al desarrollar el Framework se logró elaborar las clases y archivos necesarios para el desarrollo de aplicaciones para la adquisición de datos en dispositivos móviles sobre la plataforma Android, obteniendo así una estructura organizada y óptima con la finalidad de que el desarrollador pueda contar con las herramientas necesarias para la construcción de cualquier aplicación en base al presente proyecto.

3. Se logró definir una arquitectura adecuada para la implementación del Framework propuesto.

4. Se establecieron los lineamientos generales para la utilización y aplicación práctica del Framework, generando así un manual de instanciación e implementación de cualquier aplicación construida en base a éste.

5. Se consiguió construir un prototipo que demuestra la utilidad y efectividad del uso del Framework propuesto, logrando ponerlo a prueba bajo la supervisión de personas calificadas.

RECOMENDACIONES

1. Incluir rutinas de encriptación para el envío y recepción de los datos para proteger la información posiblemente sensible enviada a través de internet.

2. Incluir interacción entre ítems de un mismo formulario, ampliando su funcionalidad, y generando lógica de negocio entre ellos.

3. Comprimir algunas clases dentro de un archivo .jar para que los desarrolladores no modifiquen las partes de código del Framework que son elementales para conservar su correcto funcionamiento, esto también beneficiaría al desarrollador al contar con la modificación de un código fuente más limpio.

4. Ampliar la comunicación con otras bases de datos como Mysql, PostgreSQL y Oracle definiendo clases y aumentando características de polimorfismo al Framework.

5. Tomar como base la presente investigación para implementar nuevas funcionalidades, ampliando así la utilidad del Framework expuesto.

BIBLIOGRAFÍA

Libros, Artículos y Revistas

- [ECW13] CAVANAUGH, ERIN. Web services: Benefits, challenges, and a unique, visual development solution. Altova Inc. Págs. 04 - 06

- [GAR11] GARGENTA, MARKO. Learning Android. Editorial O'reilly Media, Febrero, 2011. Págs. 07 - 35.

- [GUL03] GULUTZAN, PETER Y PELZER, TRUDY. SQL Performance Tuning. Editorial Pearson Education Inc. Año 2003. Págs. 285 - 293.

- [INT06] MARRONERO EXPÓSITO, CARLOS. Interfaz Gráfica de Usuario, Aproximación semiótica y cognitiva, Universidad de la Laguna, Tenerife. Año 2006. Págs. 07 - 08.

- [NIE94] NIELSEN, JAKOB. Heuristic evaluation. In: Nielsen, Jakob and Mack, Robert L. (eds.). "Usability Inspection Methods". New York, 1994.

- [PEI10] PEI ZHENG, LIONEL NI. Smart Phone and Next Generation Mobile Computing. Editorial: Morgan Kaufmann. Julio, 2010. Págs. 49 - 51.

- [SOM05] SOMMERVILLE, IAN. Ingeniería del Software. Editorial Pearson Education. Madrid, 2005. Págs. 226 - 229.

Tesis

- [DEG11] DEGAYON CORTES, MIGUEL, "Testing" Aplicación de cuestionarios para Android. Universidad Carlos III de Madrid. Departamento de Informática. Fecha: Junio, 2011.

- [JAI09] JAIME ARANAZ TUDELA, Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma Android de Google.
  Fecha: Madrid, 2009

- [TOR12] TORRES SANZ, VICENTE.Tesis "SimDetect: aplicación de seguridad para la localización de dispositivos móviles Android". Universidad Zaragoza. Teruel, diciembre del 2012. Pág. 38.
  Consultado desde: http://docencia-eupt.unizar.es/paco/tfc/TFC_Torres.pdf
  Fecha: Abril, 2013.

Proyectos

- [MAR13] MARITACA TEAM. Proyecto Maritaca. Instituto de Ciencia y Tecnología de la Universidad Federal de Sao Paulo.
  Consultado desde: http://maritaca.unifesp.br:8080/maritaca/index.html. Brasil, Sao Paulo 2013.

- [MIT12] MIT (Massachusetts Institute of Technology). Proyecto APP INVENTOR. Consultado desde: http://appinventor.mit.edu/. Noviembre, 2012.

Referencias Web

- [DAP13] APPLE DEVELOPER. Apple Inc.
  Consultado desde: https://developer.apple.com/programs/ios/.
  Última consulta: 31 de agosto de 2013.

- [DAN13] ANDROID DEVELOPER. Google Inc.
  Consultado desde: http://developer.android.com/.
  Última consulta: 31 de agosto de 2013.

- [DEV13] DEVELOPERS ANDROID. Página Web que provee información que Google ofrece sobre Android.
  Disponible en: http://developer.android.com.
  Última consulta: 16 de julio de 2013

- [DNO13] NOKIA DEVELOPER. Nokia Corporation.
  Consultado desde: http://developer.nokia.com.
  Última consulta: 31 de agosto de 2013.

- [DUT07] DUTCHGUILDER. Iterative Development Illustration. Wikimedia Commons. Octubre, 2007.
  Disponible en: http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process.
  Última consulta: 31 de agosto de 2013.

- [FOR12] FORMIIK.
  Consultado desde: http://www.formiik.com/.
  Última consulta: 20 de enero de 2013.

- [GOO06] GOOGLE INC. Google Labs Spreadsheets.
  Enlace: http://en.wikipedia.org/wiki/Google_Docs. Junio, 2006.
  Última consulta: 31 de agosto de 2013.

- [GOO13] GOOGLE INC. Google Play.
  Consultado desde: https://play.google.com/store.
  Última consulta: 20 de enero de 2013.

- [GUT05] GUTIERREZ, JAVIER. ¿QUÉ ES UN FRAMEWORK WEB?. Consultado desde: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.
  Última consulta: 16 de septiembre de 2013.

- [MSI10] MSI Services, Inc. IBM - Developer works.
  Enlace: http://www.ibm.com/developerworks/xml/tutorials/x-andddyntut/x-andddyntut-pdf.pdf. Septiembre 2010.
  Última consulta: 31 de agosto de 2013.

- [OPH13] OPEN HANDSET ALLIANCE.
  Consultado desde: http://www.openhandsetalliance.com.
  Última consulta: 2 de enero de 2013

- [POL11] UNIVERSIDAD POLITÉCNICA DE VALENCIA. Curso: Android, programación de aplicaciones para móviles.
  Consultada en: http://www.androidcurso.com.
  Última consulta: 19 de abril de 2013.

- [ROB13] ROBOQUIZ.
  Consultado desde: http://roboquiz.sourceforge.net/.
  Última consulta: 20 de enero de 2013.

- [SCH11] SCHOLARIUM SAS. Aplicaciones para dispositivos móviles. Bogotá - Colombia. 2011.
  Consultado desde: http://www.scholarium.co/index.php?option=com_content&view=article&id=65&Itemid=97
  Última consulta: 07 de abril de 2013.

- [STA13] STATCOUNTER. Web analytics Service: Sistemas Operativos Móviles utilizados a nivel mundial y nivel Perú.
  Disponible en: http://gs.statcounter.com/.
  Última consulta: 31 de agosto de 2013.

- [TEC06] MINISTERIO DE EDUCACIÓN - PERÚ. Técnicas e instrumentos de evaluación. Año 2006.
  Enlace:
  http://190.254.1.202/ingenieria/DIPLOMADO%20DOCENCIA%20UNIVERSITARIA/Educacion%20Superior/Eval_Competencia2.pdf.
  Última consulta: 08 de octubre de 2013.

- [TIB13] TIOBE Software BV Company. TIOBE Programming Community Index from September 2013.
  Enlace: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.
  Última consulta: 08 de octubre de 2013.

ANEXOS

ANEXO A

MANUAL DE USUARIO DEL FRAMEWORK

1. INTRODUCCIÓN

El presente Framework está integrado por herramientas ampliamente conocidas que son necesarias al momento de construir una aplicación de Toma de Datos utilizando la creación dinámica de componentes gráficos. Este manual supone que el desarrollador posee conocimientos de los lenguajes de programación C#, javascript y Java, aparte de conocimientos sobre programación orientada a objetos (POO). Este manual es un instrumento de ayuda para la implementación de una aplicación con ayuda del Framework.

2. CARPETAS Y ARCHIVOS DEL FRAMEWORK

Mostraremos cómo están distribuidos los archivos, carpetas y paquetes de los módulos web y móvil, todas estas pueden ser modificadas por el desarrollador y adaptadas a sus necesidades.

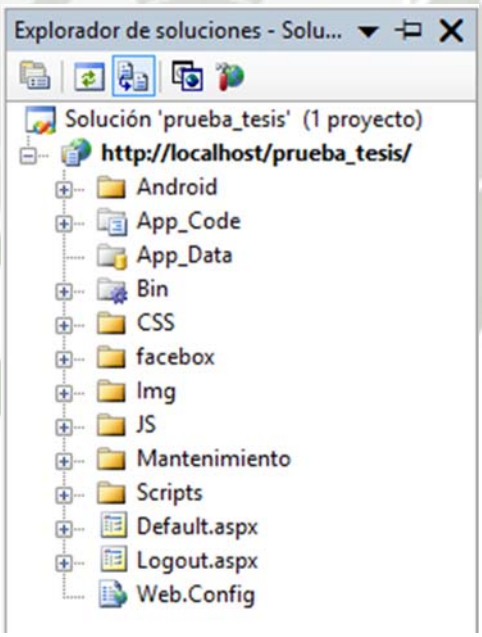

Figura Anexo A.1 - Archivos y carpetas módulo web.
Fuente: Elaboración propia

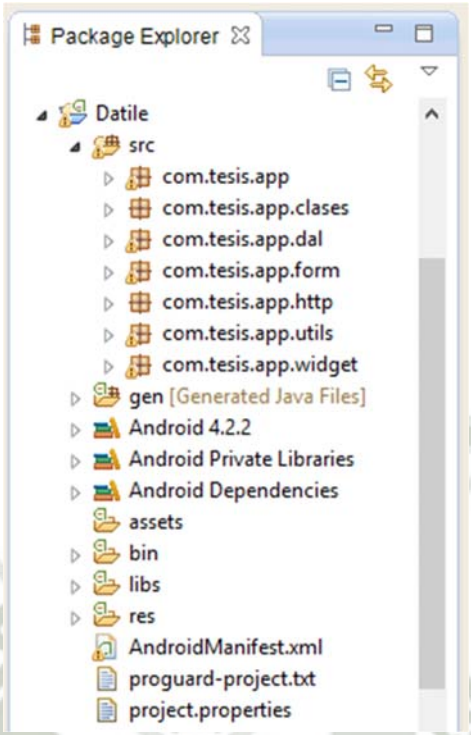

Figura Anexo A.2 - Archivos y carpetas módulo móvil.
Fuente: Elaboración propia

Para comprender mejor la composición total del Framework describiremos el contenido de las carpetas y paquetes mostrados anteriormente. Es así que podemos representar el contenido por medio de dos tablas, la primera describiendo el módulo web y la segunda los archivos, carpetas y paquetes que contiene el módulo móvil.

Tabla Anexo A.1 - Descripción de contenido módulo web.
Fuente: Elaboración propia

Nombre	Nombre Completo	Descripción
http://localhost/prueba_tesis	Módulo web	Contiene todo el proyecto web.
Android	Android	Contiene clases que proporcionan conexiones HTTP con los dispositivos móviles.
App_Code	App_Code	Contiene clases y objetos de negocio compartidos por toda la aplicación.
App_Data	App_Data	Contiene archivos de base de datos *.mdf, *.xml y otros archivos de almacén de datos, se utiliza para almacenar bases de datos locales de la aplicación.
Bin	Bin	Contiene librerías (*.dll), para los controles, componentes, o cualquier otro código que añada

		funcionalidad a la aplicación.
CSS	Hojas de estilos	Contiene todas las hojas de estilos.
facebox	Librería para ventanas modales.	Contiene archivos necesarios para el funcionamiento de las ventanas modales.
Img	Imágenes	Contiene todas las imágenes del módulo web.
JS	Javascript	Contiene archivos javascript.
Mantenimiento	Mantenimiento	Contiene las páginas encargadas de los mantenimientos de todos los objetos.
Scripts	Scripts	Contiene archivos *.txt que contienen las sentencias que serán ejecutadas en los dispositivos móviles.
Default.aspx	Default	Página de Login.
Logout.aspx	Logout	Página de Logout
web.config	web.config	Archivo XML de configuración de parámetros, permite configurar acceso a base de datos, inclusión de librerías, etc.

Tabla Anexo A.2 - Descripción de contenido módulo móvil.
Fuente: Elaboración propia

Nombre	**Nombre Completo**	**Descripción**
Datile	Proyecto Datile	Es la carpeta principal, contenedora de todo el proyecto.
src	Source	Esta carpeta contiene todo el código fuente de la aplicación, el código de la interfaz gráfica, clases auxiliares, etc. Todos los archivos que se crean aquí siguen la estructura de paquetes java definidos: - com.tesis.app - com.tesis.app.clases - com.tesis.app.dal - com.tesis.app.form - com.tesis.app.http - com.tesis.app.utils - com.tesis.app.widget
gen	Generated Java Files	Esta carpeta contiene elementos de código necesarios para el control de los recursos de la aplicación que

		son generados automáticamente al compilar el proyecto. Es importante remarcar que no se deben modificar manualmente de ninguna manera.
Android Private Libraries	Android Private Libraries	Son las librerías referenciadas existentes en la carpeta "libs".
assets	assets	Contiene los demás ficheros auxiliares para la aplicación, como por ejemplo ficheros de configuración, de datos, etc. El Framework no utiliza ningún archivo de esta carpeta.
bin	bin	Esta carpeta contiene los elementos compilados de la aplicación y otros ficheros auxiliares. Cabe destacar el fichero con extensión ".apk", que es el ejecutable de la aplicación que se instalará en el dispositivo. No se debe modificar manualmente ningún archivo de esta carpeta
libs	Libraries	Contiene las librerías auxiliares, normalmente en formato ".jar" que utilizamos en la aplicación Android. Actualmente contiene: • /jackson-all-1.8.1.jar • /android-support-v4.jar (El nivel de la librería de soporte determina la mínima versión API que el proyecto soporta)
res	Resource	Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.
AndroidManifest	Android Manifest	Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, ícono, etc.), sus componentes (pantallas, mensajes, etc.), las librerías auxiliares utilizadas, o los permisos necesarios para su ejecución.

3. DESCRIPCIÓN DE LOS PRINCIPALES ARCHIVOS

Entre los archivos que se ejecutan tanto en el lado del servidor como del cliente tenemos:

3.1. Modulo web

- **Carpeta Android**
  - Login.ashx: Se encarga de la autenticación de usuarios móviles.

Tabla Anexo A.3 - Descripción de Funciones de la página Login.ashx
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
ProcessRequest	HttpContext context	-	Permite el procesamiento de solicitudes Web HTTP.

  - ResGui.ashx: Se encarga de la recepción del archivo de respuesta.

Tabla Anexo A.4 - Descripción de Funciones de la página ResGui.ashx
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
ProcessRequest	HttpContext context	-	Permite el procesamiento de solicitudes Web HTTP.

- **Carpeta App_Code/Android**
  - CUsuario.cs: Clase usuario para dispositivos móviles.
  - ControlAndroid.cs: Agrupa métodos que permitan la interacción de las solicitudes HTTP y el acceso a datos.

Tabla Anexo A.5 - Descripción de Funciones de la clase ControlAndroid
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
autenticarUsuario	CUsuario usuario	-	Recibe el usuario ingresado, valida el ingreso.

  - DbAndroid.cs: Agrupa métodos para acceso a datos.

Tabla Anexo A.6 - Descripción de Funciones de la clase DbAndroid.
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
autenticarUsuario	CUsuario usuario	-	Recibe el usuario ingresado, se comunica con la Base de Datos, recibe la respuesta del procedimiento almacenado de autenticación.

- **Carpeta App_Code/Net**
  - CUsuario: Clase usuario para módulo web.
  - CFormato: Clase formulario para módulo web.
  - CItem: Clase item para módulo web.
  - CEstrategia: Clase estrategia para módulo web.
  - DBNet: Agrupa métodos para acceso a datos.

Tabla Anexo A.7 - Descripción de Funciones de la clase DbNet.
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
validarUsuarioWeb	string parUser, string pasPass	-	Recibe el usuario y contraseña para permitir o denegar el acceso.
getUsuarios	string usr_codigo, string usr_nombre, string usr_perfil, string usr_stat	-	Lista usuarios de acuerdo a los parámetros enviados.
getUsuariobyId	int idUser	-	Recupera un usuario identificado por el parámetro IdUser.
guardarUsuario	CUsuario usuario	-	Inserta o actualiza un usuario.

deleteUsuarios	string ids	-	Inhabilita uno o más usuarios.
getPerfiles		-	Recupera los perfiles.
getFormatos	string frm_idt, string frm_name, string frm_url, string frm_stat	-	Lista formularios de acuerdo a los parámetros enviados.
guardarFormato	CFormato formato	-	Inserto o actualiza un formulario.
getFormatobyId	int idFormato	-	Recupera un formulario identificado por el parámetro idFormato.
deleteFormatos	string ids	-	Inhabilita uno o más formularios.
getItems	String itm_name, string itm_label, string itm_form, string itm_ctrl, string itm_tsol, string itm_req, string itm_solve, string itm_stat	-	Lista items de acuerdo a los parámetros enviados.
getItembyId	int idItem	-	Recupera un item identificado por el parámetro idItem.
guardarItem	CItem item	-	Inserta o actualiza un item.
deleteItems	string ids	-	Inhabilita uno o más items.
getEstrategias	string str_form, string str_name, string str_label, string str_tsol, string str_stat	-	Lista estrategias de acuerdo a los parámetros enviados.

getEstrategiabyId	int idEstrategia	CEstrategia	Recupera un item identificado por el parámetro idEstrategia.
guardarEstrategia	CEstrategia estrategia	-	Inserta o actualiza una estrategia.
deleteEstrategias	string ids	-	Inhabilita una o más estrategias.
generarScript	String idForm, String idtForm	StringBuilder	Genera un script con los datos del formulario identificado por idForm y idtForm

- **Carpeta Bin.**
  - Newtonsoft.Json.dll: Librería Open Source que permite la serialización y deserialización de objetos a formato Json y viceversa.
- **Carpeta CSS:** Contiene hojas de estilo.
  - CSSIndex.css: Hoja de estilos para la página de "Login" del módulo web del framework.
  - CSSMaster.css: Hoja de estilos para el resto de páginas del módulo web del framework.
  - CSSMenu.css: Hoja de estilos para el menú del módulo web framework.
  - CSSVModal.css: Hoja de estilos para las ventanas modales (pop ups) del módulo web del framework.
- **Carpeta JS:** Contiene archivos javascript.
  - jquery-1.10.1.js: Librería Javascript que simplifica el manejo de HTML.
  - menu.js: Librería que se encarga de los efectos del menú.
- **Carpeta Mantenimiento:** Contiene páginas para la visualización, edición, eliminación e inserción de usuarios, formularios, items y estrategias.
  - UsuarioNuevo.aspx: Permite la creación de nuevos usuarios.
  - Usuario.aspx: Página para mantenimiento de usuario.

Tabla Anexo A.8 - Descripción de Funciones de la página Usuario.aspx.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
cargaPerfiles	-	-	Recupera todos los perfiles y llena el combo box de perfiles con el resultado.
cargaGrilla	-	-	Ejecuta getBusqueda y llena la grilla con los resultados.
getBusqueda	-	-	Lista todos los usuarios de acuerdo a parámetros enviados.
SubEliminar	-	-	Inhabilita los usuarios seleccionados.

- FormatoNuevo.aspx: Permite la creación de nuevos formularios.
- Formato.aspx: Página para mantenimiento de formularios.

Tabla Anexo A.9 - Descripción de Funciones de la página Formato.aspx.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
cargaGrilla	-	-	Ejecuta getBusqueda y llena la grilla con los resultados.
getBusqueda	-	-	Lista todos los formularios de acuerdo a parámetros enviados.
SubEliminar	-	-	Inhabilita los formularios seleccionados.

- ItemNuevo.aspx: Permite la creación de nuevos ítems.
- Item.aspx: Página para mantenimiento de ítems.

Tabla Anexo A.10 - Descripción de Funciones de la página Item.aspx.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
cargaFormularios	-	-	Recupera todos los formularios y llena el combo box de formularios con el resultado
cargaControles	-	-	Recupera todos los controles y llena el combo box de controles con el resultado
cargTSolucion	-	-	Recupera todos los tipos de solución y llena el combo box de tipos de solución con el resultado
cargaGrilla	-	-	Ejecuta getBusqueda y llena la grilla con los resultados.
getBusqueda	-	-	Lista todos los items de acuerdo a parámetros enviados.
SubEliminar	-	-	Inhabilita los items seleccionados.

- EstrategiaNueva.aspx: Permite la creación de nuevas estrategias.
- Estrategia.aspx: Página para mantenimiento de estrategias.

Tabla Anexo A.11 - Descripción de Funciones de la página Estrategia.aspx.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
cargaFormularios	-	-	Recupera todos los formularios y llena el combo box de formularios con el resultado
cargTSolucion	-	-	Recupera todos los tipos de solución y llena el combo box de tipos de solución con el resultado
cargaGrilla	-	-	Ejecuta getBusqueda y llena la grilla con los resultados.

getBusqueda	-	-	Lista todas las estrategias de acuerdo a parámetros enviados.
SubEliminar	-	-	Inhabilita las estrategias seleccionadas.

3.2. Módulo móvil

- **Carpeta src**

  - ActMenu.java: Clase que se encarga de la creación dinámica de los componentes del menú por medio de un control de grilla. Extiende de la clase Activity.

Tabla Anexo A.12 - Descripción de Funciones de la clase ActMenu.java.
Fuente: Elaboración propia

Método	Parámetros Entrada	Parámetros Salida	Descripción
onCreate	savedInstanceState	-	Establece el comportamiento de los controles creados en el menú por el método SubSetControles.
SubSetControles	-	-	Sitúa los controles en una Vista Grid para que puedan ser correctamente distribuidos en la pantalla del dispositivo móvil.

  - ActTipoSinc.java: Clase que sitúa al fichero layout "activity_act_tipsinc" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

Tabla Anexo A.13 - Descripción de Funciones de la clase ActTipoSinc.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
onCreate	Bundle savedInstanceState	-	Al momento de crearse la actividad se establece el funcionamiento de los

			controles.
onActivityResult	int requestCode, int resultCode, Intent intent	-	Este método es llamado cuando la actividad para obtener el código QR termina, procediendo a situar el texto leído por el escáner en el control EditText.
SincForm	String formSincCode	-	Método que se dedica exclusivamente a la sincronización del formulario ingresado mediante su identificador.

- ActTipoEv.java: Clase que sitúa al fichero layout "activity_act_tipev" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

Tabla Anexo A.14 - Descripción de Funciones de la clase ActTipoEv.java.
Fuente: Elaboración propia

Métodos	**Parámetros Entrada**	**Parámetros Salida**	**Descripción**
onCreate	Bundle savedInstanceState	-	Al momento de crearse la actividad obtiene los elementos que deben ser situados en el control spinner. También establece la manera en que el control botón funcionará, enviando el identificador del formulario seleccionado a otra clase.

- ActTipoRes.java: Clase que sitúa al fichero layout "activity_act_tipres" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

Tabla Anexo A.15 - Descripción de Funciones de la clase ActTipoRes.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
onCreate	Bundle savedInstanceState	-	Al momento de crearse la actividad obtiene los elementos que deben ser situados en el control spinner.

- CForm. java: Esta clase representa al objeto Form, almacenará sus atributos y métodos. Implementa la interface Parcelable que permite que las instancias del objeto puedan ser escritas y restauradas en la clase Parcel. La clase Parcel es un contenedor de datos y referencias de objetos que pueden ser enviados.

Tabla Anexo A.16 - Descripción de Funciones de la clase CForm.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
CForm	-	-	Constructor de la clase, inicializa los objetos contenidos en esta.
CForm	Parcel in	-	Constructor de la clase sobrecargado para recibir un objeto Parcel, para luego enviarlo al método "readFromParcel"
describeContents	-	int	Describe contenidos de la clase Parcelable.
writeToParcel	Parcel dest, int flags	-	Realiza una especie de compresión de los objetos en la clase contenedora Parcel.
readFromParcel	Parcel in	-	Realiza la lectura (descompresión) de los objetos desde la clase Parcel.

- CItem.java: Esta clase representa a cada ítem que compone un formulario.

Tabla Anexo A.17 - Descripción de Funciones de la clase CItem.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
CItem	-	-	Constructor de la clase, inicializa los objetos contenidos en esta.
CItem	Parcel in	-	Constructor de la clase sobrecargado para recibir un objeto Parcel, para luego enviarlo al método "readFromParcel"
describeContents	-	int	Describe contenidos de la clase Parcelable.
writeToParcel	Parcel dest, int flags	-	Realiza una especie de compresión de los objetos en la clase contenedora Parcel.
readFromParcel	Parcel in	-	Realiza la lectura (descompresión) de los objetos desde la clase Parcel.

- CMapper.java: Esta clase se encarga de la conversión de un objeto desde y hacia el formato Json.

Tabla Anexo A.18 - Descripción de Funciones de la clase CMapper.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
fromJson	String jsonAsString, Class<T> BeanClass	Object	Recibe una cadena Json y lo convierte en un Objeto.
toJson	Object Bean, boolean prettyPrint	String	Convierte el objeto ingresado a una cadena Json y si la variable prettyPrint es "true", la cadena se imprime con saltos de línea e indentaciones que agregan facilidad al momento de ser leído el Json por el usuario.

- DBHandler.java: Extiende de la clase SQLiteOpenHelper la cual nos permite crear la base de datos y actualizar la estructura de las tablas y manejar datos dentro de ella.

Tabla Anexo A.19 - Descripción de Funciones de la clase DBHandler.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
DBHandler	Context context	-	Es el constructor de la clase DBHandler, sitúa el nombre de la base de datos y su versión.
onCreate	SQLiteDatabase db	-	Se le llama cuando la base de datos se crea por primera vez. Aquí es donde se define la estructura de las tablas y se cargan eventualmente los datos iniciales
onUpgrade	SQLiteDatabase db, int oldVersion, int newVersion	-	Recibe el nombre de la base de datos y llama al método onCreate para volver a crear la base de datos.
sincForm_setForm	String formSincCode	boolean	Inserta los datos del formulario sincronizado en la base de datos.
actTipoEv_getList	-	String[][]	Selecciona todos los formularios que estén activos en la base de datos del dispositivo móvil.
runForm_getForm	CForm form, String formCode	CForm	Recupera registros de la base de datos para almacenarlo en la clase CForm.
resolForm_setItemsData	CForm theForm	-	Inserta en la base de datos los datos del formulario ingresados por el usuario.
resolForm_setResult	CForm theForm	-	Inserta el resultado final

	,Float result, String strname		de la evaluación en la base de datos.
actResult_getStrlabel	CForm theForm	String	Obtiene el texto que fue parametrizado como resultado en el módulo web.
actResult_getResval	CForm theForm	Float	Obtiene el valor del resultado guardado anteriormente en la base de datos.
actResult_setName	CForm theForm, String formName	boolean	Inserta el nombre que se le da a una resolución determinada de un formulario.
actTipoRes_getList	-	String[][]	Obtiene la lista de las resoluciones de formularios guardadas en el dispositivo móvil.
actResultD_getForm	CForm theForm, int formID, int version	CForm	Obtiene el formulario según su identificador y su versión.

- XmlGuiEditText.java: Esta clase contiene al control EditText el cuál será creado dinámicamente, la clase extiende de LinearLayout, que facilita su posición dentro de la vista que se mostrará al usuario.

Tabla Anexo A.20 - Descripción de Funciones de la clase XmlGuiEditText.java. Fuente: Elaboración propia

Métodos	**Parámetros Entrada**	**Parámetros Salida**	**Descripción**
XmlGuiEditText	Context context,String labelText,String initialText	-	Constructor de la clase. Establece las propiedades del control Label y EditText.
makeNumeric	-	-	Hace que la información ingresada en el control EditText sea mediante teclado numérico.

getValue	-	String	Obtiene el texto del control EditText.
setValue	String v	-	Establece el String entrante como valor del texto del EditText.

- XmlGuiChoice.java: Contiene el control Spinner, el cuál se creará dinámicamente.

Tabla Anexo A.21 - Descripción de Funciones de la clase XmlGuiChoice.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
XmlGuiChoice	Context context,String labelText,String options		Constructor de la clase. Establece las propiedades del control Label y Spinner.
getValue	-	String	Obtiene el texto del item seleccionado.
getPosition	-	String	Obtiene el índice de posición del item seleccionado.

- RunForm.java: Esta clase carga los datos del formulario sincronizado y se encarga de situar los controles dinámicamente en la vista para ser mostrada en el dispositivo móvil.

Tabla Anexo A.22 - Descripción de Funciones de la clase RunForm.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
onCreate	Bundle savedInstanceState	-	Constructor de la clase. Extrae los datos del formulario electo.
DisplayForm	-	boolean	Sitúa dinámicamente los controles en la vista.
CheckForm	-	boolean	Revisa si los campos requeridos están vacíos o no.

SaveDataForm	-	boolean	Guarda datos ingresados en la clase CItemData.

- ResolForm.java: Esta clase se encarga de la resolución de las respuestas ingresadas por el usuario.

Tabla Anexo A.23 - Descripción de Funciones de la clase ResolForm.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
fnResolutor	CForm theForm,Context context	-	Constructor de la clase. Establece los tipos de resolución con que se tratarán los datos.
fnResolutor	String type,CForm theForm, int i	String	Resuelve y extrae el puntaje correspondiente.
fnEvalLimits	String itemVal, String itemDatVal	boolean	Evalúa si el valor de ambos ítems está dentro de los límites establecidos.
fnGenerateResults	CForm theForm ,String result	String	Evalúa el resultado final según estrategias.

- SendForm.java: Esta clase se carga los enviar los datos del formulario resuelto a la dirección especificada dentro del formulario.

Tabla Anexo A.24 - Descripción de Funciones de la clase SendForm.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
onCreate	Bundle savedInstanceState	-	Constructor de la clase. Crea el diálogo de espera mientras transmite datos.
TransmitFormData	CForm theForm	String	Conecta a la aplicación con la dirección del servidor descrito en el campo URL del formulario.

- HttpSincronizar.java: Esta clase sirve para hacer la conexión con el servidor al sincronizar un archivo.

Tabla Anexo A.25 - Descripción de Funciones de la clase HttpSincronizar.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
SubConHttp	-	-	Llama al manejador de la base de datos para sincronizar el formulario.
fnSincronizar	Context setcontext, EnumUrl poEnumUrl, String formSincCode	BufferedReader	Hace la conexión para la sincronización.

- Configuracion.java: Comprende más que todo las rutas de archivos que podremos configurar más adelante.

Tabla Anexo A.26 - Descripción de Funciones de la clase Configuración.java.
Fuente: Elaboración propia

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
fnUrl	Context poContext, EnumUrl poEnumUrl , String formSincCode	String	Asigna las rutas del servidor con el que se hará conexión y sus archivos.

- Utilitario.java: En esta clase se encuentran los métodos utilitarios listos para ser utilizados.

Tabla Anexo A.27 - Descripción de Funciones de la clase Utilitario.java.
Fuente: Elaboración propia.

Métodos	Parámetros Entrada	Parámetros Salida	Descripción
fnVerSignal	Context poContext	boolean	Verifica si el dispositivo móvil cuenta con una señal wi-fi o 3g activa.
fnVerStorage	-	boolean	Verifica si el dispositivo móvil

			posee memoria externa que tenga permisos para que se pueda leer y escribir en ella.
fnNumEquipo	Context poContext	String	Obtiene el número de celular del dispositivo móvil.
fnRedondeo	double pdValor, int piPrecision	double	Devuelve el valor ingresado redondeando a este a la precisión deseada.
fnFechaFormato	String psFecha	String	Le da un formato definido a la fecha ingresada.

- **Carpeta libs**
  - android-support-v4.jar: Es una librería de soporte de Android, proporciona soporte adicional para desarrollar funciones específicas. Permite utilizar algunas de las APIs más recientes de Android aunque nuestra aplicación se ejecute en versiones más antiguas. Esta librería nos permite ejecutar en Android 1.6 (nivel API 4) y superiores.
  - jackson-all-1.8.1.jar: Es una librería de utilidad de Java que nos simplifica el trabajo de serializar (convertir un objeto Java en una cadena de texto con su representación JSON), y deserializar (convertir una cadena de texto con una representación de JSON de un objeto en un objeto real de Java) objetos JSON.

- **Carpeta res/layout:** Un layout define la estructura visual de para determinada interfaz de usuario como una activity.
  - activity_act_menu.xml: Este layout está compuesto por una vista Grid que hace que los elementos cargados en el menú puedan ser correctamente distribuidos a lo largo de la pantalla del dispositivo móvil.
  - activity_act_tipsinc.xml: Este layout muestra un control EditText en el cuál se podrá ingresar el identificador del formulario que se quiere sincronizar.
  - activity_act_tipev.xml: Permite elegir el tipo de formulario que queremos resolver.
  - activity_act_tipres.xml: Nos muestra en una lista las resoluciones que ingresamos de todos los formularios y nos permite revisar estas respuestas en detalle.

- activity_act_result.xml: Este layout muestra los items evaluados indicando si la respuesta que ingresamos es correcta o no.
- acitvity_act_resultmain.xml: Muestra el resultado final de determinada resolución de determinado formulario, incorpora en su vista la lista generada en “activity_act_result” dando la opción de desplazar arriba y abajo esta lista.
- activity_act_send.xml: Este layout nos indica si la resolución de formulario enviado al servidor obtuvo una respuesta favorable.

- Carpeta res/values
  - strings.xml: El fichero strings.xml permite definir y almacenar cadenas de texto (strings) utilizando el formato XML para su definición que pueden ser utilizadas en cualquier fichero xml o código android.
  - styles.xml: Fichero donde se define el formato y vista de la interfaz de usuario, puede ser aplicada a una vista individual desde un archivo layout o a la actividad entera y aplicación desde el Android Manifest.

- Carpeta res/xml
  - custom_button_green.xml: En este archivo están definidos los estilos de un botón conforme al color de fondo, de línea y forma de éste.
  - custom_button_red.xml: Tiene definidos los estilos de un botón conforme al color de fondo, de línea y forma de éste.
  - custom_edittext.xml: Define los estilos de un control EditText.
  - custom_spinner.xml: Están establecidos los estilos del control Spinner.
  - custom_spinnerlist.xml: Define el estilo visual que se le podrá aplicar a la lista de un control Spinner.

4. UTILIZACIÓN DEL FRAMEWORK

4.1. Configuraciones Iniciales

4.1.1. Módulo Web

a) Verificar el correcto funcionamiento de IIS (Internet Information Services).

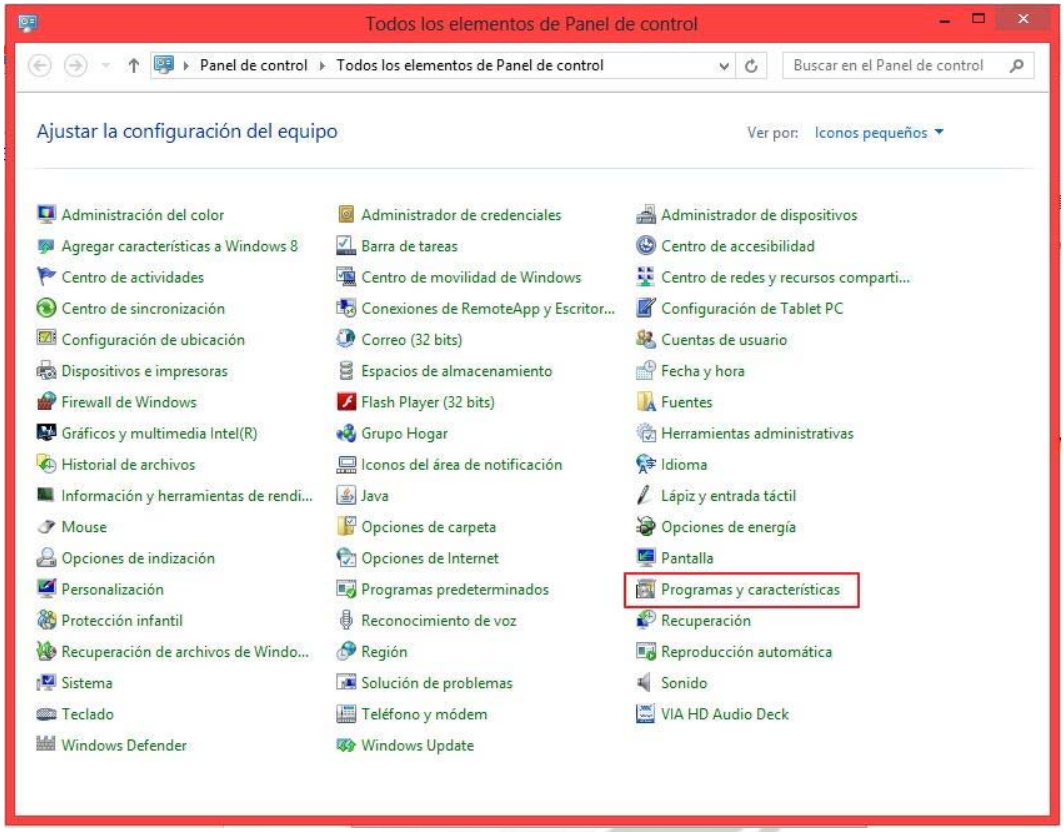

Figura Anexo A.3 - Panel de Control.
Fuente: Elaboración propia

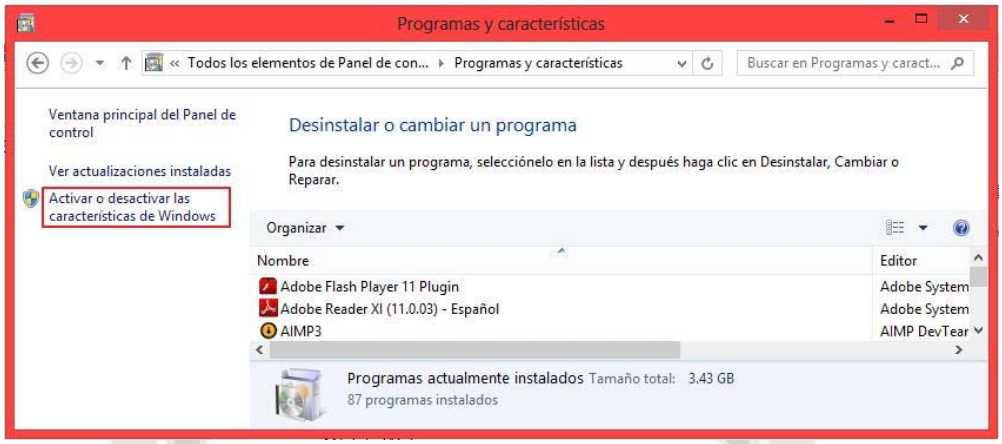

Figura Anexo A.4 - Programas y características.
Fuente: Elaboración propia

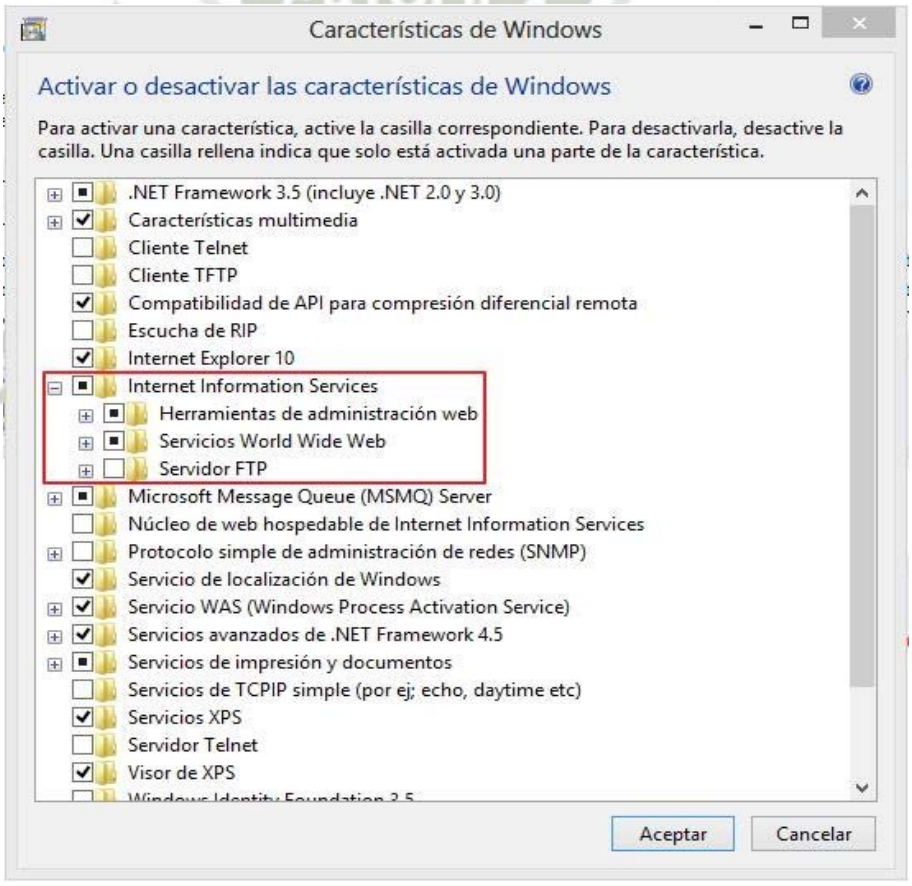

Figura Anexo A.5 - Características de Windows.
Fuente: Elaboración propia

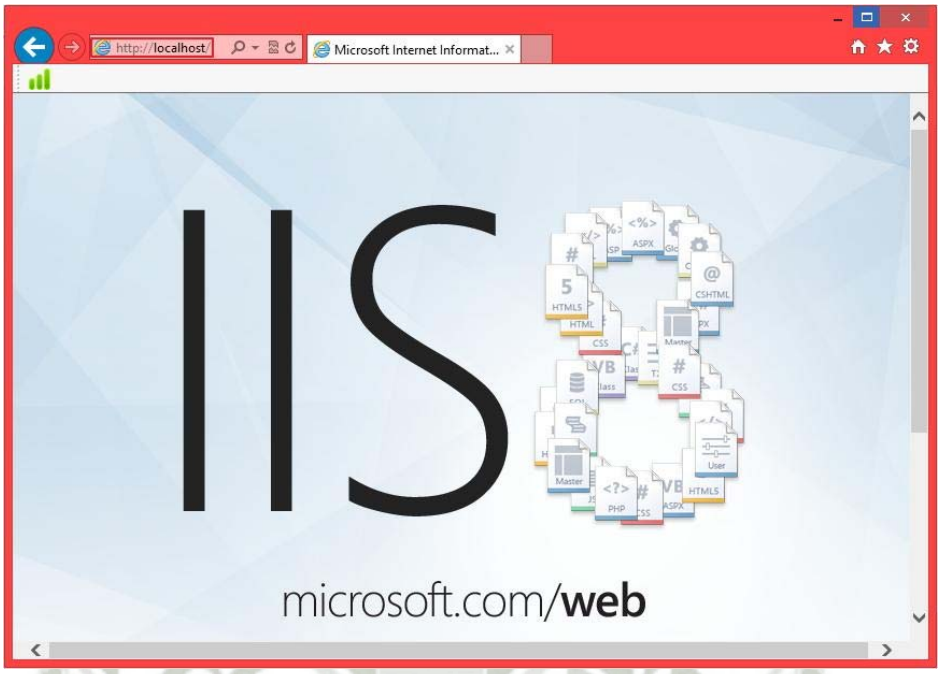

Figura Anexo A.6 - Funcionamiento IIS.
Fuente: Elaboración propia

b) Descomprimir el archivo "FRAMEWORK_DATILE.rar", encontrará 3 carpetas: Web, Móvil y DB. Copiar la carpeta Web al directorio c:\inetpub\wwwroot.

c) Iniciar el Administrador de IIS (Internet Information Services) y cree un nuevo sitio web.

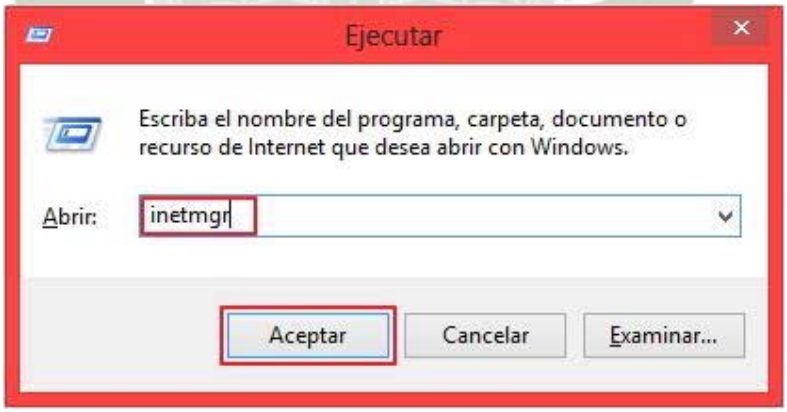

Figura Anexo A.7 - Iniciar Administrador de IIS.
Fuente: Elaboración propia

En la siguiente figura se muestra la ejecución del IIS desde "Ejecutar"

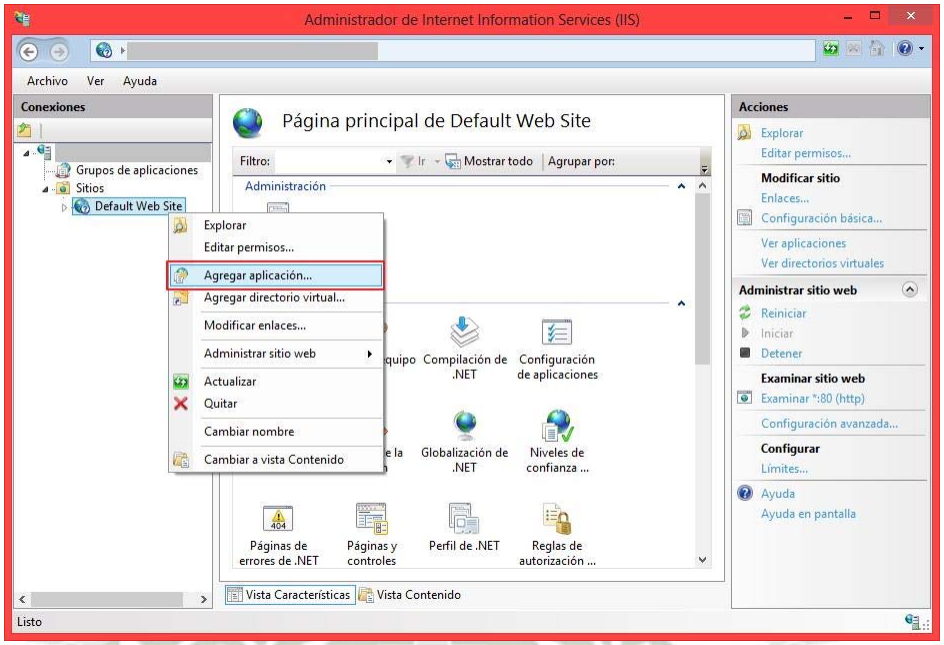

Figura Anexo A.8 - Administrador de IIS.
Fuente: Elaboración propia

En la siguiente figura se muestra el Administrador de IIS

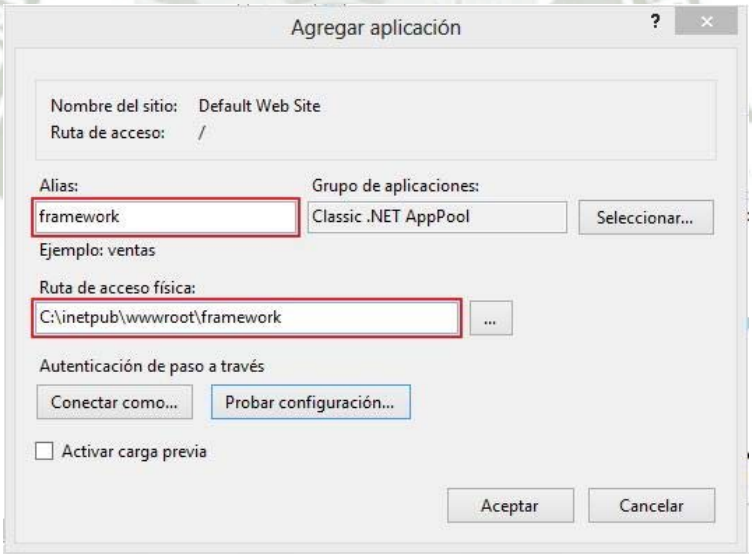

Figura Anexo A.9 - Administrador de IIS.
Fuente: Elaboración propia

d) Crear una base de datos nueva y restaurar el backup ubicado en la carpeta DB.

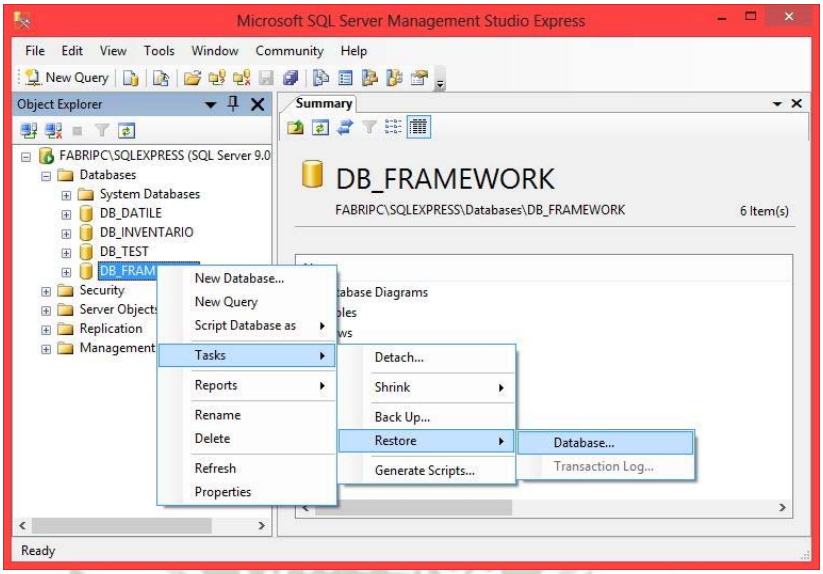

Figura Anexo A.10 - Microsoft SQL Server Management Studio.
Fuente: Elaboración propia

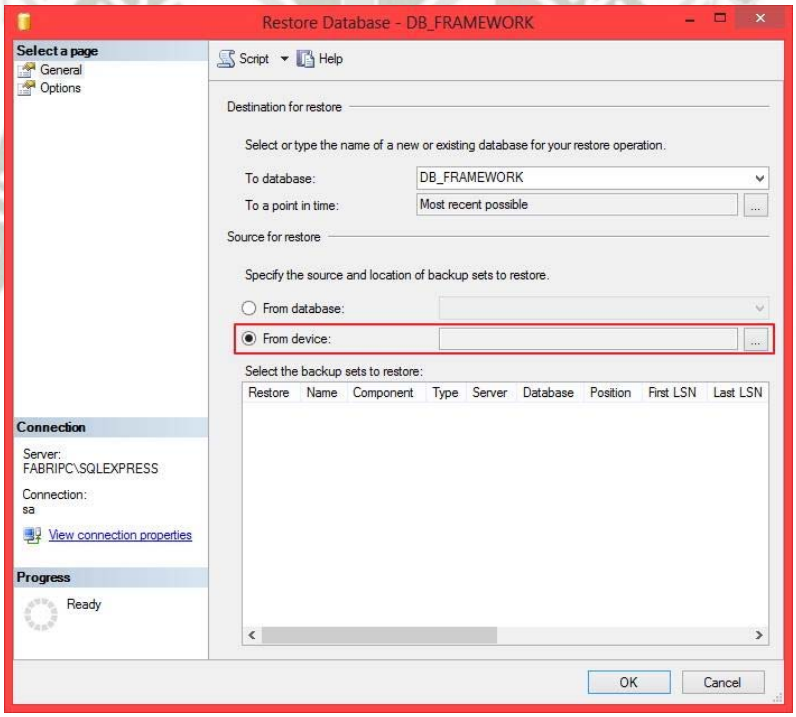

Figura Anexo A.11 - Restaurar Base de Datos.
Fuente: Elaboración propia

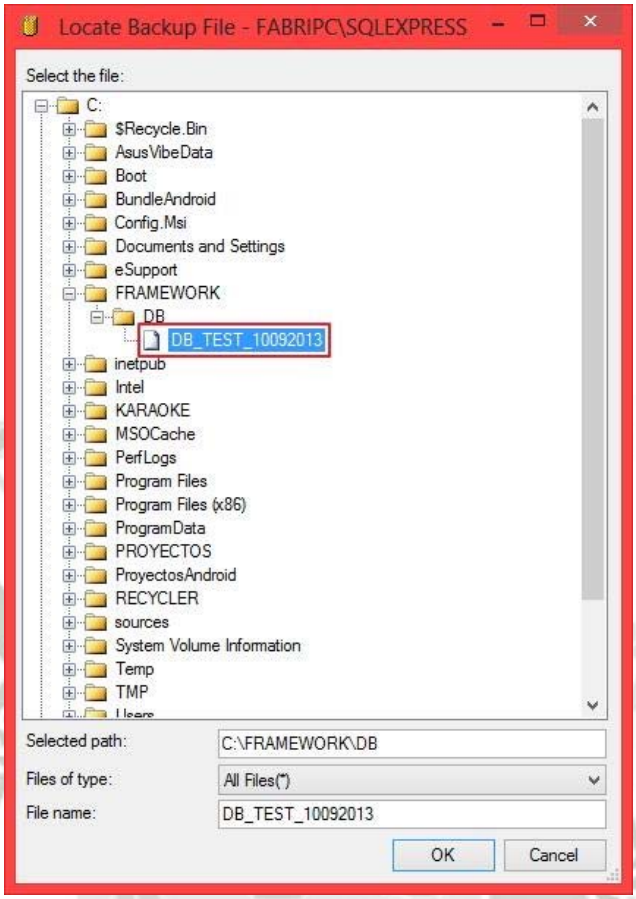

Figura Anexo A.12 - Seleccionar archivo Backup.
Fuente: Elaboración propia

e) Para efectuar algún cambio en el módulo web, deberá abrir el IDE (Integrated Development Enviroment) Visual Studio e importar el proyecto web. Específicamente para el archivo web.config se debe cambiar la cadena de conexión con la base de datos.

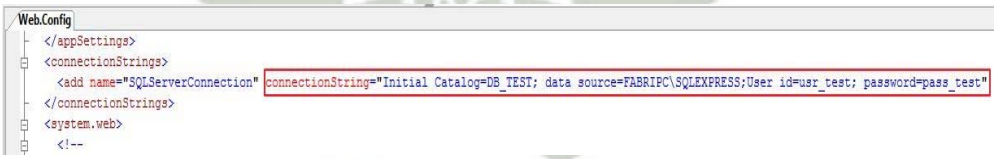

Figura Anexo A.13 - Edición Web.config.
Fuente: Elaboración propia

4.1.2. Módulo móvil

a) Abrir el IDE Eclipse e importar el proyecto “DATILE” situado en la carpeta Móvil. Esto se hace seleccionando la opción File en el menú de herramientas,

luego la opción Import, seguidamente “Existing Projects into Workspace” y se selecciona el proyecto deseado.

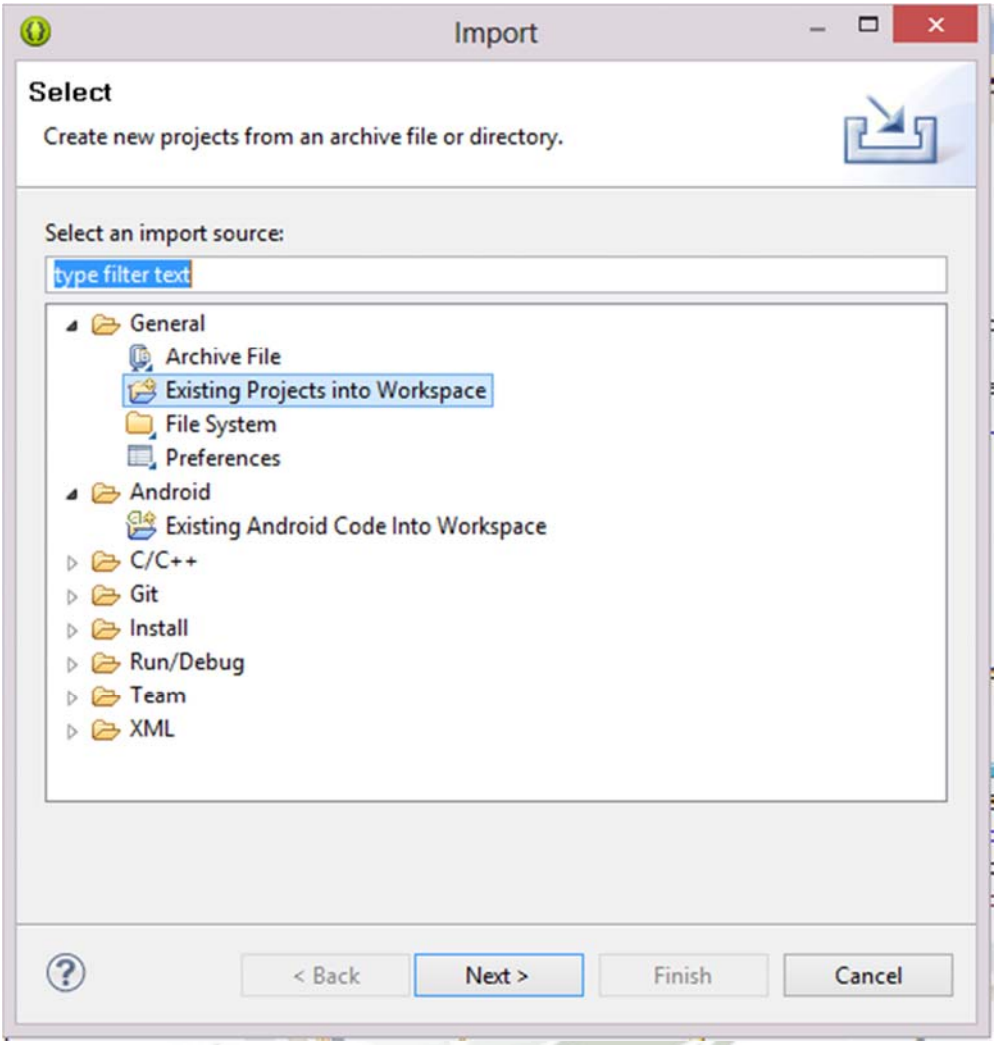

Figura Anexo A.14 - Importar proyecto móvil.
Fuente: Elaboración propia

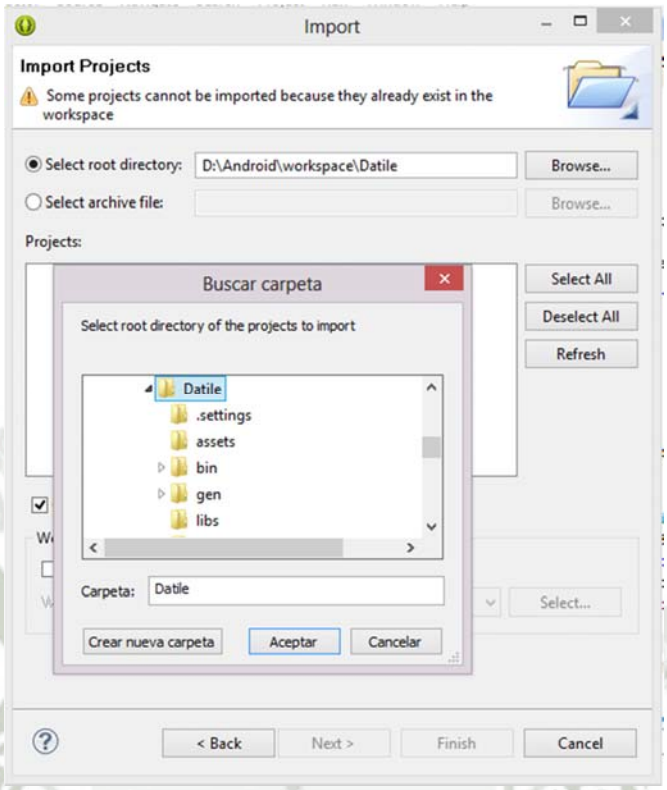

Figura Anexo A.15 - Seleccionar proyecto móvil.
Fuente: Elaboración propia

b) Ahora se necesita apuntar al servidor para poder descargar los formularios, para este caso la dirección del servidor está en el archivo “strings.xml” de la carpeta “values”.

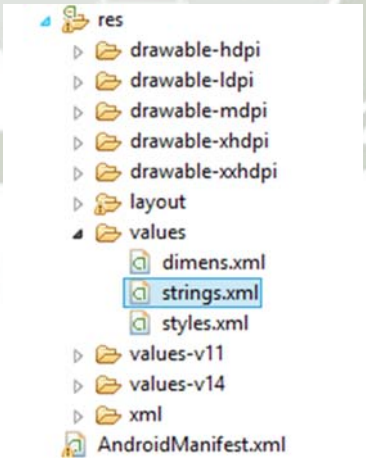

Figura Anexo A.16 - Configurar variables.
Fuente: Elaboración propia

Una vez ingresado al archivo, buscar la variable “app_urlserver” y reemplazar la dirección por la dirección de su servidor donde se encuentran almacenados los formularios. Para este caso, nuestra dirección es “192.168.1.33:8080/prueba”.

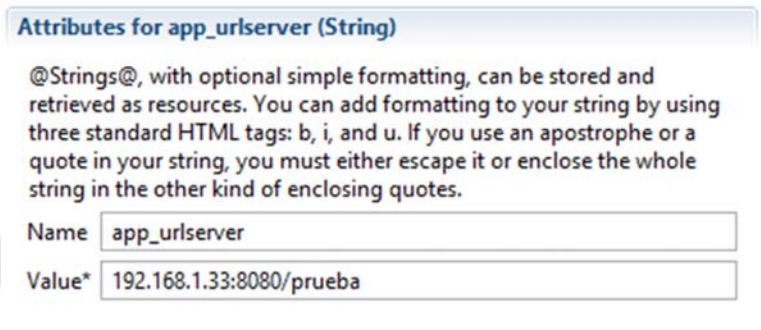

Figura Anexo A.17 - Configurar variable app_urlserver.
Fuente: Elaboración propia

c) Para generar el instalador del aplicativo y probarlo en un dispositivo virtual o físico se selecciona el proyecto y presionando la opción de Menú Contextual del mouse seleccionar “Run As” y seguidamente “Android Application”. La aplicación Android se instalará en el dispositivo y su consola empezará a escribir los datos sobre el estado de la aplicación.

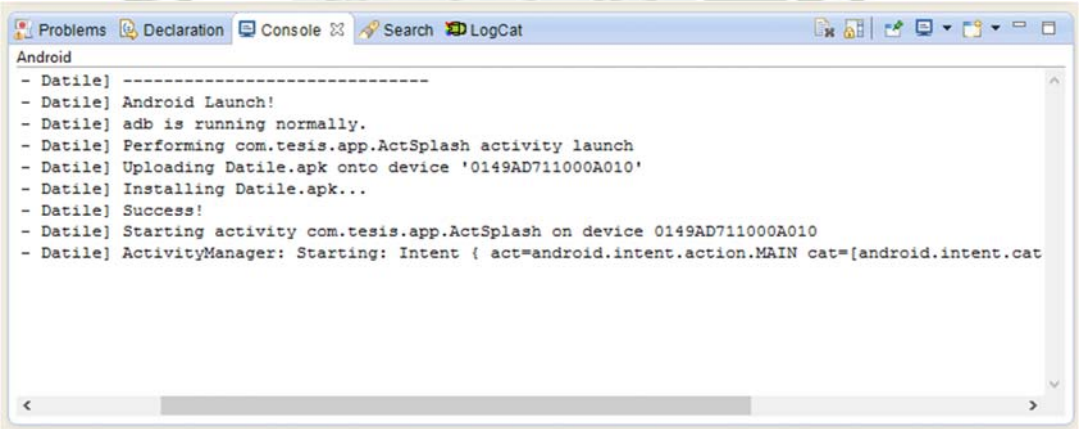

Figura Anexo A.18 - Consola Eclipse.
Fuente: Elaboración propia

4.2 Implementación

Después de haber verificado el funcionamiento del módulo web y de la aplicación móvil podremos continuar con la implementación en base al Framework, para esto se toma como ejemplo la creación de un nuevo componente gráfico.

4.2.1. Módulo web

PASO 1.- Para agregar un nuevo componente gráfico primero debe ser agregado en la base de datos, se debe ejecutar la sentencia con el nombre de control que se desee implementar.

```
INSERT INTO TBM_CTRLTYPE VALUES('GPS')
```

PASO 2.- Seguidamente se ingresa al documento "ScriptDB.txt" de la carpeta Scripts y en la parte de la creación de la tabla "TBM_CTRLTYPE" agregar una sentencia SQL insertando el identificador y nombre del control que se creará. Esta inserción se hará en la base de datos del dispositivo móvil.

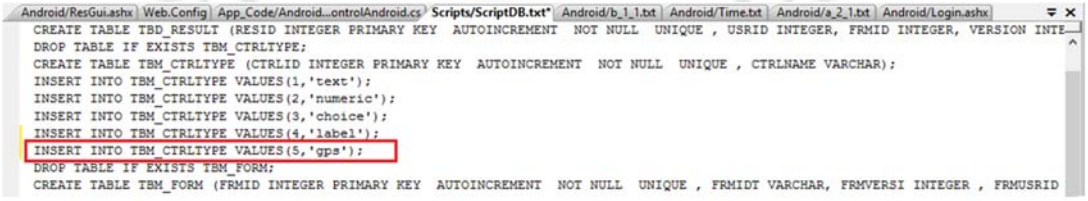

Figura Anexo A.19 - Archivo ScriptDB.txt.
Fuente: Elaboración propia

4.2.2. Módulo móvil

PASO 3.- Primeramente debemos fijarnos cómo están compuestos los diferentes controles que el Framework posee para poder seguir su modelo, fíjese que todos ellos heredan propiedades de la clase LinearLayout, estos se encuentran en el paquete "com.tesis.app.widget".

PASO 4.- En el Entorno Integrado de Desarrollo Eclipse agregaremos un nuevo control, para esto situarse en el paquete "com.tesis.app.widget" de la carpeta "src" del proyecto y luego hacer click derecho sobre dicho paquete y crear una nueva clase. Nombrarla siguiendo la estructura de los otros elementos, por ejemplo "XmlGuiGps". Aceptar.

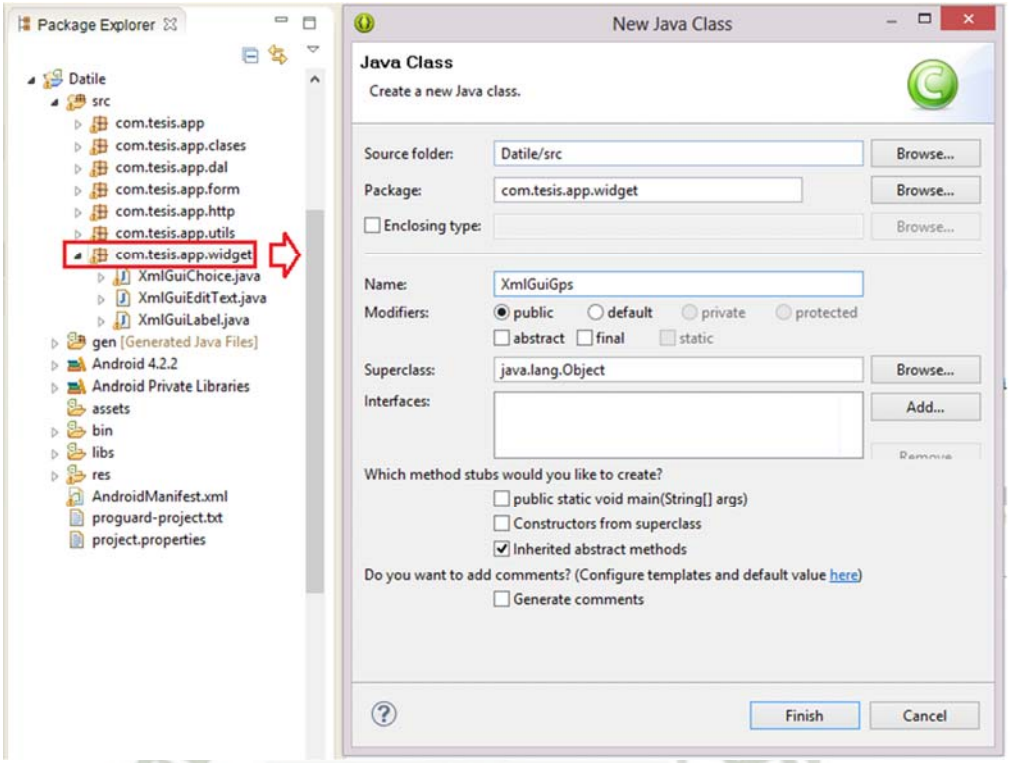

Figura Anexo A.20 - Agregar nueva Clase.
Fuente: Elaboración propia

PASO 5.- Para empezar a trabajar con esta clase, debemos hacer que herede métodos de la clase LinearLayout para que luego pueda ser tratada correctamente dentro de una vista. Seguidamente declararemos los controles que se utilizarán.

```
public class XmlGuiGps  extends LinearLayout  {
  TextView label;
  EditText txtBox;
  EditText txtBox1;
  Button buttonGps;

  public  XmlGuiGps(Context  context,String  labelText,String
buttonText) {
        super(context);
  }
  public XmlGuiGps(Context context, AttributeSet attrs) {
        super(context, attrs);
  }
}
```

PASO 6.- Así quedaría la implementación del widget definiendo sus atributos y acciones mediante programación, lo cuál permite que pueda ser creado dinámicamente:

```
public class XmlGuiGps  extends LinearLayout {
  TextView label;
  EditText txtBox;
  EditText txtBox1;
  Button buttonGps;

  public      XmlGuiGps(final      Context      context,String
labelText,String buttonText) {
        super(context);
        this.setOrientation(LinearLayout.VERTICAL);

        try {
              label = new TextView(context);
              label.setTextSize(16);
              label.setText(labelText);

              txtBox = new EditText(context);
              txtBox.setEnabled(false);
              txtBox1 = new EditText(context);
              txtBox1.setEnabled(false);

              buttonGps = new Button(context);
              buttonGps.setBackgroundResource
                        (R.xml.custom_button_red);
              LayoutParams params = new LayoutParams(
                      LayoutParams.WRAP_CONTENT,
                      LayoutParams.WRAP_CONTENT
              );
              //params.setMargins(0,20,0,0);
              buttonGps.setLayoutParams(params);
              buttonGps.setText(buttonText);
              buttonGps.setOnClickListener(new
Button.OnClickListener() {
                    public void onClick(View v) {
                          LocationManager   locationManager   =
(LocationManager)context.getSystemService(Context.LOCATION_SE
RVICE);
                          Location          location          =
locationManager.getLastKnownLocation(LocationManager.NETWORK_
PROVIDER );
                      if (location!=null){
                        double           longitude           =
location.getLongitude();
                        double            latitude            =
location.getLatitude();
                        txtBox.setText("Latitud:
"+String.valueOf(location.getLatitude()));
                              txtBox1.setText("Longitud:
"+String.valueOf(location.getLongitude()));
                    }
                      else
                      {
```

```
                        txtBox.setText("No        se        puede
                        determinar.");
                    }
                }
            } );

            this.addView(label);
            this.addView(txtBox);
            this.addView(txtBox1);
            this.addView(buttonGps);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    //---------------- SETTERS & GETTERS --------------------
    public String getValue()
    {
        return
txtBox.getText().toString()+"/"+txtBox1.getText().toString();
    }
}
```

PASO 7.- En la clase "RunForm.java" se tiene que agregar el código fuente que se ejecutará en el método DisplayForm().

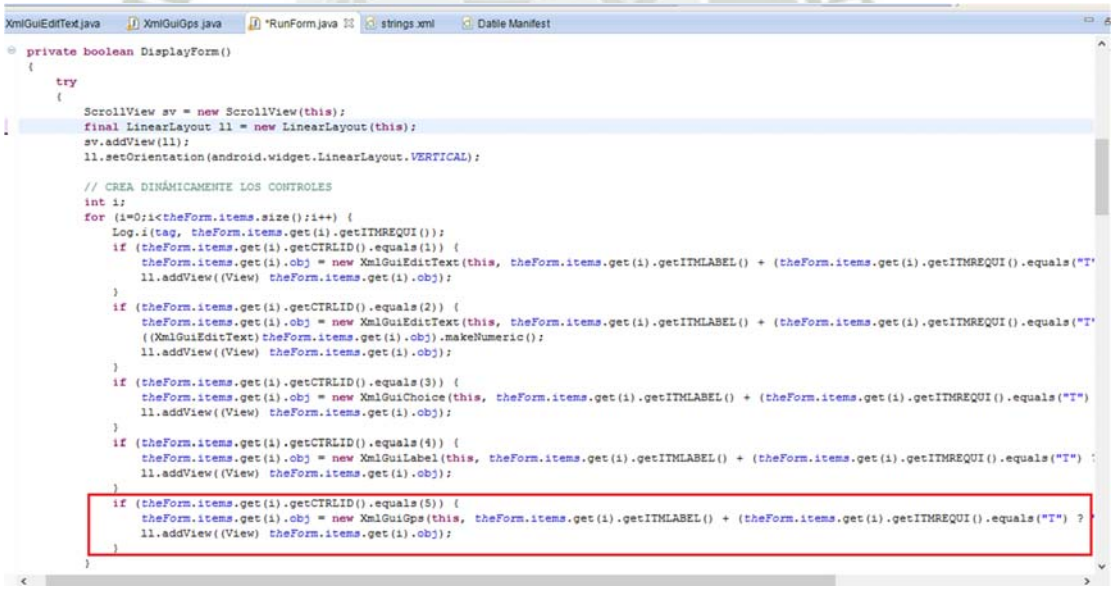

Figura Anexo A.21 - Agregar nueva control en clase RunForm.java.
Fuente: Elaboración propia

Se muestra una breve explicación del código agregado a la clase RunForm.java:

Tabla Anexo A.28 - Código agregado a la clase "RunForm.java"
Fuente: Elaboración propia.

Código fuente	Explicación
`if (theForm.items.get(i).getCTRLID( ).equals(5)) {`	Si el formulario a mostrar contiene un control con identificador = 5
`theForm.items.get(i).obj = new XmlGuiGps(`	El objeto perteneciente a la clase Items almacenará la información de la instanciación de un nuevo widget de clase "XmlGuiGps"
`this,`	1er parámetro: Contexto
`theForm.items.get(i).getITMLABEL() + ( theForm.items.get(i).getITMREQUI().equals("T") ? "*" : ""),`	2do parámetro: El label establecido para esa vista y si es requerido que sea llenado por el usuario se le agrega un asterisco (*) al final del label.
`"Obtiene GPS");`	3er parámetro: El texto que aparecerá en el botón.
`ll.addView((View) theForm.items.get(i).obj); }`	La vista es agregada al LinearLayout definido con anterioridad.

PASO 8.- En el archivo Manifest necesitamos agregar el permiso correspondiente a una lectura de GPS:

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

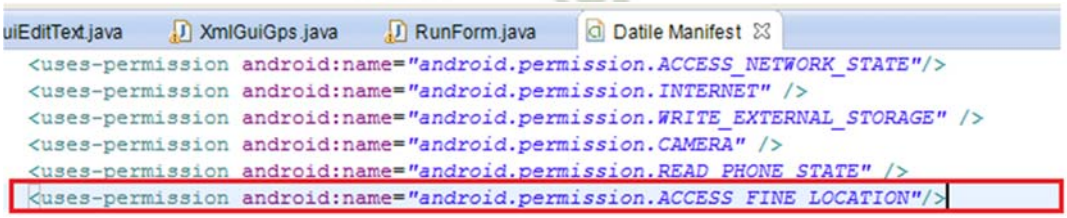

Figura Anexo A.22 - Agregar permisos en Manifest.xml.
Fuente: Elaboración propia

PASO 9.- En la clase CItem situada en el paquete “com.tesis.app.clases” se debe agregar el identificador del control y la acción que queremos que realice en el método GetData().

```
public Object getData()
{
    if (CTRLID.equals(1) || CTRLID.equals(2)) //1 = Text y 2 = Numeric
    {
        if (obj != null) {
            XmlGuiEditText b = (XmlGuiEditText) obj;
            return b.getValue();
        }
    }
    if (CTRLID.equals(3)) { // 3 = Choice
        if (obj != null) {
            XmlGuiChoice po = (XmlGuiChoice) obj;
            return po.getPosition();
        }
    }
    if (CTRLID.equals(4)) { // 4 = label
        return "";
    }
    if (CTRLID.equals(5)) { // 5 = gps
        if (obj != null) {
            XmlGuiGps b = (XmlGuiGps) obj;
            return b.getValue();
        }
    }
    return null;
}
```

Figura Anexo A.23 - Agregar acción de nuevo control.
Fuente: Elaboración propia

Se presenta el código agregado:

```
if (CTRLID.equals(5)) { // 5 = gps
  if (obj != null) {
        XmlGuiGps b = (XmlGuiGps) obj;
        return b.getValue();
  }
}
```

PASO 10.- Si todos los pasos se siguieron correctamente, usted podrá almacenar la latitud y longitud de su ubicación en cualquier momento al presionar el botón perteneciente al widget implementado, el resultado final se muestra en la siguiente figura.

Figura Anexo A.24 - Resultado de implementación del control GPS.
Fuente: Elaboración propia

4.3 Aplicación

Una vez realizados los cambios en el código fuente procederemos a parametrizar dos formularios los cuales ejemplifican la capacidad de toma de datos y de resolución en base a reglas parametrizadas.

4.3.1. Caso 1: Formulario Solicitud de crédito.

Tabla Anexo A.29 - Items formulario 1: Solicitud de Crédito.
Fuente: Elaboración propia.

Nombre	Label	Opciones	Valores	Puntajes	Control	Tip. Sol.	Req.	Res.
INFO	Formulario para solicitud de créditos.	-	-	-	Label	-	-	-
NOMRZ	Nombre/Razón Social	-	-	-	Text	-	Si	-
DNIRUC	DNI/RUC	-	-	-	Text	-	Si	-
EDAD	Edad	-	-	-	Numeric	-	Si	No
DIRECCION	Dirección	-	-	-	Text	-	Si	-
UBICACION	Ubicación	-	-	-	GPS	-	No	-
TLFCEL	Telf/Cel	-	-	-	Text	-	Si	-
ACTIVIDAD	Actividad	Abarrotes, Ferreteria, Agricultura	-	-	Choice	-	Si	No
TIPOCRED	Tipo de crédito	Pequeña empresa, Personal, Vivienda	-	-	Choice	-	Si	No
MONTO	Monto	-	-	-	Numeric	-	Si	No
PLAZO	Plazo (Meses)	-	-	-	Numeric	-	Si	No
OBSERV	Observaciones	-	-	-	Text	-	Si	-

Para parametrizar este formulario, ingresamos al módulo web del framework.

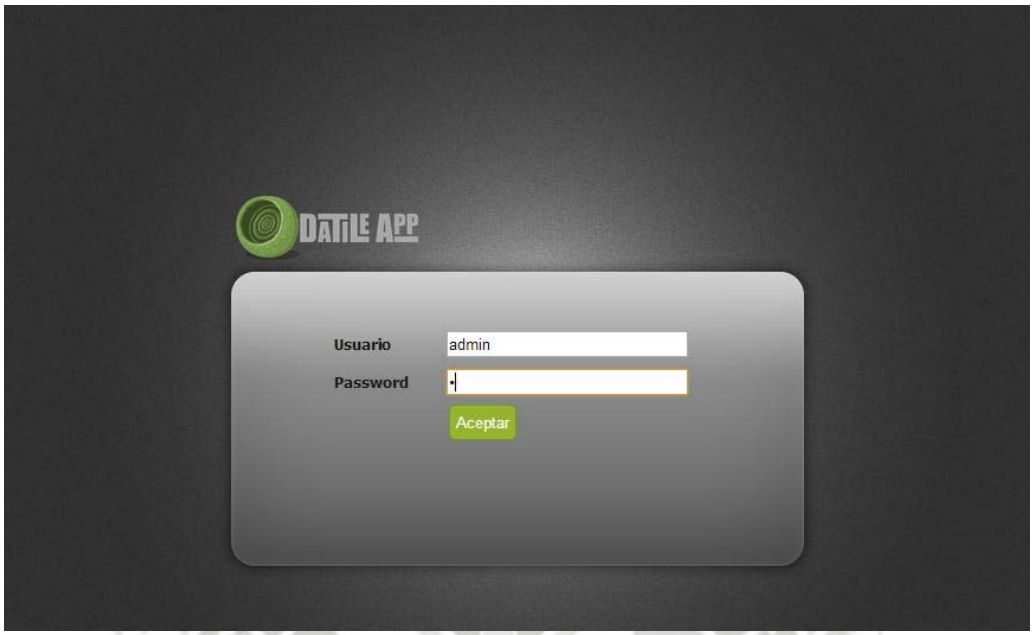

Figura Anexo A.25 - Formulario1: Ingreso a módulo web.
Fuente: Elaboración propia

Creamos el nuevo formulario.

Figura Anexo A.26 - Formulario1: Mantenimiento de Formularios.
Fuente: Elaboración propia

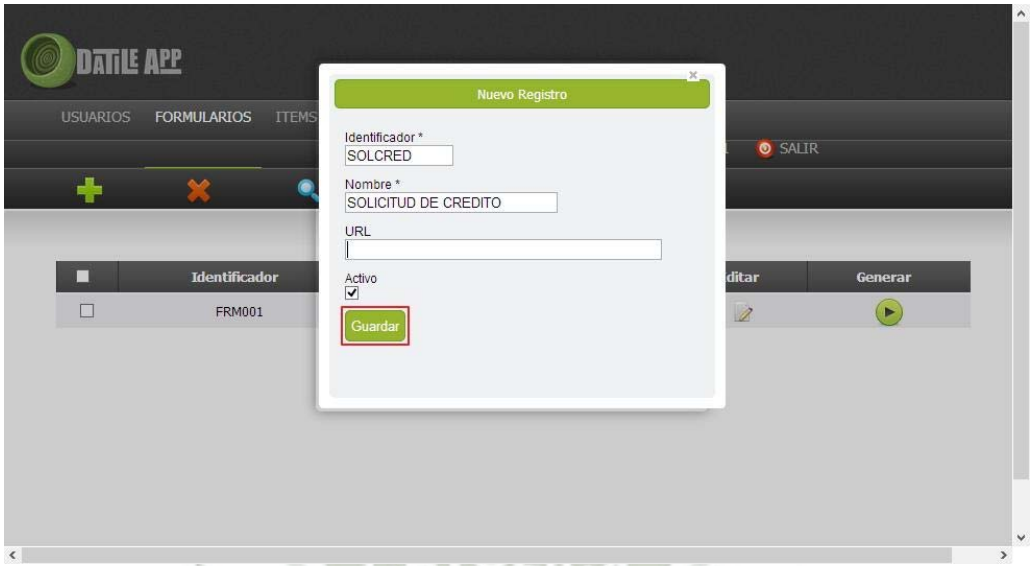

Figura Anexo A.27 - Creación de formulario: Caso 1.
Fuente: Elaboración propia

Creamos todos los items necesarios.

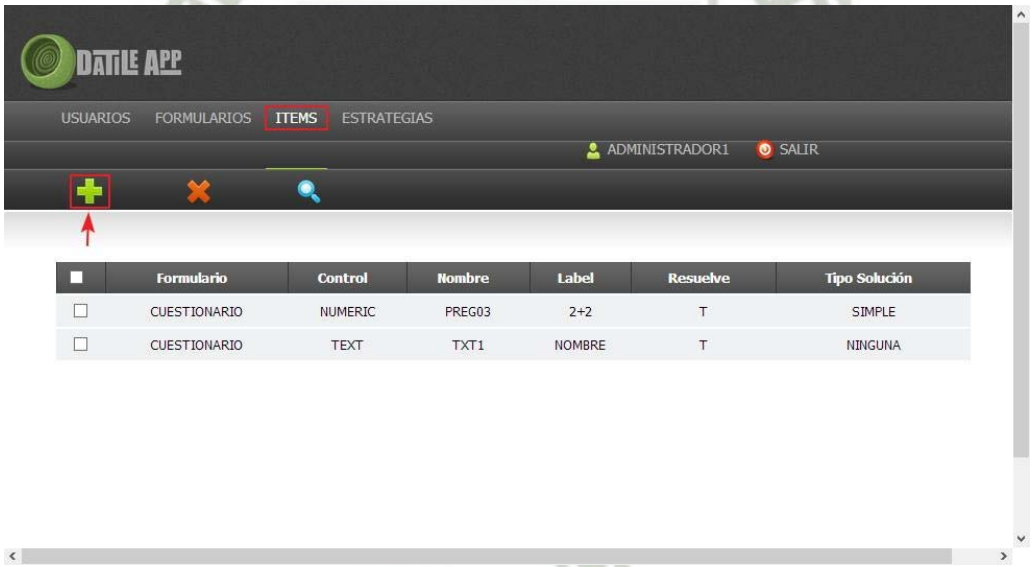

Figura Anexo A.28 - Formulario1: Mantenimiento de Ítems.
Fuente: Elaboración propia

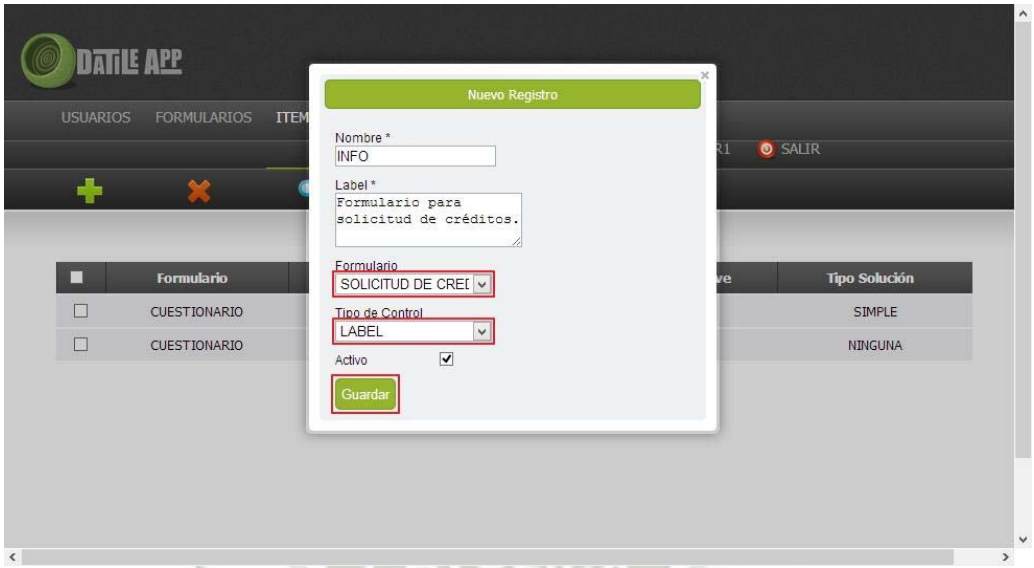

Figura Anexo A.29 - Formulario1: Agregar Ítem 1
Fuente: Elaboración propia

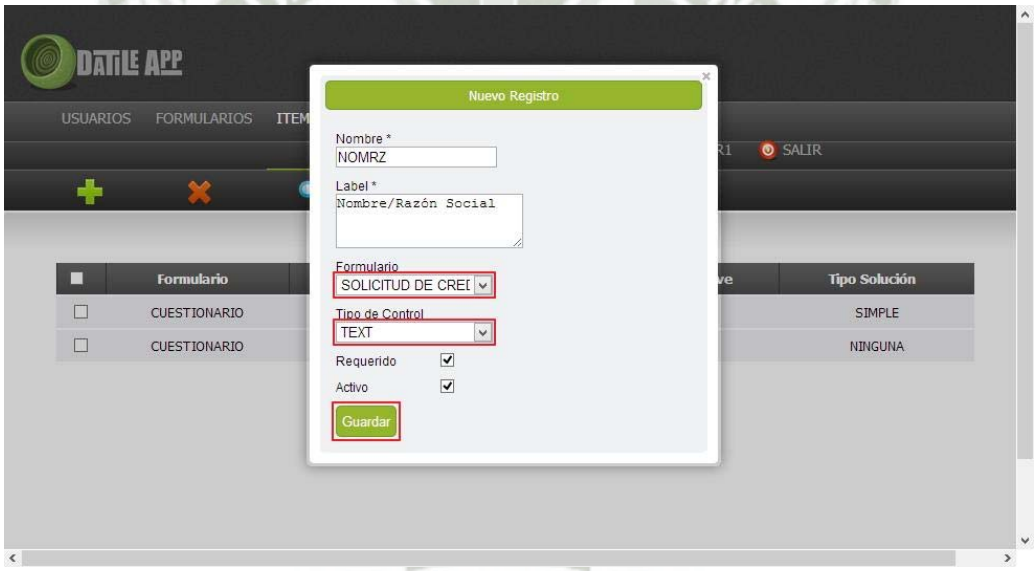

Figura Anexo A.30 - Formulario1: Agregar Ítem 2
Fuente: Elaboración propia

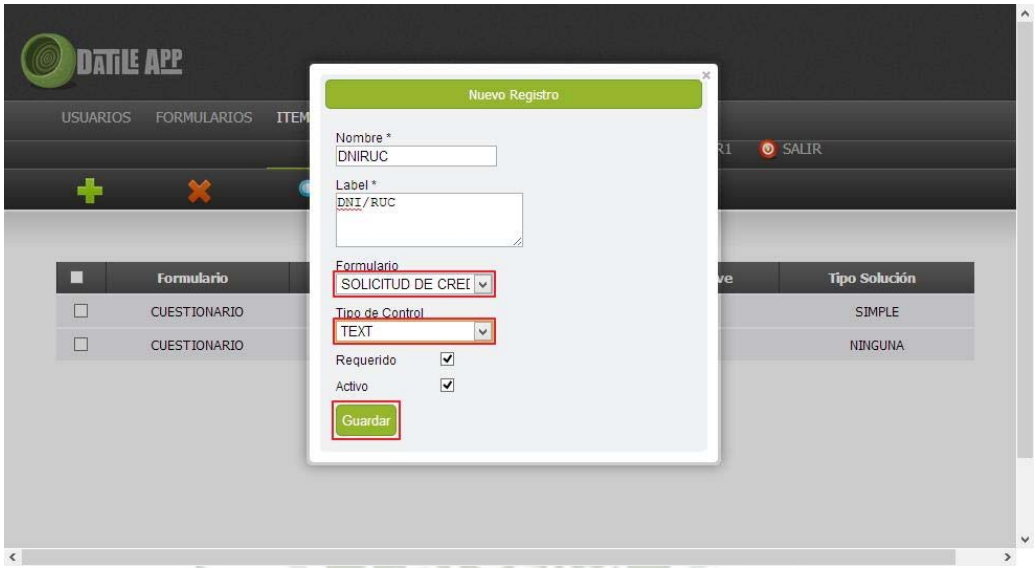

Figura Anexo A.31 - Formulario1: Agregar Ítem 3
Fuente: Elaboración propia

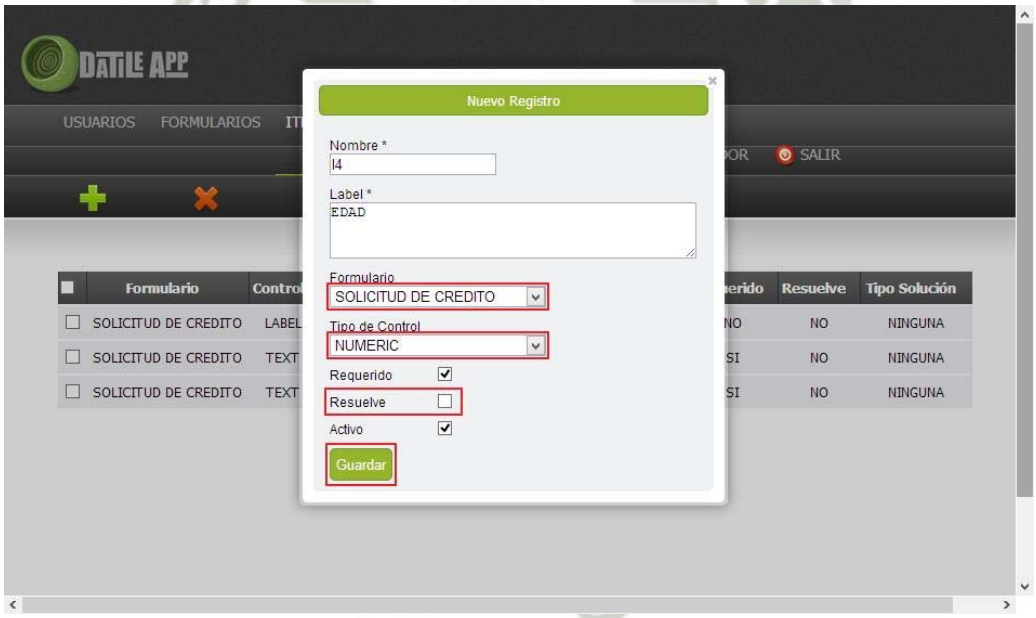

Figura Anexo A.32 - Formulario1: Agregar Ítem 4
Fuente: Elaboración propia

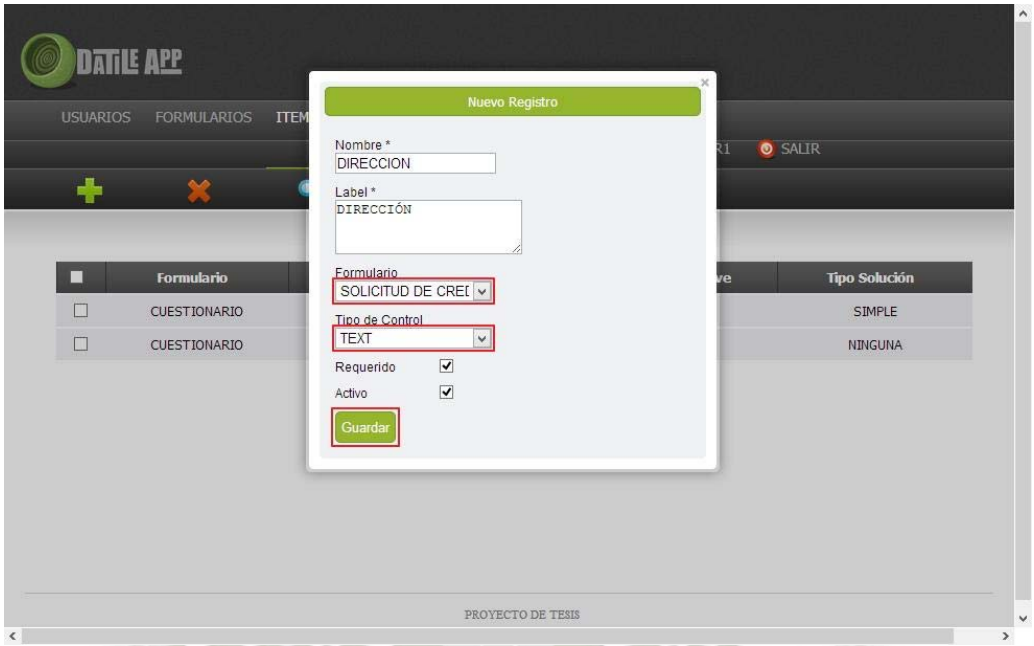

Figura Anexo A.33 - Formulario1: Agregar Ítem 5
Fuente: Elaboración propia

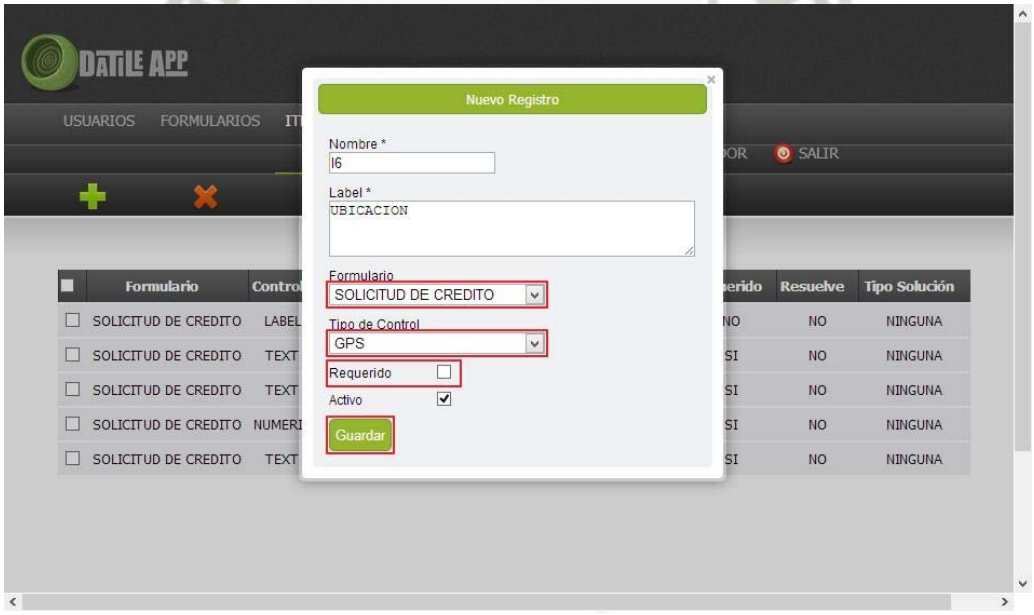

Figura Anexo A.34 - Formulario1: Agregar Ítem 6
Fuente: Elaboración propia

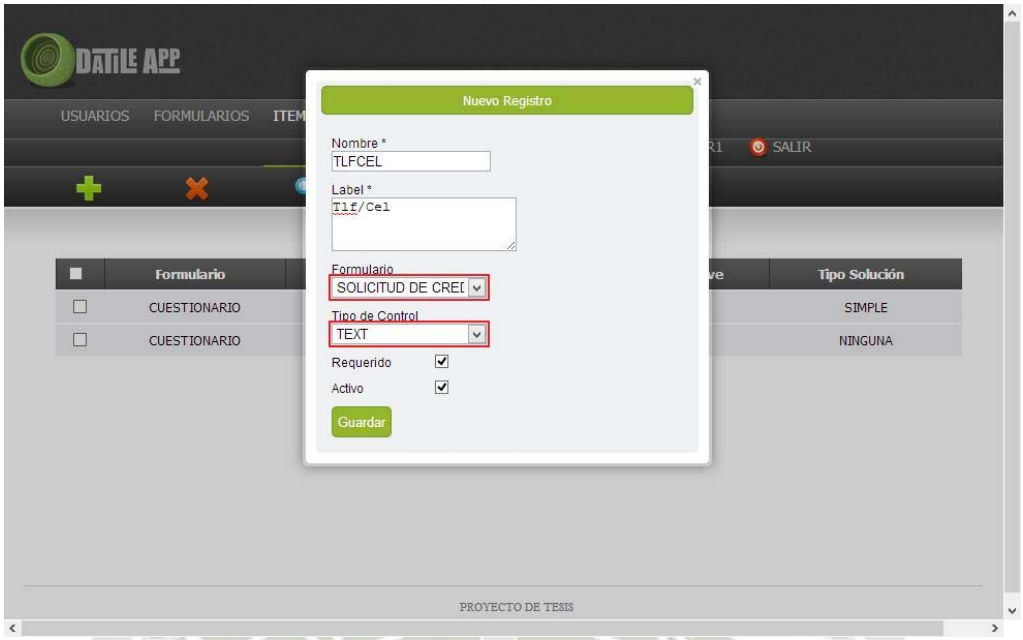

Figura Anexo A.35 - Formulario1: Agregar Ítem 7
Fuente: Elaboración propia

Para el caso de los Items que utilicen el control "Choice" debe crearse una lista de posibles opciones y agregarlas con el botón antes de guardar el Ítem.

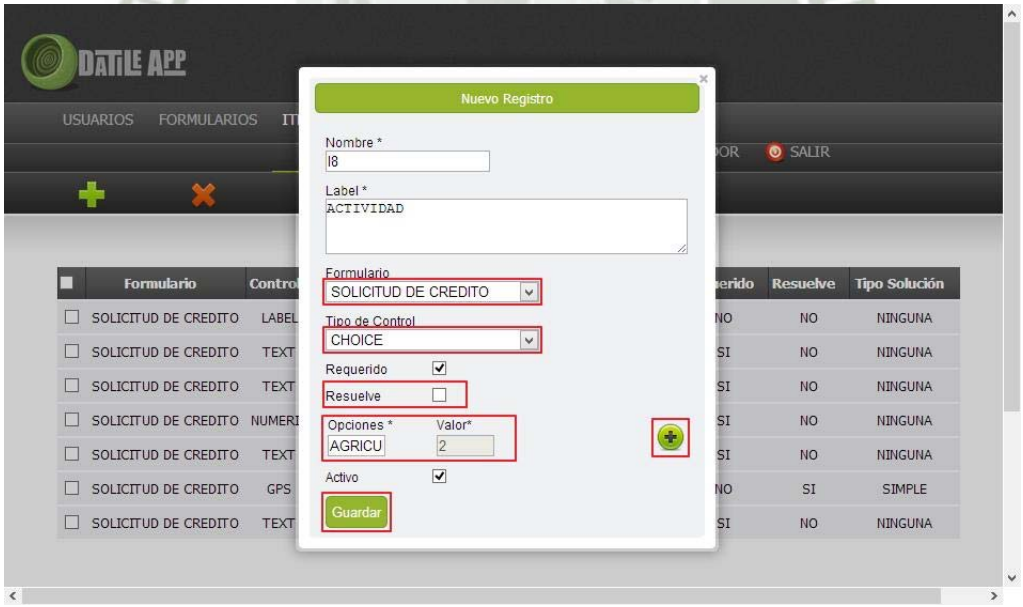

Figura Anexo A.36 - Formulario1: Agregar Ítem 8
Fuente: Elaboración propia

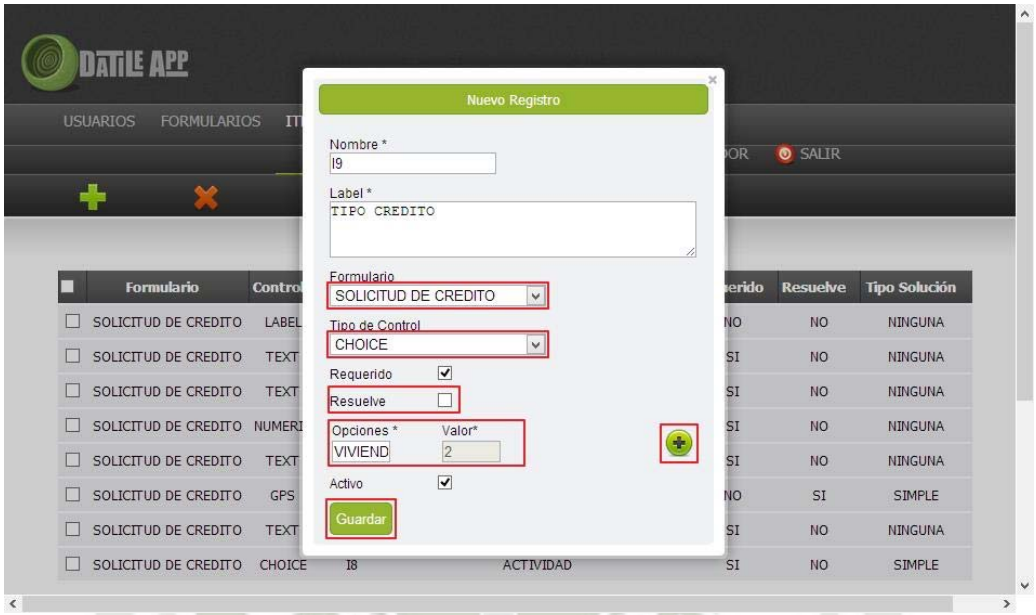

Figura Anexo A.37 - Formulario1: Agregar Ítem 9
Fuente: Elaboración propia

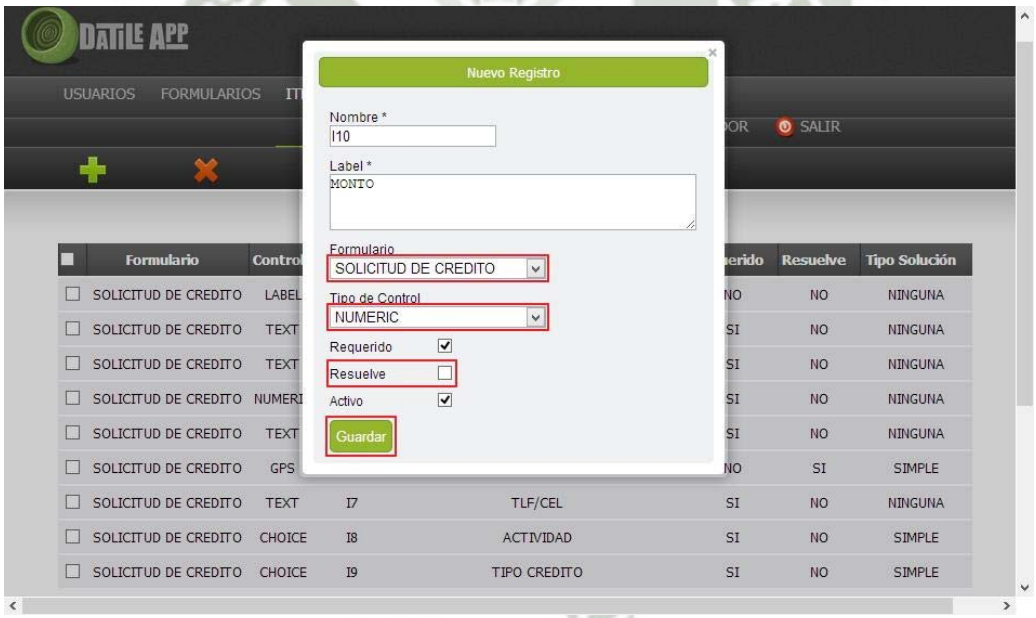

Figura Anexo A.38 - Formulario1: Agregar Ítem 10
Fuente: Elaboración propia

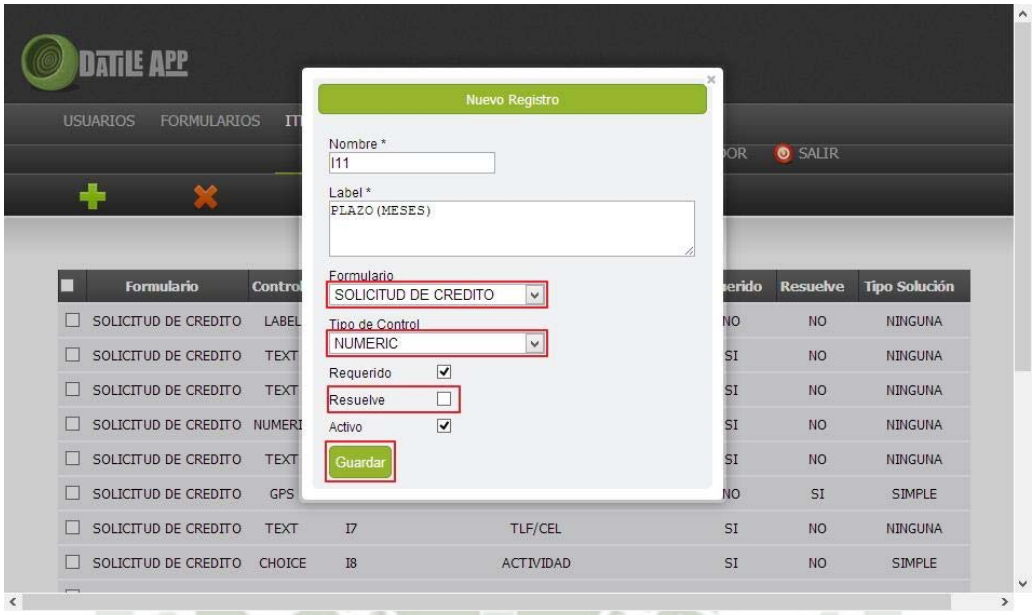

Figura Anexo A.39 - Formulario1: Agregar Ítem 11
Fuente: Elaboración propia

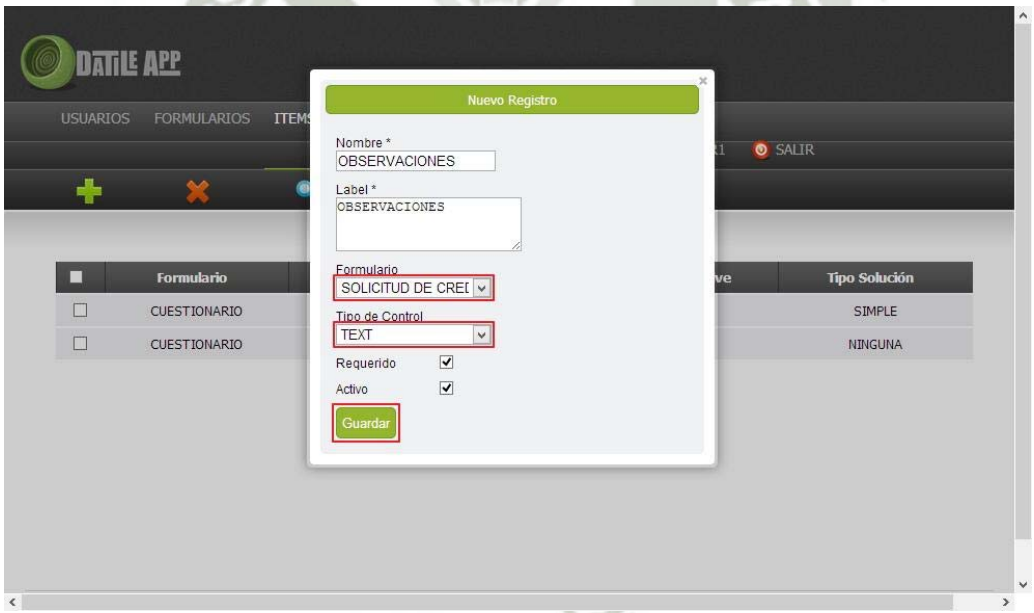

Figura Anexo A.40 - Formulario1: Agregar Ítem 12
Fuente: Elaboración propia

Una vez que se han creado todos los Items, pueden crearse las estrategias de ser necesario si no es así se procede a generar el Script de Inserción para la aplicación móvil.

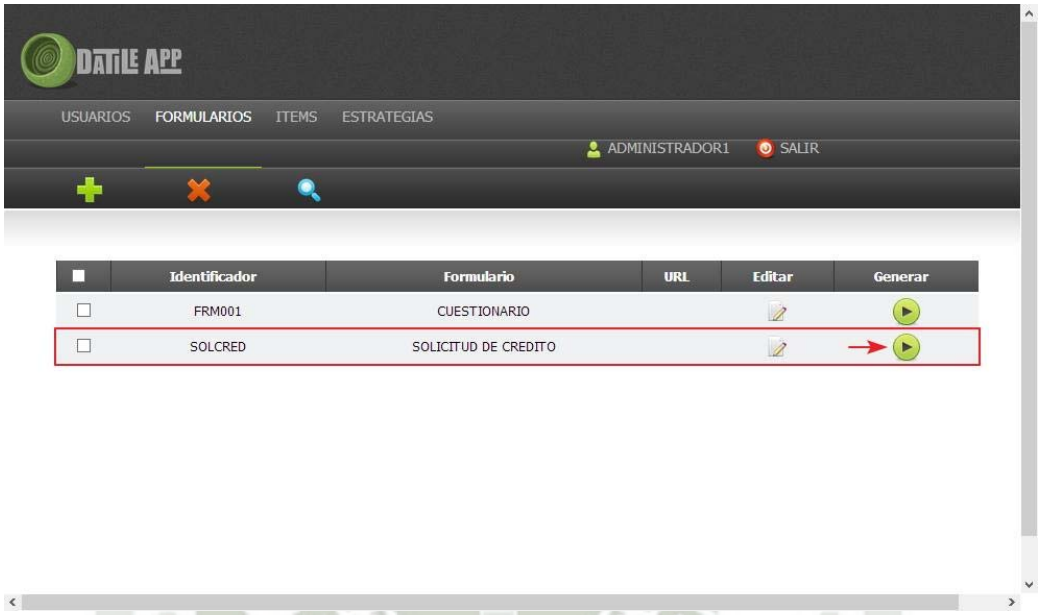

Figura Anexo A.41 - Generación de Formulario 1.
Fuente: Elaboración propia

Una vez que el proceso ha concluido, podemos probar la aplicación móvil, sincronizando el formulario parametrizado.

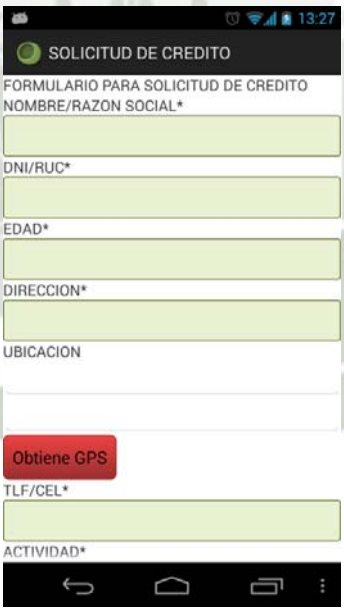

Figura Anexo A.42 - App.Movil: Formulario 1.
Fuente: Elaboración propia

4.3.1. Caso 2: Cuestionario Cultura General Perú.

Tabla Anexo A.30 - Ítems parametrización Caso 2.
Fuente: Elaboración propia.

Nombre	Label	Opciones	Valor	Puntaje	Control	Tip. Sol.	Req.	Res.
INFO	Form. Cultura General Perú	-	-	-	Label	-	-	-
NOMBRE	Nombres y Apellidos	-	-	-	Text	-	Si	-
EDAD	Edad	-	-	-	Numeric	-	No	No
BICENT	Perú celebrará el Bicentenario de su independencia en el año:	-	2021	5	Numeric	Simple	Si	Si
TLC	¿Con que país Perú no tiene TLC?	Singapur, Libia China, USA	-	0, 5, 0, 0	Choice	Simple	Si	Si
HABIT	¿Alrededor de cuántos millones de habitantes tiene Perú? (0-60)	-	0-20, 21-40, 41-60	0, 5, 0	Numeric	Límite	Si	Si
RIOS	Es uno de los ríos más largos del Perú	Urubamba, Amazonas, Chili, Majes	-	5, 5, 0, 0	Choice	Simple	Si	Si
PATR	¿Cuál cree que es el patrimonio cultural más importante de Perú?	Danza de tijeras, Señor de los Milagros, Caballo de paso	-	-	Choice	-	No	No

Tabla Anexo A.31 - Estrategias parametrización Caso 2.
Fuente: Elaboración propia.

Valor	Tipo Solución	Label	Nombre
0-10	Límites	DESAPROBADO	STR1
11-20	Límites	APROBADO	STR2

Para parametrizar este formulario, ingresamos al módulo web del framework.

Figura Anexo A.43 - Formulario 2: Ingreso a módulo web.
Fuente: Elaboración propia

Creamos el nuevo formulario.

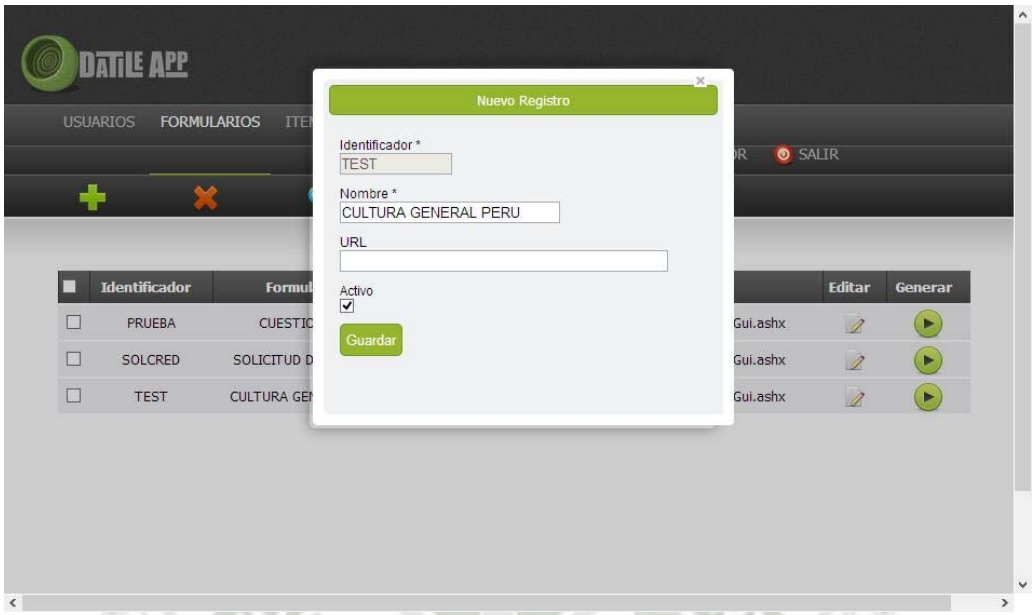

Figura Anexo A.44 - Creación de formulario: Caso 2.
Fuente: Elaboración propia

Creamos todos los items necesarios de la misma forma que en el primer ejemplo.

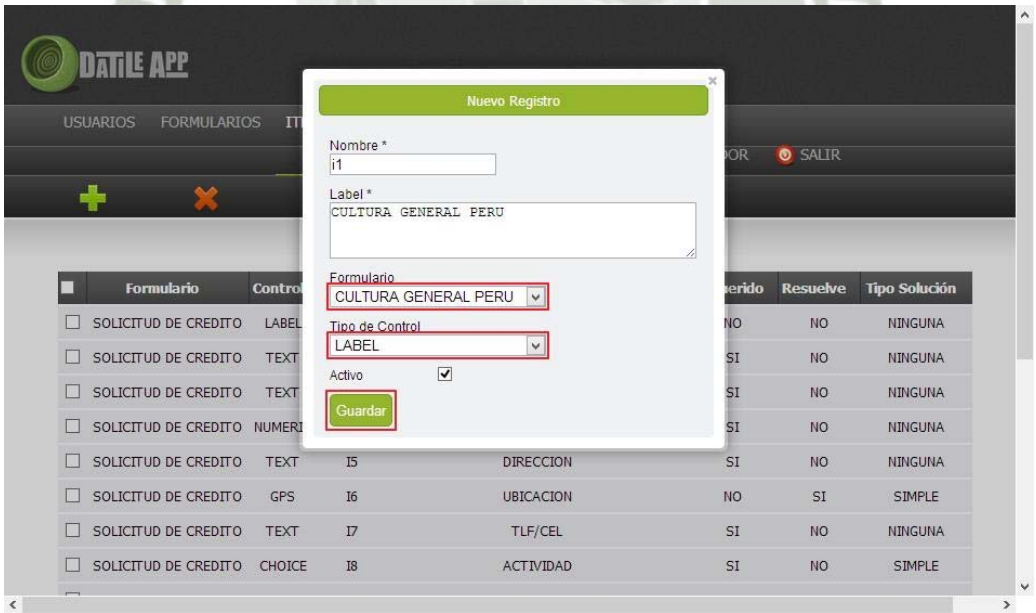

Figura Anexo A.45 - Formulario 2: Agregar Item 1.
Fuente: Elaboración propia

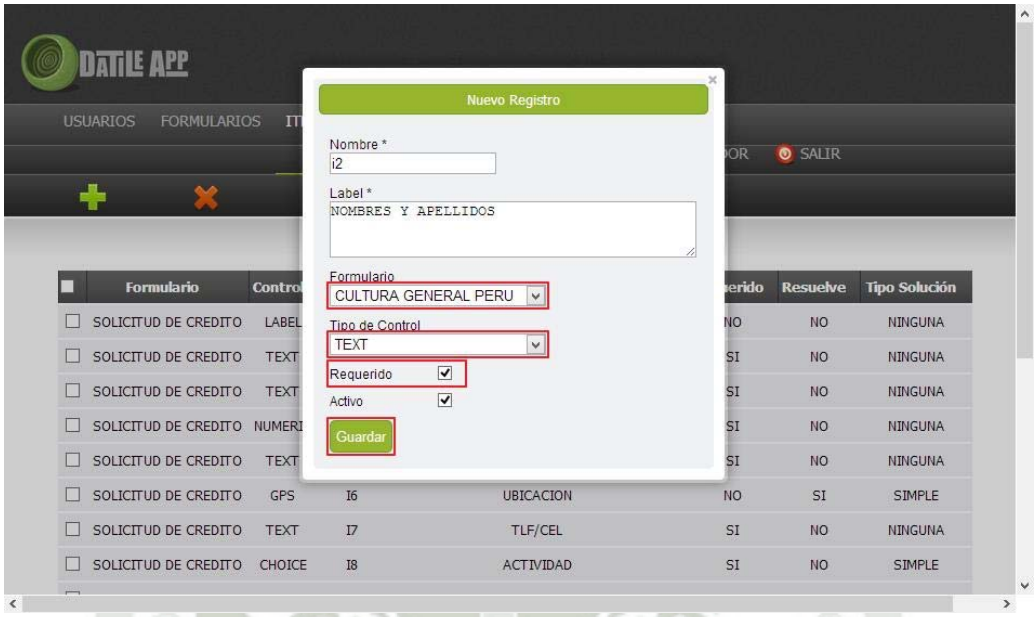

Figura Anexo A.46 - Formulario 2: Agregar Item 2.
Fuente: Elaboración propia

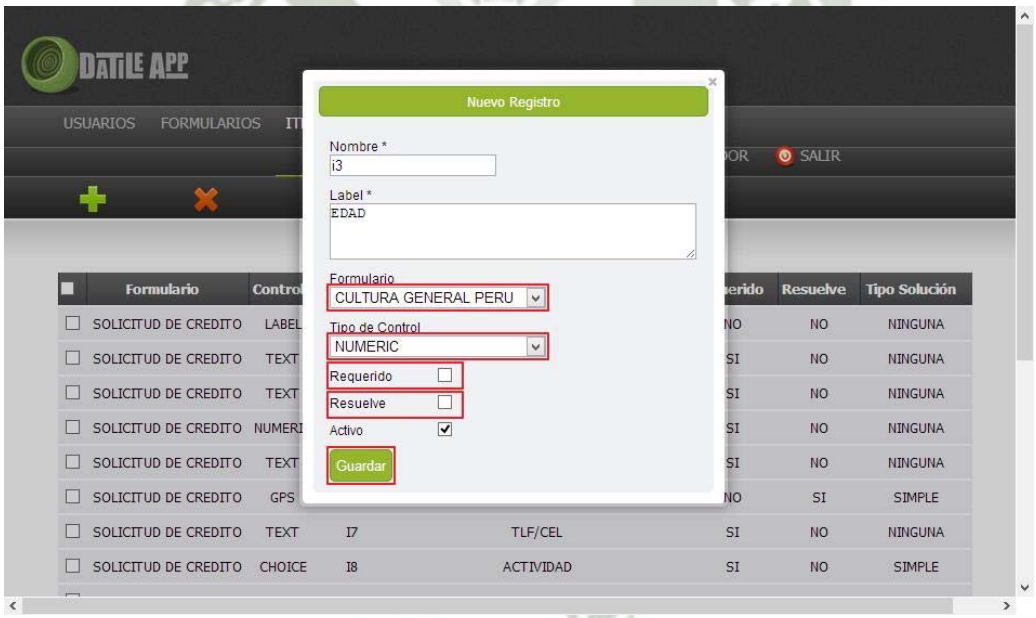

Figura Anexo A.47 - Formulario 2: Agregar Item 3.
Fuente: Elaboración propia

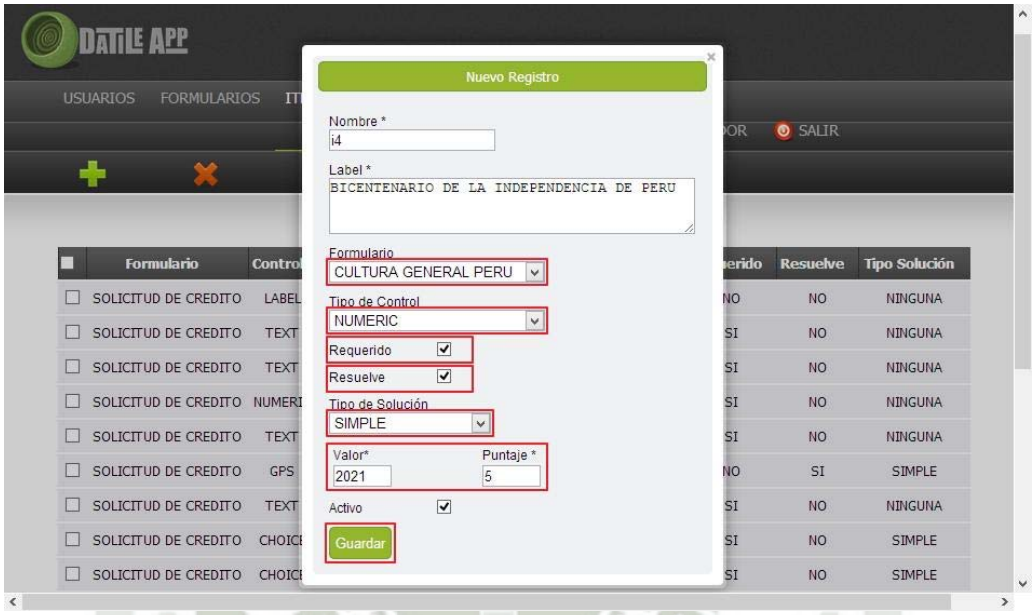

Figura Anexo A.48 - Formulario 2: Agregar Item 4.
Fuente: Elaboración propia

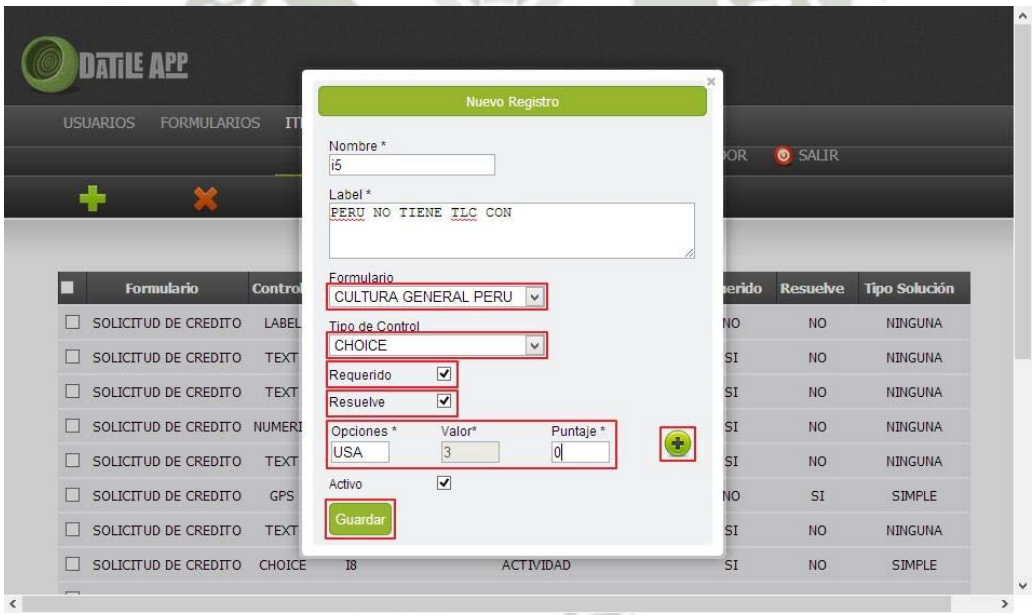

Figura Anexo A.49 - Formulario 2: Agregar Item 5.
Fuente: Elaboración propia

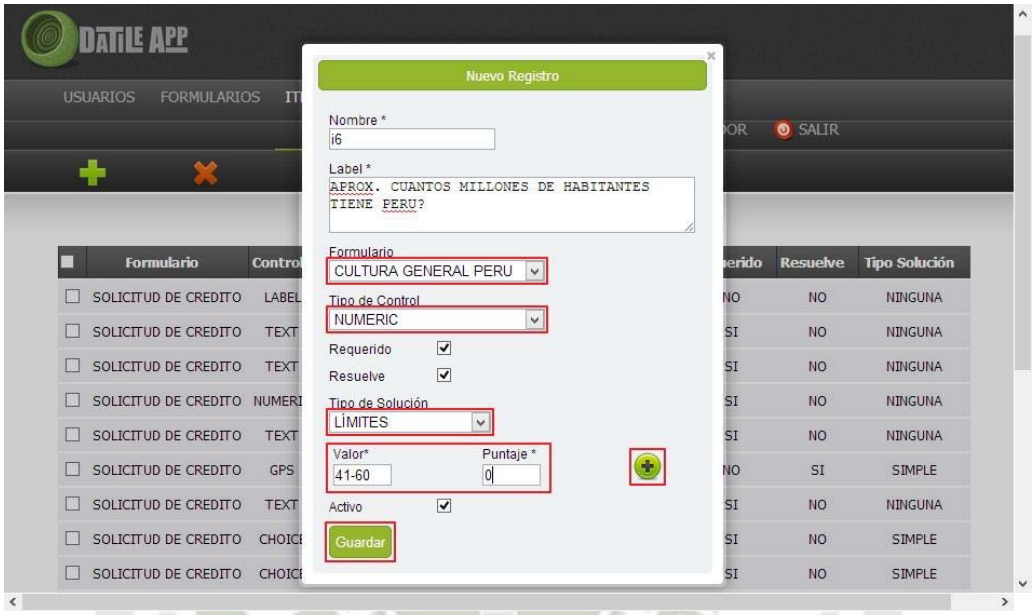

Figura Anexo A.50 - Formulario 2: Agregar Item 6.
Fuente: Elaboración propia

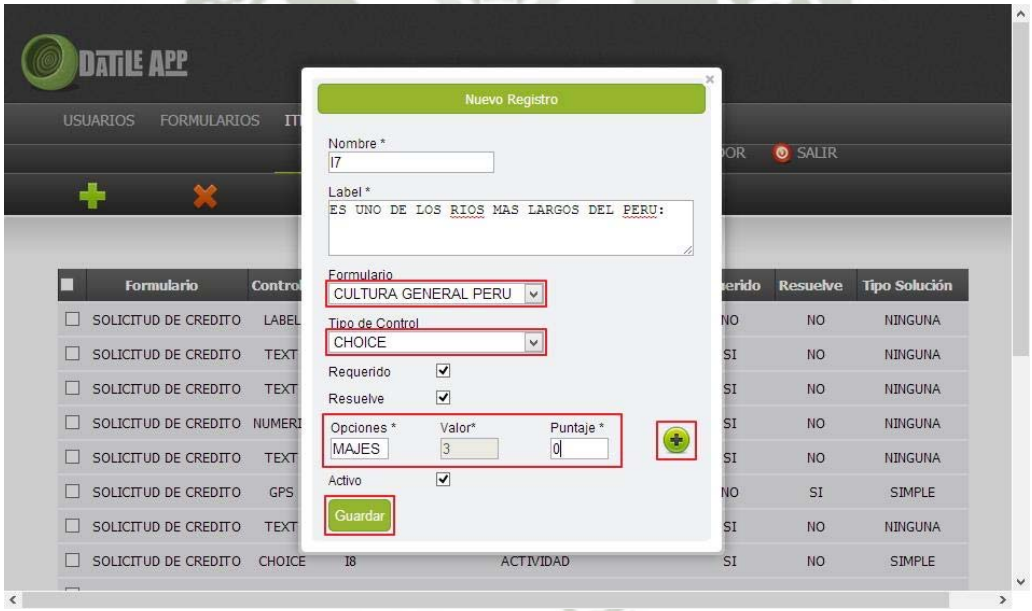

Figura Anexo A.51 - Formulario 2: Agregar Item 7.
Fuente: Elaboración propia

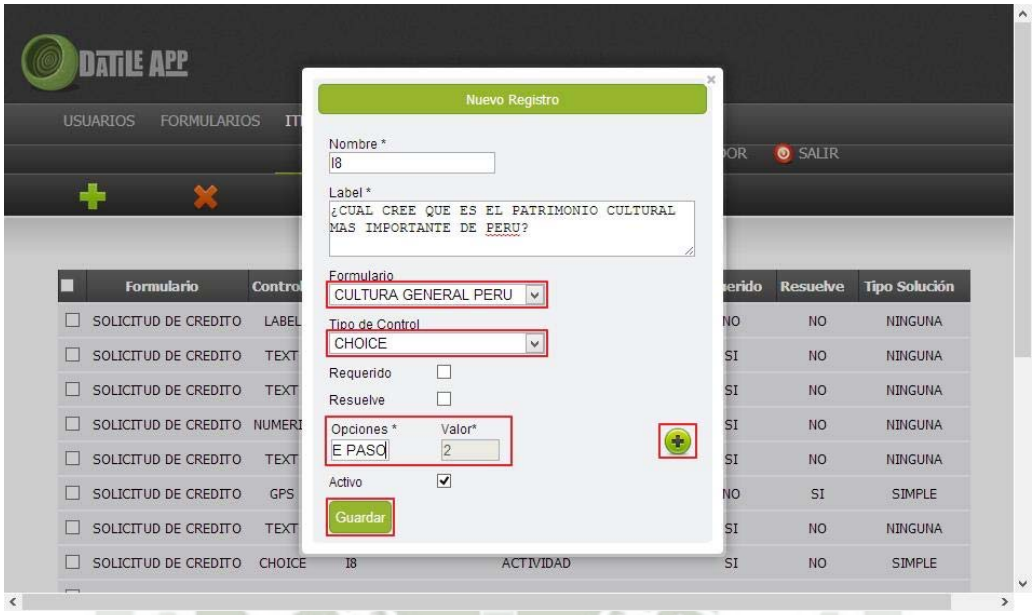

Figura Anexo A.52 - Formulario 2: Agregar Item 8.
Fuente: Elaboración propia

Luego de tener agregados los ítems, se procede a crear las estrategias para este formulario, ya que son necesarias para que pueda ser resuelto.

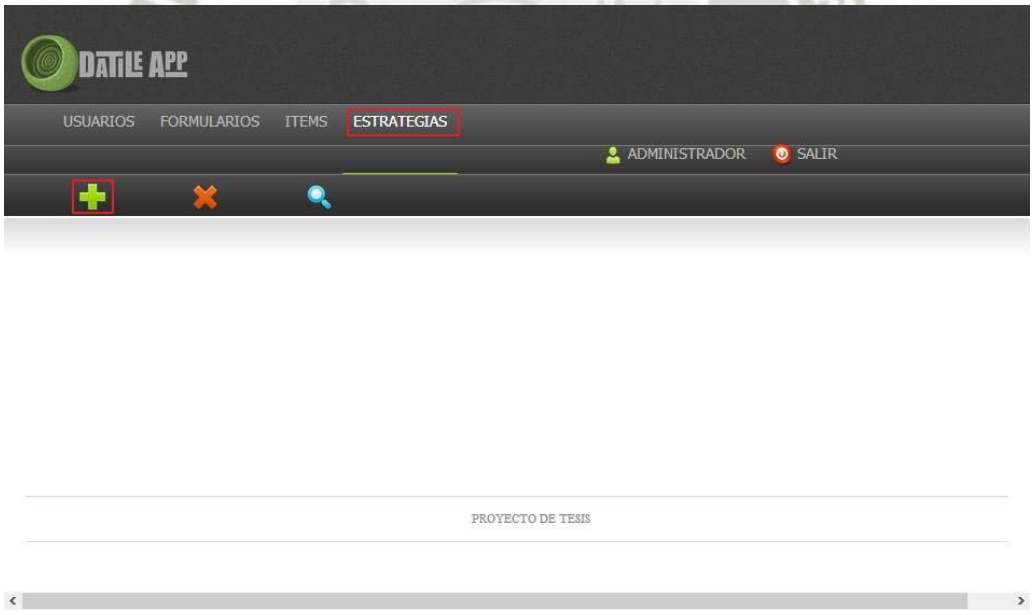

Figura Anexo A.53 - Formulario 2: Agregar Estrategias.
Fuente: Elaboración propia

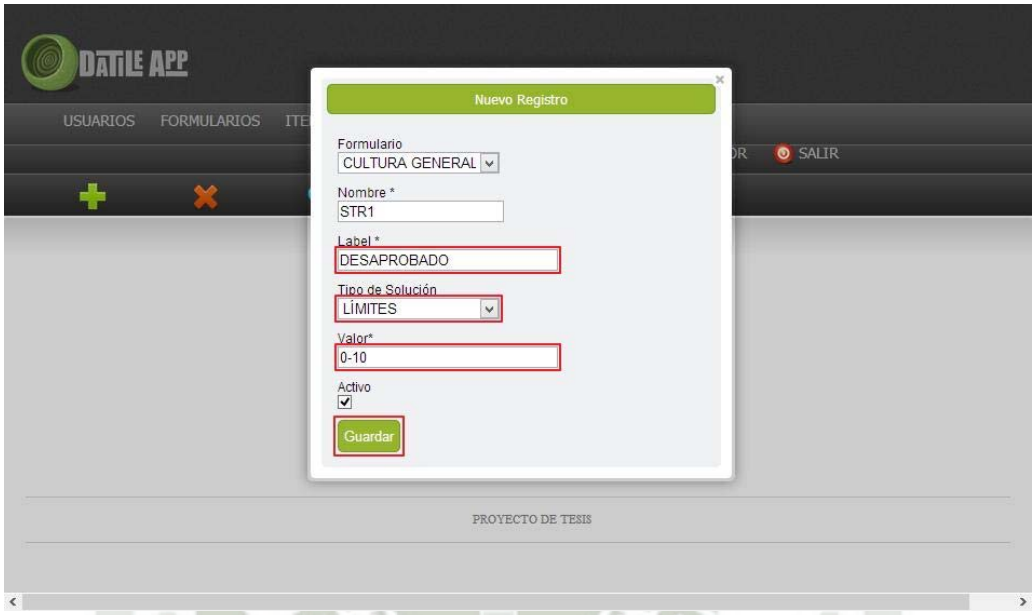

Figura Anexo A.54 - Formulario 2: Agregar Estrategia 1.
Fuente: Elaboración propia

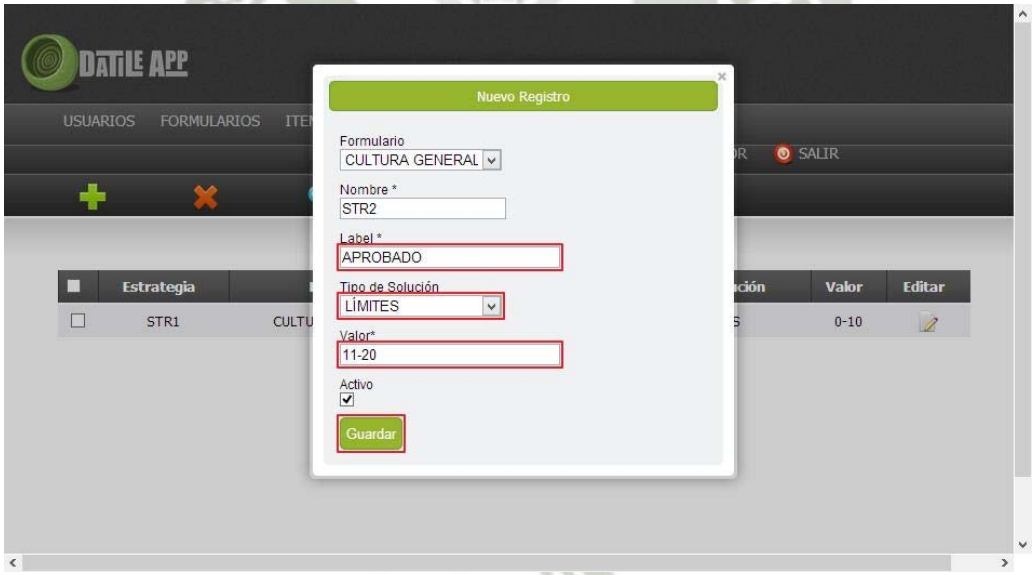

Figura Anexo A.55 - Formulario 2: Agregar Estrategia 2.
Fuente: Elaboración propia

Se genera el Script de Inserción para la aplicación móvil.

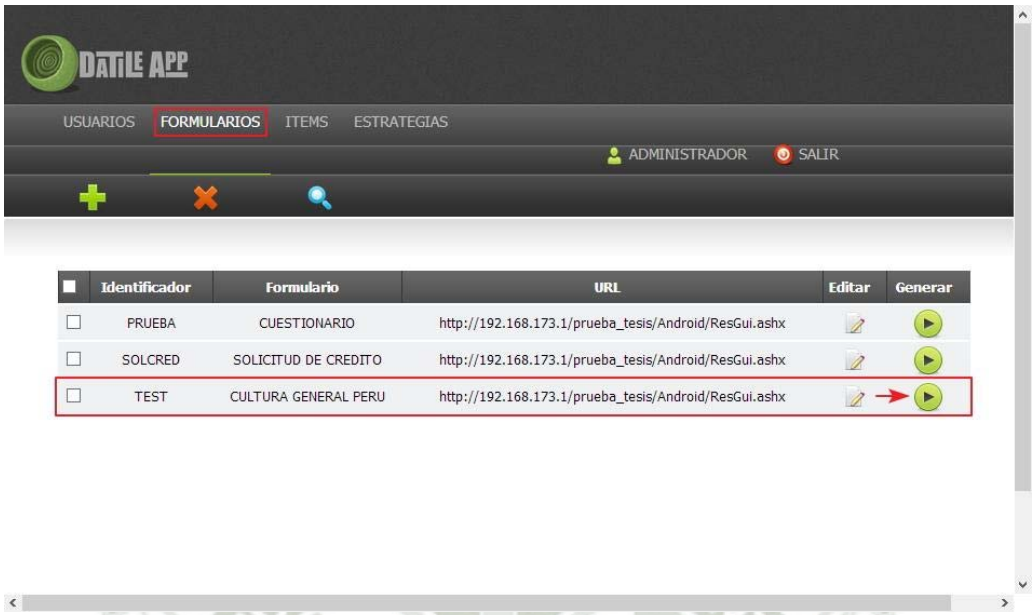

Figura Anexo A.56 - Generación de Formulario 2.
Fuente: Elaboración propia

Finalmente podemos probar el funcionamiento de la aplicación móvil para el Formulario 2.

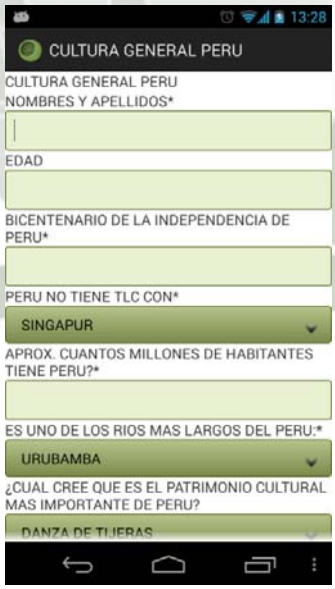

Figura Anexo A.57 - App. Movil: Formulario 2.
Fuente: Elaboración propia

ANEXO B

CÓDIGO FUENTE DEL FRAMEWORK

1. MÓDULO WEB.

- **Carpeta principal del Proyecto**
  - **Default.aspx:**
    - Función SubLogin: Valida el ingreso de usuarios.

```
private void subLogin()
{
      DataSet data =
      DBNet.validarUsuarioWeb(txtUsuario.Value.Trim(),
      txtPassword.Value.Trim());

       string id_usuario = "-1";
       String nombre = "";
       String perfil = "";

         foreach (DataRow drow in data.Tables[0].Rows)
         {
            id_usuario =
            drow["USRCOD"].ToString().Trim().ToUpper();
             nombre = drow["USRDSC"].ToString().Trim().ToUpper();
             perfil =
drow["USRPERFIL"].ToString().Trim().ToUpper();
        }

        if (id_usuario == "-1")
        {
            mensaje.Visible = true;
            txtUsuario.Value = String.Empty;
            txtPassword.Value = String.Empty;
        }
        else
           {
           FormsAuthentication.RedirectFromLoginPage("Admin",
           false);
            Session["Usuario"] = nombre.ToUpper().Trim();
            Response.Redirect("Mantenimiento/Usuario.aspx");
        }
```

```
}
```

- **Logout.aspx:**

  - Función Page_Load: Cierra las sesión abierta, limpia variables de sesión.

```
protected void Page_Load(object sender, EventArgs e)
{
      Session.Abandon();
      Response.Redirect("~/Default.aspx");
}
```

- **Carpeta Android.**

  - **Login.ashx:** Encargada de permitir o denegar el acceso a la aplicación móvil.

    - Función ProcessRequest: Procesa la solicitud de validación del dispositivo móvil, interpreta la cadena Json y serializa otra para enviar la respuesta.

```
public void ProcessRequest (HttpContext context) {
        // Recepcion de Datos
        String lsJson;
        byte[] loinputBinary;
        int liBinaryBytes;

        liBinaryBytes = context.Request.TotalBytes;
           loinputBinary =
           context.Request.BinaryRead(liBinaryBytes);
        lsJson = Utils.fnConvertirBinaryToString(loinputBinary);
       CUsuario Usuario =
       Newtonsoft.Json.JsonConvert.DeserializeObject<CUsuario>(
       lsJson);
        Usuario = ControlAndroid.autenticarUsuario(Usuario);
     context.Response.Write(Newtonsoft.Json.JsonConvert.Serializ
     eObject(Usuario));
 }
```

  - **ResGui.ashx:** Recibe los resultados enviados por el dispositivo móvil.

    - Función ProcessRequest: Procesa los datos enviados por el dispositivo móvil, y los escribe en un archivo txt.

```
public void ProcessRequest (HttpContext context) {
      byte[] loinputBinary;
      int liBinaryBytes;
      liBinaryBytes = context.Request.TotalBytes;
      loinputBinary = context.Request.BinaryRead(liBinaryBytes);
      string str =
    Utils.fnConvertirBinaryToString(loinputBinary);

      string[] array = str.Split(new char[] { ':' }, 2);
      string appPath = context.Request.PhysicalApplicationPath;
      string titulo = array[0].Trim();
      string filePath = appPath + "Android/"+titulo+".txt";

      System.IO.StreamWriter sw = new
      System.IO.StreamWriter(filePath);
      sw.WriteLine(array[1]);
      sw.Close();

      HttpResponse Response = HttpContext.Current.Response;
      context.Response.Write("SUCCESS");
      if (System.IO.File.Exists(filePath))
      context.Response.Write(" (EXISTS) ");
      else
      context.Response.Write(" (NOHAY) ");
      context.Response.Write(" : " + filePath);
}
```

- **Carpeta AppCode/Android**

  - **Clase CUsuario.cs:** Clase Usuario para dispositivos móviles.

    - Atributos de la clase CUsuario:

```
public String codigo;
public String clave;
public String nombre;
public int idResultado;
public String resultado;
```

  - **ControlAndroid.cs:** Clase que funciona como nexo entre el acceso a datos y las solicitudes de los dispositivos móviles.

    - Función autenticarUsuario: Envía los datos recibidos desde el dispositivo móvil hasta las funciones de acceso a datos y retorna la respuesta.

```
public static CUsuario autenticarUsuario(CUsuario Usuario)
{
      CUsuario loBEGRUsuario = new CUsuario();
      try
      {
            DataSet data = DBAndroid.autenticarUsuario(Usuario);

            foreach (DataRow drow in data.Tables[0].Rows)
            {
                  loBEGRUsuario.idResultado =
                  Int32.Parse(drow["RPTA"].ToString().Trim());
                  switch (loBEGRUsuario.idResultado)
                  {
                  case 1:
                        loBEGRUsuario.nombre =
                        (String)drow["USRDSC"].ToString().Trim();
                        ;
                        loBEGRUsuario.codigo = Usuario.codigo;
                        loBEGRUsuario.clave = Usuario.clave;
                              loBEGRUsuario.resultado = "Bienvenido
                        " + loBEGRUsuario.nombre;
                        break;
                  case -1:
                        loBEGRUsuario.resultado = "Usuario
                        incorrecto";
                        break;
                  case 0:
                        loBEGRUsuario.resultado = "Password
                        incorrecto";
                        break;
                  default:
                        break;
                  }
            }
      }
      catch (Exception ex)
      {
            loBEGRUsuario.resultado = ex.ToString();
            loBEGRUsuario.idResultado = -1;
      }
      return loBEGRUsuario;
}
```

- **DbAndroid.cs:** Contiene funciones de acceso a datos necesarias para la interacción de los dispositivos móviles con los datos almacenados.

```
public static DataSet autenticarUsuario(CUsuario pUser)
{
        ArrayList alParameters = new ArrayList();
        SqlParameter parameter;
```

```
            parameter = new SqlParameter("@PAR_USRCOD",
        SqlDbType.VarChar, 20);
            parameter.Value = pUser.codigo;
            alParameters.Add(parameter);

            parameter = new SqlParameter("@PAR_USRPWD",
        SqlDbType.VarChar, 100);
            parameter.Value = pUser.clave;
            alParameters.Add(parameter);

            return DBUtilSQLServer.getDataset("SPS_LOGIN",
        alParameters);
}
```

- **Carpeta AppCode/Net**

  - **Clase CUsuario.cs:** Clase usuario para módulo web.

    - Atributos de la clase CUsuario:

```
public int id;
public String codigo;
public String clave;
public String nombre;
public String perfil;
public String stat;
```

  - **Clase CFormato.cs:** Clase formato para módulo web.

    - Atributos de la clase CFormato:

```
public int id;
public String idt;
public String name;
public String url;
public String stat;
```

  - **Clase CItem.cs:**

    - Atributos de la clase CItem:

```
public int id;
public String idForm;
public String idControl;
public String name;
```

```
public String label;
public String requerido;
public String resuelve;
public String idTipoSol;
public String options;
public String values;
public String points;
public String stat;
```

- **Clase CEstrategia.cs:**

  - Atributos de la clase CEstrategia:

```
public int id;
public String form;
public String name;
public String label;
public String tsol;
public String valor;
public String stat;
```

- **DbNet.cs:** Contiene funciones para acceso a datos desde el módulo web.

  - Función validarUsuarioWeb: Valida los datos ingresados por el usuario.

```
public static DataSet validarUsuarioWeb(string parUser, string
pasPass)
{
          ArrayList alParameters = new ArrayList();
          SqlParameter parameter;

          parameter = new SqlParameter("@IN_USER",
          SqlDbType.VarChar, 20);
          parameter.Value = parUser;
          alParameters.Add(parameter);

          parameter = new SqlParameter("@IN_PASS",
          SqlDbType.VarChar, 100);
          parameter.Value = pasPass;
          alParameters.Add(parameter);

          return DBUtilSQLServer.getDataset("SPS_LoginWeb",
          alParameters);
}
```

- Función getPerfiles: Lista todos los perfiles.

```
public static DataSet getPerfiles()
{
          ArrayList alParameters = new ArrayList();
           DataSet dst =
          DBUtilSQLServer.getDataset("SPS_MTProfileSel",
          alParameters);
           return dst;
}
```

- Función getUsuariobyId: Recupera los datos de un usuario basándose en el id del mismo.

```
public static CUsuario getUsuariobyId(int idUser)
{
          CUsuario usuario = null;
          SqlConnection conn = null;
          SqlCommand cmd = null;
          SqlDataReader loDr = null;
          Int32 contador = 1;
          try
          {
              conn = new SqlConnection(prCadConexion);
              conn.Open();
              cmd = new SqlCommand("SPS_MTUserSelObj", conn);
              cmd.CommandTimeout = 0;
              cmd.CommandType = CommandType.StoredProcedure;
              cmd.Parameters.Add(new SqlParameter("@USR_ID",
              idUser));
              loDr = cmd.ExecuteReader();
              while (loDr.Read())
              {
                  usuario = new CUsuario();
                   usuario.id =
                   Int32.Parse(loDr.GetValue(loDr.GetOrdinal(
                   "USRID")).ToString().Trim());
                  usuario.codigo =
                  loDr.GetValue(loDr.GetOrdinal("USRCOD")).ToS
                  tring().Trim();
                  usuario.nombre =
                  loDr.GetValue(loDr.GetOrdinal("USRNAME")).To
                  String().Trim();
```

```
                    usuario.clave =
                    loDr.GetValue(loDr.GetOrdinal("USRPWD")).ToS
                    tring().Trim();
                    usuario.perfil =
                    loDr.GetValue(loDr.GetOrdinal("USRPROF")).To
                    String().Trim();
                    usuario.stat =
                    loDr.GetValue(loDr.GetOrdinal("USRSTAT")).To
                    String().Trim();
                }
            }
            catch (Exception e)
            {
                throw e;
            }
            finally
            {
                #region Liberando Recursos
                loDr.Close();
                conn.Close();
                #endregion
            }
            return usuario;
}
```

- Función getUsuarios: Lista usuarios según los parámetros enviados.

```
public static DataTable getUsuarios(string usr_codigo, string
usr_nombre, string usr_perfil, string usr_stat)
{
            ArrayList lvArrayParameter = new ArrayList();

            SqlParameter lvParametro = new
            SqlParameter("@USR_COD", SqlDbType.VarChar, 20);
            lvParametro.Value = fnNull(usr_codigo).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@USR_NAME",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(usr_nombre).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@USR_PROF",
            SqlDbType.VarChar, 20);
            lvParametro.Value = fnNull(usr_perfil).Trim();
```

```
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@USR_STAT",
            SqlDbType.VarChar, 20);
            lvParametro.Value = fnNull(usr_stat).Trim();
            lvArrayParameter.Add(lvParametro);
            DataSet dst =
            DBUtilSQLServer.getDataset("SPS_MTUserSel",
            lvArrayParameter);

            if (dst != null && dst.Tables.Count > 0)
            {
                return dst.Tables[0];
            }
            else
                return new DataTable();
}
```

- Función guardarUsuario: Inserta los datos de un usuario.

```
public static void guardarUsuario(CUsuario usuario)
{
            using (SqlConnection conn = new
            SqlConnection(prCadConexion))
            {
                using (SqlCommand cmd = new
                SqlCommand("SPS_MTUserInsUpd", conn))
                {
                    conn.Open();
                    cmd.CommandTimeout = 0;
                     cmd.CommandType =
                     CommandType.StoredProcedure;
                     cmd.Parameters.Add(new
                     SqlParameter("@USR_ID", usuario.id));
                    cmd.Parameters.Add(new
                    SqlParameter("@USR_COD", usuario.codigo));
                    cmd.Parameters.Add(new
                    SqlParameter("@USR_NAME", usuario.nombre));
                    cmd.Parameters.Add(new
                    SqlParameter("@USR_PWD", usuario.clave));
                    cmd.Parameters.Add(new
                    SqlParameter("@USR_PROF", usuario.perfil));
                    cmd.Parameters.Add(new
                    SqlParameter("@USR_STAT", usuario.stat));
                    cmd.ExecuteNonQuery();
                }
```

```
            }
}
```

- Función deleteUsuarios: Inactiva los usuarios en base de sus identificadores.

```
public static void deleteUsuarios(string ids)
{
            ArrayList lvArrayParameter = new ArrayList();
            SqlParameter lvParametro = new SqlParameter("@IDS",
            SqlDbType.VarChar, 200);
            lvParametro.Value = ids;
            lvArrayParameter.Add(lvParametro);
            try
            {
                DBUtilSQLServer.executeNonQuery("SPS_MTUserDel",
                lvArrayParameter);
            }
            catch (Exception e)
            {
                lbResultado = false;
            }
 }
```

- Función guardarFormato: Inserta los datos de un formulario.

```
public static void guardarFormato(CFormato formato)
 {
              using (SqlConnection conn = new
           SqlConnection(prCadConexion))
            {
              using (SqlCommand cmd = new
           SqlCommand("SPS_MTFormInsUpd", conn))
                {
                    conn.Open();
                    cmd.CommandTimeout = 0;
                    cmd.CommandType =
CommandType.StoredProcedure;
                     cmd.Parameters.Add(new
                     SqlParameter("@FRM_ID",
                     formato.id.ToString()));
                    cmd.Parameters.Add(new
                    SqlParameter("@FRM_IDT", formato.idt));
                    cmd.Parameters.Add(new
                    SqlParameter("@FRM_NOMBRE", formato.name));
                    cmd.Parameters.Add(new
                    SqlParameter("@FRM_URL", formato.url));
```

```
                    cmd.Parameters.Add(new
                    SqlParameter("@FRM_STAT", formato.stat));

                    cmd.ExecuteNonQuery();
                }
            }
}
```

- Función getFormatos: Lista formularios según los parámetros enviados.

```
public static DataTable getFormatos(string frm_idt, string
frm_name, string frm_url, string frm_stat)
{
            ArrayList lvArrayParameter = new ArrayList();

            SqlParameter lvParametro = new
            SqlParameter("@FRM_IDT", SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(frm_idt).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@FRM_NOMBRE",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(frm_name).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@FRM_URL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(frm_url).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@FRM_STAT",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(frm_stat).Trim();
            lvArrayParameter.Add(lvParametro);

            DataSet dst =
            DBUtilSQLServer.getDataset("SPS_MTFormSel",
            lvArrayParameter);

            if (dst != null && dst.Tables.Count > 0)
            {
                return dst.Tables[0];
            }
            else
                return new DataTable();
```

```
}
```

- Función getFormatobyId: Recupera los datos de un formulario basándose en el id del mismo.

```
public static CFormato getFormatobyId(int idFormato)
{
            CFormato formato = null;
            SqlConnection conn = null;
            SqlCommand cmd = null;
            SqlDataReader loDr = null;
            Int32 contador = 1;

            try
            {
                conn = new SqlConnection(prCadConexion);
                conn.Open();

                cmd = new SqlCommand("SPS_MTFormSelObj", conn);

                cmd.CommandTimeout = 0;
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add(new SqlParameter("@FRM_ID",
                idFormato));

                loDr = cmd.ExecuteReader();

                while (loDr.Read())
                {
                    formato = new CFormato();

                    formato.id =
                    Int32.Parse(loDr.GetValue(loDr.GetOrdinal("F
                    RMID")).ToString().Trim());
                    formato.idt =
                    loDr.GetValue(loDr.GetOrdinal("FRMIDT")).ToS
                    tring().Trim();
                    formato.name =
                    loDr.GetValue(loDr.GetOrdinal("FRMNAME")).To
                    String().Trim();
                    formato.url =
                    loDr.GetValue(loDr.GetOrdinal("FRMURL")).ToS
                    tring().Trim();
                    formato.stat =
                    loDr.GetValue(loDr.GetOrdinal("FRMSTAT")).To
                    String().Trim();
```

```
            }
        }
        catch (Exception e)
        {
            throw e;
        }
        finally
        {
            #region Liberando Recursos
            loDr.Close();
            conn.Close();
            #endregion
        }
        return formato;
}
```

- Función deleteFormatos: Inactiva los formularios en base de sus identificadores.

```
public static void deleteFormatos(string ids)
{
        ArrayList lvArrayParameter = new ArrayList();
        Boolean lbResultado = true;
        SqlParameter lvParametro = new SqlParameter("@IDS",
        SqlDbType.VarChar, 200);
        lvParametro.Value = ids;
        lvArrayParameter.Add(lvParametro);
        try
           {
           DBUtilSQLServer.executeNonQuery("SPS_MTFormDel",
           lvArrayParameter);
        }
        catch (Exception e)
        {
            lbResultado = false;
        }
}
```

- Función getLstFormatos: Lista todos los formularios disponibles.

```
public static DataSet getLstFormatos()
{
        ArrayList alParameters = new ArrayList();
        DataSet dst =
       DBUtilSQLServer.getDataset("SPS_MTFormLstSel",
       alParameters);
        return dst;
}
```

- Función getLstControles: Lista todos los controles disponibles.

```
public static DataSet getLstControles()
{
            ArrayList alParameters = new ArrayList();
       DataSet dst =
     DBUtilSQLServer.getDataset("SPS_MTCtrlTypeLstSel",
     alParameters);
            return dst;
}
```

- Función getLstTSolucion: Lista todos los tipos de solución disponibles.

```
public static DataSet getLstTSolucion()
{
            ArrayList alParameters = new ArrayList();
                DataSet dst =
                DBUtilSQLServer.getDataset("SPS_MTTypeSolLstSel"
                , alParameters);
            return dst;
}
```

- Función getItems: Lista items según los parámetros enviados.

```
public static DataTable getItems(string itm_name, string
itm_label, string itm_form, string itm_ctrl,
            string itm_tsol, string itm_req, string itm_solve,
string itm_stat)
{
            ArrayList lvArrayParameter = new ArrayList();

            SqlParameter lvParametro = new
            SqlParameter("@ITM_NAME", SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_name).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_LABEL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_label).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_FORM",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_form).Trim();
```

```
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_CTRL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_ctrl).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_TSOL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_tsol).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_REQ",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_req).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_SOLVE",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_solve).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@ITM_STAT",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(itm_stat).Trim();
            lvArrayParameter.Add(lvParametro);

            DataSet dst =
            DBUtilSQLServer.getDataset("SPS_MTItemSel",
            lvArrayParameter);

            if (dst != null && dst.Tables.Count > 0)
            {
                return dst.Tables[0];
            }
            else
                return new DataTable();
}
```

- Función getItembyId: Recupera los datos de un item basándose en el id del mismo.

```
public static CItem getItembyId(int idItem)
{
            CItem item = null;
            SqlConnection conn = null;
```

```
SqlCommand cmd = null;
SqlDataReader loDr = null;
Int32 contador = 1;

try
{
    conn = new SqlConnection(prCadConexion);
    conn.Open();

    cmd = new SqlCommand("SPS_MTItemSelObj", conn);

    cmd.CommandTimeout = 0;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(new SqlParameter("@ITM_ID",
    idItem));

    loDr = cmd.ExecuteReader();
    while (loDr.Read())
    {
        item = new CItem();

        item.id =
        Int32.Parse(loDr.GetValue(loDr.GetOrdinal("I
        TMID")).ToString().Trim());
        item.idForm =
        loDr.GetValue(loDr.GetOrdinal("FRMID")).ToSt
        ring().Trim();
        item.idControl =
        loDr.GetValue(loDr.GetOrdinal("CTRLID")).ToS
        tring().Trim();
        item.name =
        loDr.GetValue(loDr.GetOrdinal("ITMNAME")).To
        String().Trim();
        item.label =
        loDr.GetValue(loDr.GetOrdinal("ITMLABEL")).T
        oString().Trim();
        item.requerido =
        loDr.GetValue(loDr.GetOrdinal("ITMREQUI")).T
        oString().Trim();
        item.resuelve =
        loDr.GetValue(loDr.GetOrdinal("ITMSOLVE")).T
        oString().Trim();
        item.idTipoSol =
        loDr.GetValue(loDr.GetOrdinal("TSOLID")).ToS
        tring().Trim();
```

```
                    item.stat =
                    loDr.GetValue(loDr.GetOrdinal("ITMSTAT")).To
                    String().Trim();
                }
            }
            catch (Exception e)
            {
                throw e;
            }
            finally
            {
                #region Liberando Recursos
                loDr.Close();
                conn.Close();
                #endregion
            }
            return item;
        }
```

- Función guardarItem: Inserta los datos de un item.

```
public static void guardarItem(CItem item)
{
         using (SqlConnection conn = new
        SqlConnection(prCadConexion))
         {
                using (SqlCommand cmd = new
                SqlCommand("SPS_MTItemInsUpd", conn))
            {
                conn.Open();
                cmd.CommandTimeout = 0;
                cmd.CommandType =
CommandType.StoredProcedure;
                  cmd.Parameters.Add(new
                  SqlParameter("@ITM_ID",
                  item.id.ToString()));
                cmd.Parameters.Add(new
                SqlParameter("@FRM_ID", item.idForm));
                cmd.Parameters.Add(new
                SqlParameter("@CTRL_ID", item.idControl));
                  cmd.Parameters.Add(new
                  SqlParameter("@ITM_NAME", item.name));
                cmd.Parameters.Add(new
                SqlParameter("@ITM_LABEL", item.label));
                cmd.Parameters.Add(new
                SqlParameter("@ITM_REQUI", item.requerido));
```

```
                    cmd.Parameters.Add(new
                    SqlParameter("@ITM_SOLVE", item.resuelve));
                    cmd.Parameters.Add(new
                    SqlParameter("@ITM_OPTIONS", item.options));
                    cmd.Parameters.Add(new
                    SqlParameter("@ITM_VALUES", item.values));
                    cmd.Parameters.Add(new
                    SqlParameter("@ITM_POINTS", item.points));
                    cmd.Parameters.Add(new
                    SqlParameter("@TSOL_ID", item.idTipoSol));
                    cmd.Parameters.Add(new
                    SqlParameter("@ITM_STAT", item.stat));

                    cmd.ExecuteNonQuery();
                }
            }
}
```

- Función deleteItems: Inactiva los items en base de sus identificadores.

```
public static void deleteItems(string ids)
{
            ArrayList lvArrayParameter = new ArrayList();
            Boolean lbResultado = true;

            SqlParameter lvParametro = new SqlParameter("@IDS",
            SqlDbType.VarChar, 200);
            lvParametro.Value = ids;
            lvArrayParameter.Add(lvParametro);

            try
           {
                DBUtilSQLServer.executeNonQuery("SPS_MTItemDel",
                lvArrayParameter);
            }
            catch (Exception e)
            {
                lbResultado = false;
            }
}
```

- Función getEstrategias: Lista estrategias según los parámetros enviados.

```
public static DataTable getEstrategias(string str_form, string
str_name, string str_label, string str_tsol, string str_stat)
```

```
        {
            ArrayList lvArrayParameter = new ArrayList();

            SqlParameter lvParametro = new
            SqlParameter("@STR_NAME", SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(str_name).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@STR_FORM",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(str_form).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@STR_TSOL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(str_tsol).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@STR_LABEL",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(str_label).Trim();
            lvArrayParameter.Add(lvParametro);

            lvParametro = new SqlParameter("@STR_STAT",
            SqlDbType.VarChar, 100);
            lvParametro.Value = fnNull(str_stat).Trim();
            lvArrayParameter.Add(lvParametro);

            DataSet dst =
            DBUtilSQLServer.getDataset("SPS_MTStrategySel",
            lvArrayParameter);
            if (dst != null && dst.Tables.Count > 0)
            {
                return dst.Tables[0];
            }
            else
                return new DataTable();
}
```

- Función getEstrategiabyId: Recupera los datos de una estrategia basándose en el id del mismo.

```
public static CEstrategia getEstrategiabyId(int idEstrategia)
{
            CEstrategia estrategia = null;
            SqlConnection conn = null;
            SqlCommand cmd = null;
```

```
SqlDataReader loDr = null;
Int32 contador = 1;

try
{
    conn = new SqlConnection(prCadConexion);
    conn.Open();

    cmd = new SqlCommand("SPS_MTStrategySelObj",
    conn);

    cmd.CommandTimeout = 0;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(new SqlParameter("@STR_ID",
    idEstrategia));

    loDr = cmd.ExecuteReader();
    while (loDr.Read())
    {
        estrategia = new CEstrategia();

        estrategia.id =
        Int32.Parse(loDr.GetValue(loDr.GetOrdinal("S
        TRID")).ToString().Trim());
        estrategia.form =
        loDr.GetValue(loDr.GetOrdinal("FRMID")).ToSt
        ring().Trim();
        estrategia.name =
        loDr.GetValue(loDr.GetOrdinal("STRNAME")).To
        String().Trim();
        estrategia.label =
        loDr.GetValue(loDr.GetOrdinal("STRLABEL")).T
        oString().Trim();
        estrategia.tsol =
        loDr.GetValue(loDr.GetOrdinal("TSOLID")).ToS
        tring().Trim();
        estrategia.valor =
        loDr.GetValue(loDr.GetOrdinal("STRVAL")).ToS
        tring().Trim();
        estrategia.stat =
        loDr.GetValue(loDr.GetOrdinal("STRSTAT")).To
        String().Trim();
    }
}
catch (Exception e)
{
    throw e;
```

```
            }
            finally
            {
                #region Liberando Recursos
                loDr.Close();
                conn.Close();
                #endregion
            }
            return estrategia;
}
```

- Función guardarEstrategia: Inserta los datos de una estrategia.

```
public static void guardarEstrategia(CEstrategia estrategia)
{
          using (SqlConnection conn = new
          SqlConnection(prCadConexion))
            {
                using (SqlCommand cmd = new
                SqlCommand("SPS_MTStrategyInsUpd", conn))
                {
                    conn.Open();
                    cmd.CommandTimeout = 0;
                     cmd.CommandType =
                     CommandType.StoredProcedure;
                   cmd.Parameters.Add(new
                SqlParameter("@STR_ID",
                estrategia.id.ToString()));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_FORM", estrategia.form));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_NAME", estrategia.name));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_LABEL",
                    estrategia.label));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_TSOL", estrategia.tsol));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_VAL", estrategia.valor));
                    cmd.Parameters.Add(new
                    SqlParameter("@STR_STAT", estrategia.stat));

                    cmd.ExecuteNonQuery();
                }
            }
}
```

- Función deleteEstrategias: Inactiva las estrategias en base de sus identificadores.

```
public static void deleteEstrategias(string ids)
 {
            ArrayList lvArrayParameter = new ArrayList();
            Boolean lbResultado = true;

            SqlParameter lvParametro = new SqlParameter("@IDS",
            SqlDbType.VarChar, 200);
            lvParametro.Value = ids;
            lvArrayParameter.Add(lvParametro);

            try
                {
                DBUtilSQLServer.executeNonQuery("SPS_MTStrategyD
                el", lvArrayParameter);
            }
            catch (Exception e)
            {
                lbResultado = false;
            }
}
```

- Función generarScript: Genera un script de inserción según el id de formulario requerido.

```
public static StringBuilder generarScript(String idForm, String
idtForm)
{
            StringBuilder loStringScript = new StringBuilder();
            SqlConnection conn = null;
            SqlCommand cmd = null;
            SqlDataReader loDr = null;
            int liCount = 0;
            try
            {
                conn = new SqlConnection(prCadConexion);
                conn.Open();

                cmd = new SqlCommand("SPM_GenerateScript",
conn);
                cmd.CommandTimeout = 0;
                cmd.CommandType = CommandType.StoredProcedure;
```

```
                cmd.Parameters.Add(new SqlParameter("@FRM_ID",
                idForm));
                cmd.Parameters.Add(new SqlParameter("@FRM_IDT",
                idtForm));

                loDr = cmd.ExecuteReader();
                while (loDr.Read())
                {
                loStringScript.Append((String)loDr.GetValue(loDr
                .GetOrdinal("SCRIPT")));
                    loStringScript.Append(Environment.NewLine);
                    liCount++;
                }
            }
            catch (Exception e){}
            finally
            {
                #region Liberando Recursos
                loDr.Close();
                conn.Close();
                #endregion
            }
            return loStringScript;
}
```

- **Carpeta Mantenimiento**

  - **Usuario.aspx:** Página para mantenimiento de usuarios.

    - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            _Usuario = (String)Session["Usuario"];
            if (_Usuario == null)
            {
           Response.Redirect(HttpContext.Current.Request.Applica
           tionPath + "/Default.aspx");
            }
            lblUser.Text = _Usuario;
            cargaPerfiles();
        }
        cargaGrilla();
```

```
}
```

- Función cargaPerfiles: Llena el combobox de perfiles.

```
private void cargaPerfiles()
{
        DataSet dsPerfil = DBNet.getPerfiles();
        cmbPerfil.DataSource = dsPerfil.Tables[0];
        cmbPerfil.DataValueField = "PER_ABREVIA";
        cmbPerfil.DataTextField = "PER_NOMBRE";
        cmbPerfil.DataBind();

        if (cmbPerfil.Items.Count > 1)
        {
            cmbPerfil.Items.Insert(0, " -- Todos -- ");
            cmbPerfil.Items[0].Value = string.Empty;
        }
}
```

- Función cargaGrilla: Carga el gridview.

```
private void cargaGrilla()
{
        string codigo = txtCodigo.Value.Trim();
        string nombre = txtName.Value.Trim();
        string perfil = cmbPerfil.SelectedValue;

        Session["usrcod"] = codigo;
        Session["usrname"] = nombre;
        Session["usrprof"] = perfil;
        Session["usrstat"] = (chkEstado.Checked) ? "A" : "I";

        grilla.DataSource = getBusqueda();
        grilla.DataBind();
 }
```

- Función getBusqueda: Lista los usuarios de acuerdo a los parámetros.

```
public DataTable getBusqueda()
{
        DataTable loDT;
        string usr_cod = (String)Session["usrcod"];
        string usr_name = (String)Session["usrname"];
        string usr_prof = (String)Session["usrprof"];
        string usr_stat = (String)Session["usrstat"];
```

```
        loDT = DBNet.getUsuarios(usr_cod, usr_name, usr_prof,
usr_stat);

        if (loDT.Rows.Count != 0)//Mostrar numero de campos
devueltos
        { }
        else{ //Mostrar "No hay resultados para la busqueda"
        }
        return loDT;
}
```

- Función SubEliminar: Recorre toda la grilla encontrando las filas marcadas y recuperando sus identificadores para eliminar.

```
private void SubEliminar()
{
        String ids = "";
        foreach (GridViewRow fila in grilla.Rows)
        {
                CheckBox check = fila.FindControl("chkElegir")
                as CheckBox;

            if (check.Checked)
            {
                ids = ids + fila.Cells[0].Text + "|";
            }
        }
        ids = ids.Trim();
        DBNet.deleteUsuarios(ids);
}
```

- **UsuarioNuevo.aspx:** Página para creación de nuevos usuarios.

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
                //-- Código que debe ejecutarse la primera vez
                que se carga la página
            if (Request.QueryString["USR_ID"] != null)
            {
```

```
                //-- Recuperar parámetro enviado
                hidPk.Value = Request.QueryString["USR_ID"];

                //-- Recuperar objeto
                CUsuario usuario =
                DBNet.getUsuariobyId(Int32.Parse(hidPk.Value.ToS
                tring()));
                txtCodigo.Value = usuario.codigo;
                hidPassword.Value = usuario.clave;
                txtNombre.Value = usuario.nombre;
                cmbPerfil.SelectedValue = usuario.perfil;

                txtCodigo.Disabled = true;
                if (usuario.stat == "A")
                    chkEstado.Checked = true;
                else
                    chkEstado.Checked = false;
            }
            else
            {
                hidPk.Value = "0";
            }
            cargaPerfiles();
        }
}
```

- Función cargaPerfiles: Carga los perfiles en el combobox.

```
private void cargaPerfiles()
{
        DataSet dsPerfil = DBNet.getPerfiles();
        cmbPerfil.DataSource = dsPerfil.Tables[0];
        cmbPerfil.DataValueField = "PER_ABREVIA";
        cmbPerfil.DataTextField = "PER_NOMBRE";
        cmbPerfil.DataBind();
}
```

- Función btnGuardar_Click: Guarda los datos ingresados.

```
protected void btnGuardar_Click(object sender, EventArgs e)
{
        CUsuario usuario = new CUsuario();
        usuario.id = Int32.Parse(hidPk.Value);
        usuario.codigo = txtCodigo.Value.Trim().ToUpper();
        usuario.nombre = txtNombre.Value.Trim().ToUpper();
        usuario.clave = txtClave.Value.Trim();
        usuario.perfil = cmbPerfil.SelectedValue;
```

```
        usuario.stat = (chkEstado.Checked) ? "A" : "I";
        try
        {   //GRABA Y ACTUALIZA
            DBNet.guardarUsuario(usuario);
            Response.Redirect("Usuario.aspx");
        }
        catch (Exception ex)
        { }
 }
```

- **Formato.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            _Usuario = (String)Session["Usuario"];
            if (_Usuario == null)
            {
                Response.Redirect(HttpContext.Current.Request.Ap
           plicationPath + "/Default.aspx");
            }
            lblUser.Text = _Usuario;
        }
        cargaGrilla();
}
```

  - Función cargaGrilla: Carga el gridview.

```
 private void cargaGrilla()
{
        string idt = txtIdt.Value;
        string name = txtName.Value;
        string url = txtUrl.Value;

        Session["frmidt"] = idt;
        Session["frmname"] = name;
        Session["frmurl"] = url;
        Session["frmstat"] = (chkEstado.Checked) ? "A" : "I";

        grilla.DataSource = getBusqueda();
        grilla.DataBind();
 }
```

- Función getBusqueda: Lista los formularios de acuerdo a los parámetros.

```
public DataTable getBusqueda()
{
        DataTable loDT;
        string frm_idt = (String)Session["frmidt"];
        string frm_name = (String)Session["frmname"];
        string frm_url = (String)Session["frmurl"];
        string frm_stat = (String)Session["frmstat"];
     loDT = DBNet.getFormatos(frm_idt, frm_name, frm_url,
     frm_stat);
        return loDT;
}
```

- Función SubEliminar: Recorre toda la grilla encontrando las filas marcadas y recuperando sus identificadores para eliminar.

```
private void SubEliminar()
{
        String ids = "";
        foreach (GridViewRow fila in grilla.Rows)
        {
          CheckBox check = fila.FindControl("chkElegir") as
         CheckBox;
          if (check.Checked)
          {
              ids = ids + fila.Cells[0].Text + "|";
          }
        }
        ids = ids.Trim();
        DBNet.deleteFormatos(ids);
}
```

- **FormatoNuevo.aspx:**

- Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            if (Request.QueryString["FRM_ID"] != null)
            {
                hidPk.Value = Request.QueryString["FRM_ID"];
```

```
                CFormato formato =
                DBNet.getFormatobyId(Int32.Parse(hidPk.Value.ToS
                tring()));
                txtIdt.Value = formato.idt;
                txtIdt.Disabled = true;
                txtName.Value = formato.name;
                txtUrl.Value = formato.url;

                if (formato.stat == "A")
                    chkEstado.Checked = true;
                else
                    chkEstado.Checked = false;
            }
            else
            {
                hidPk.Value = "0";
            }
        }
}
```

- Función btnGuardar_Click: Guarda los datos ingresados.

```
protected void btnGuardar_Click(object sender, EventArgs e)
{
            CFormato formato = new CFormato();
             formato.id = Int32.Parse(hidPk.Value);
             formato.idt = txtIdt.Value.Trim().ToUpper();
             formato.name = txtName.Value.Trim().ToUpper();
             formato.url = txtUrl.Value.Trim().ToUpper();
             formato.stat = (chkEstado.Checked) ? "A" : "I";

             try
             {   //GRABA Y ACTUALIZA
                 DBNet.guardarFormato(formato);
                 Response.Redirect("Formato.aspx");
             }
             catch (Exception ex)
             { }
}
```

- **Item.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            _Usuario = (String)Session["Usuario"];
            if (_Usuario == null)
            {
                Response.Redirect(HttpContext.Current.Request.Ap
          plicationPath + "/Default.aspx");
            }
            lblUser.Text = _Usuario;
            cargaFormularios();
            cargaControles();
            cargaTSolucion();
            cargaGrilla();
        }
 }
```

  - Función cargaFormularios: Carga los formularios en el combobox.

```
private void cargaFormularios()
{
        DataSet dsForms = DBNet.getLstFormatos();
        cmbForm.DataSource = dsForms.Tables[0];
        cmbForm.DataValueField = "FRMID";
        cmbForm.DataTextField = "FRMNAME";
        cmbForm.DataBind();
        if (cmbForm.Items.Count > 1)
        {
            cmbForm.Items.Insert(0, " -- Todos -- ");
            cmbForm.Items[0].Value = string.Empty;
        }
}
```

  - Función cargaControles: Carga los controles en el combobox.

```
private void cargaControles()
{
        DataSet dsControles = DBNet.getLstControles();
        cmbCtrl.DataSource = dsControles.Tables[0];
```

```
        cmbCtrl.DataValueField = "CTRLID";
        cmbCtrl.DataTextField = "CTRLNAME";
        cmbCtrl.DataBind();
        if (cmbCtrl.Items.Count > 1)
        {
            cmbCtrl.Items.Insert(0, " -- Todos -- ");
            cmbCtrl.Items[0].Value = string.Empty;
        }
}
```

- Función cargaTSolucion: Carga los tipos de solución en el combobox.

```
private void cargaTSolucion()
{
        DataSet dsTSol = DBNet.getLstTSolucion();
        cmbTSol.DataSource = dsTSol.Tables[0];
        cmbTSol.DataValueField = "TSOLID";
        cmbTSol.DataTextField = "TSOLNAME";
        cmbTSol.DataBind();
        if (cmbTSol.Items.Count > 1)
        {
            cmbTSol.Items.Insert(0, " -- Todos -- ");
            cmbTSol.Items[0].Value = string.Empty;
        }
}
```

- Función cargaGrilla: Carga el gridview.

```
private void cargaGrilla()
{
        string name = txtName.Value.Trim();
        string label = txtLabel.Value.Trim();
        string idform = cmbForm.SelectedValue;
        string idctrl = cmbCtrl.SelectedValue;
        string idtsol = cmbTSol.SelectedValue;

        Session["itmname"] = name;
        Session["itmlabel"] = label;
        Session["itmform"] = idform;
        Session["itmctrl"] = idctrl;
        Session["itmtsol"] = idtsol;
        Session["itmreq"] = chkReq.Checked ? "T" : "F";
        Session["itmsolve"] = chkResuelve.Checked ? "T" : "F";
        Session["itmstat"] = chkEstado.Checked ? "A" : "I";
```

```
        grilla.DataSource = getBusqueda();
        grilla.DataBind();
}
```

- Función getBusqueda: Lista los items de acuerdo a los parámetros.

```
public DataTable getBusqueda()
{
        DataTable loDT;
        string itm_name = (String)Session["itmname"];
        string itm_label = (String)Session["itmlabel"];
        string itm_form = (String)Session["itmform"];
        string itm_ctrl = (String)Session["itmctrl"];
        string itm_tsol = (String)Session["itmtsol"];
        string itm_req = (String)Session["itmreq"];
        string itm_solve = (String)Session["itmsolve"];
        string itm_stat = (String)Session["itmstat"];

        loDT = DBNet.getItems(itm_name, itm_label, itm_form,
        itm_ctrl, itm_tsol, itm_req, itm_solve, itm_stat);
        return loDT;
}
```

- Función SubEliminar: Recorre toda la grilla encontrando las filas marcadas y recuperando sus identificadores para eliminar.

```
private void SubEliminar()
{
        String ids = "";
        foreach (GridViewRow fila in grilla.Rows)
        {
           CheckBox check = fila.FindControl("chkElegir") as
           CheckBox;
            if (check.Checked)
            {
                ids = ids + fila.Cells[0].Text + "|";
            }
        }
        ids = ids.Trim();
        DBNet.deleteItems(ids);
}
```

- **ItemNuevo.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            if (Request.QueryString["ITM_ID"] != null)
            {
                hidPk.Value = Request.QueryString["ITM_ID"];

                //Recuperar objeto
                CItem item =
                DBNet.getItembyId(Int32.Parse(hidPk.Value.ToStri
                ng()));
                txtNombre.Value = item.name;
                txtLabel.Value = item.label;
                cmbFormulario.SelectedValue = item.idForm;
                cmbTipoSol.SelectedValue = item.idTipoSol;
                cmbTCtrl.SelectedValue = item.idControl;

                if (item.requerido == "T")
                    chkRequerido.Checked = true;
                else
                    chkRequerido.Checked = false;
                if (item.resuelve == "T")
                    chkResuelve.Checked = true;
                else
                    chkResuelve.Checked = false;
                if (item.stat == "A")
                    chkEstado.Checked = true;
                else
                    chkEstado.Checked = false;
            }
            else
            {
                hidPk.Value = "0";
            }
            cargaFormularios();
            cargaControles();
            cargaTSolucion();
        }
}
```

- Función cargaFormularios: Carga los formularios en el combobox.

```
private void cargaFormularios()
{
        DataSet dsForms = DBNet.getLstFormatos();
        cmbFormulario.DataSource = dsForms.Tables[0];
        cmbFormulario.DataValueField = "FRMID";
        cmbFormulario.DataTextField = "FRMNAME";
        cmbFormulario.DataBind();
}
```

- Función cargaControles: Carga los controles en el combobox.

```
private void cargaControles()
{
        DataSet dsControles = DBNet.getLstControles();
        cmbTCtrl.DataSource = dsControles.Tables[0];
        cmbTCtrl.DataValueField = "CTRLID";
        cmbTCtrl.DataTextField = "CTRLNAME";
        cmbTCtrl.DataBind();
}
```

- Función cargaTSolucion: Carga los tipos de solución en el combobox.

```
private void cargaTSolucion()
{
        DataSet dsTSol = DBNet.getLstTSolucion();
        cmbTipoSol.DataSource = dsTSol.Tables[0];
        cmbTipoSol.DataValueField = "TSOLID";
        cmbTipoSol.DataTextField = "TSOLNAME";
        cmbTipoSol.DataBind();
}
```

- Función btnGuardar_Click: Guarda los datos ingresados.

```
protected void btnGuardar_Click(object sender, EventArgs e)
{
        CItem item = new CItem();
        item.id = Int32.Parse(hidPk.Value);
        item.name = txtNombre.Value.Trim().ToUpper();
        item.label = txtLabel.Value.Trim().ToUpper();
        item.idForm = cmbFormulario.SelectedValue;
        item.idControl = cmbTCtrl.SelectedValue;
```

```
            item.idTipoSol = cmbTipoSol.SelectedValue;
            item.requerido = (chkRequerido.Checked) ? "T" : "F";
            item.resuelve = (chkResuelve.Checked) ? "T" : "F";
            item.stat = (chkEstado.Checked) ? "A" : "I";

            item.options = hidOptions.Value.Trim();
            item.values = hidValues.Value.Trim();
            item.points = hidPoints.Value.Trim();

            if (cmbTCtrl.SelectedValue.Equals("1") ||
            cmbTCtrl.SelectedValue.Equals("4")){
                item.idTipoSol = "-1";
            }
            else if (cmbTCtrl.SelectedValue.Equals("3")){
                item.idTipoSol = "1";
            }
            try
            {   //GRABA Y ACTUALIZA
                DBNet.guardarItem(item);
                Response.Redirect("Item.aspx");
            }
            catch (Exception ex)
            { }
}
```

- **Estrategia.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
            if (!Page.IsPostBack)
            {
                _Usuario = (String)Session["Usuario"];
                if (_Usuario == null)
                {
                    Response.Redirect(HttpContext.Current.Request.Ap
                    plicationPath + "/Default.aspx");
                }
                lblUser.Text = _Usuario;
                cargaFormularios();
                cargaTSolucion();
                cargaGrilla();
            }
}
```

- Función cargaFormularios: Carga los formularios en el combobox.

```
private void cargaFormularios()
{
        DataSet dsForms = DBNet.getLstFormatos();
        cmbSForm.DataSource = dsForms.Tables[0];
        cmbSForm.DataValueField = "FRMID";
        cmbSForm.DataTextField = "FRMNAME";
        cmbSForm.DataBind();

        if (cmbSForm.Items.Count > 1)
        {
            cmbSForm.Items.Insert(0, " -- Todos -- ");
            cmbSForm.Items[0].Value = string.Empty;
        }
}
```

- Función cargaTSolucion: Carga los tipos de solución en el combobox.

```
private void cargaTSolucion()
{
        DataSet dsTSol = DBNet.getLstTSolucion();
        cmbSTSol.DataSource = dsTSol.Tables[0];
        cmbSTSol.DataValueField = "TSOLID";
        cmbSTSol.DataTextField = "TSOLNAME";
        cmbSTSol.DataBind();

        if (cmbSTSol.Items.Count > 1)
        {
            cmbSTSol.Items.Insert(0, " -- Todos -- ");
            cmbSTSol.Items[0].Value = string.Empty;
        }
}
```

- Función cargaGrilla: Carga el gridview.

```
private void cargaGrilla()
{
        string name = txtSNombre.Value;
        string label = txtSLabel.Value;
        string idform = cmbSForm.SelectedValue;
        string idtsol = cmbSTSol.SelectedValue;

        Session["strname"] = name;
```

```
        Session["strlabel"] = label;
        Session["strform"] = idform;
        Session["strtsol"] = idtsol;
        Session["strstat"] = (chkEstado.Checked) ? "A" : "I";

        grilla.DataSource = getBusqueda();
        grilla.DataBind();
}
```

- Función getBusqueda: Lista las estrategias de acuerdo a los parámetros.

```
public DataTable getBusqueda()
{
        DataTable loDT;
        string str_name = (String)Session["strname"];
        string str_label = (String)Session["strlabel"];
        string str_stat = (String)Session["strstat"];
        string str_form = (String)Session["strform"];
        string str_tsol = (String)Session["strtsol"];

        loDT = DBNet.getEstrategias(str_form, str_name,
str_label, str_tsol, str_stat);

        if (loDT.Rows.Count != 0)
        {
            //Mostrar numero de campos devueltos
        }

        else{ //Mostrar "No hay resultados para la busqueda"
        }

        return loDT;
}
```

- Función SubEliminar: Recorre toda la grilla encontrando las filas marcadas y recuperando sus identificadores para eliminar.

```
private void SubEliminar()
{
        String ids = "";
        foreach (GridViewRow fila in grilla.Rows)
        {
           CheckBox check = fila.FindControl("chkElegir") as
           CheckBox;
```

```
            if (check.Checked)
            {
                ids = ids + fila.Cells[0].Text + "|";
            }
        }
        ids = ids.Trim();
        DBNet.deleteEstrategias(ids);
}
```

- **EstrategiaNueva.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            if (Request.QueryString["STR_ID"] != null)
            {
                hidPk.Value = Request.QueryString["STR_ID"];

                CEstrategia estrategia =
                DBNet.getEstrategiabyId(Int32.Parse(hidPk.Value.
                ToString()));
                txtName.Value = estrategia.name;
                txtLabel.Value = estrategia.label;
                txtValor.Value = estrategia.valor;
                cmbFormulario.SelectedValue = estrategia.form;
                cmbTipoSol.SelectedValue = estrategia.tsol;

                if (estrategia.stat == "A")
                    chkEstado.Checked = true;
                else
                    chkEstado.Checked = false;
            }
            else
            {
                hidPk.Value = "0";
            }
            cargaFormularios();
            cargaTipoSolucion();

    }
}
```

- Función cargaFormularios: Carga los formularios en el combobox.

```
private void cargaFormularios()
{
        DataSet dsForms = DBNet.getLstFormatos();
        cmbFormulario.DataSource = dsForms.Tables[0];
        cmbFormulario.DataValueField = "FRMID";
        cmbFormulario.DataTextField = "FRMNAME";
        cmbFormulario.DataBind();
}
```

- Función cargaTipoSolucion: Carga los tipos de solución en el combobox.

```
private void cargaTipoSolucion()
{
        DataSet dsTSol = DBNet.getLstTSolucion();
        cmbTipoSol.DataSource = dsTSol.Tables[0];
        cmbTipoSol.DataValueField = "TSOLID";
        cmbTipoSol.DataTextField = "TSOLNAME";
        cmbTipoSol.DataBind();
}
```

- Función btnGuardar_Click: Guarda los datos ingresados.

```
protected void btnGuardar_Click(object sender, EventArgs e)
{
            CEstrategia estrategia = new CEstrategia();
            estrategia.id = Int32.Parse(hidPk.Value);
            estrategia.name = txtName.Value.Trim().ToUpper();
            estrategia.label = txtLabel.Value.Trim().ToUpper();
            estrategia.valor = txtValor.Value.Trim().ToUpper();
                estrategia.form =
                cmbFormulario.SelectedValue.ToUpper();
            estrategia.tsol =
           cmbTipoSol.SelectedValue.ToUpper();
            estrategia.stat = (chkEstado.Checked) ? "A" : "I";
            try
            {   //GRABA Y ACTUALIZA
                DBNet.guardarEstrategia(estrategia);
                Response.Redirect("Estrategia.aspx");
            }
            catch (Exception ex)
            { }
}
```

- **Generar.aspx:**

  - Función Page_Load: Evento que se ejecuta cada vez que se carga la página.

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            //Recibe como parámetro FRM_ID
            if (Request.QueryString["FRM_ID"] != null)
            {
                hidForm.Value = Request.QueryString["FRM_ID"];
                hidtForm.Value = Request.QueryString["FRM_IDT"];

                GenerarScript(hidForm.Value.Trim(),
                hidtForm.Value.Trim(), "txt");
            }
            else
            {
                hidForm.Value = "0";
                hidtForm.Value = "0";
            }
        }
}
```

  - Función GenerarScript: Genera un archivo txt con los datos necesarios para la creación de formularios en los dispositivos móviles.

```
private void GenerarScript(String idForm, String idtForm, String
extension)
{
        String filePath = "../Scripts/" + idtForm.ToString() +
        "." + extension.ToString();
        String fic = Server.MapPath(filePath.ToString());

        using (FileStream fstr = File.Open(fic,
        FileMode.OpenOrCreate, FileAccess.ReadWrite))
        {
            StreamWriter sw = new StreamWriter(fstr);
            sw.Write(DBNet.generarScript(idForm,
           idtForm).ToString());
            sw.Flush();
            sw.Dispose();
        }
```

```
        Page page = HttpContext.Current.Handler as Page;
        if (page != null) {
      ClientScript.RegisterStartupScript(page.GetType(),
     "myScript", "close_facebox()",true);
        }
}
```

- **Hojas de Estilo**

  - **CSSIndex.css:** Estilos para página de Login.

```
#login-bg {
background: url(../Img/login_bg.jpg) no-repeat top center;
}

#login-holder {
margin: 0px auto 0 auto;
width: 508px;
}

#logo-login {
float: left;
height: 60px;
margin: 145px 0 0 15px;
}

.clear {
clear: both;
font-size: 0px;
height: 0;
line-height: 0px;
margin: 0px;
padding: 0px;
}

#loginbox {
background: url(../Img/loginbox_bg.png) no-repeat;
font-size: 12px;
height: 212px;
line-height: 12px;
padding-top: 60px;
position: relative;
width: 508px;
}

#login-inner {
```

```
color: #161616;
font-family: Tahoma;
font-size: 13px;
line-height: 12px;
margin: 0 auto;
width: 310px;
}

#login-inner th {
padding: 0 0 6px 0;
text-align: left;
width: 95px;
}

#login-inner td {
padding: 0 0 6px 0;
}

.login-inp {
color: #000;
font-size: 13px;
width: 200px;
}

.submit-login {
      height:30px;
      padding: 5px;
      background: #94B52C;
      color: #FFFFFF;
      border: 1px solid #719C27;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
}

#mensaje{
      margin-bottom:20px;
      font-family: Tahoma;
      font-weight: bold;
      padding:  5px 5px 5px 20px;
      background: #FBD3B1;
      color: #ce2700;
      border: 1px solid #F5C69A;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
}
```

- **CSSMaster.css:** Estilos para las páginas de mantenimientos.

```
textarea:focus,input:focus {outline: none;}

.princ{
	min-height: 100%;
	height: auto!important;
}

fieldset {
	border: 0;
	margin: 0 0;
	padding: 0;
}

/*-- CABECERA --*/

#cont_top {
	background: url(../Img/top_fondo.jpg) top center repeat-x;
	border-bottom: 1px solid #7e7e7e;
	height: 92px;
}

#top {
	margin: 0 auto;
	max-width: 1260px;
	min-width: 850px;
	position: relative;
}

#logo {
	float: left;
	margin: 20px 0 0 15px;
}

.clear{
	clear: both;
	font-size: 0px;
	height: 0;
	line-height: 0px;
	margin: 0px;
	padding: 0px;
}
```

```
#contBotones{
	height: 30px;
	background: url(../Img/bot_fondo.jpg) repeat-x;
	color:#000;
	border:none;
	display:block;
	padding-top:10px;
	padding-left: 50px;
}

.img_btn{
	position:relative;
	float:left;
	margin: 0px 20px 0 20px;
	width: 65px;
}

#contCampos{
	height: 300px;
	background-color:#F0F1F3;
	color:#000;
	border:none;
	padding:5px;
	-webkit-border-radius: 5px;
	-moz-border-radius: 5px;
	border-radius: 5px;
}

#contGrilla{
	height: 100%;
	color:#000;
	border:none;
	padding:45px;
	-webkit-border-radius: 5px;
	-moz-border-radius: 5px;
	border-radius: 5px;
}

#contData{
	height: 100%;
	background: url(../Img/cont_fondo.jpg) repeat-x;
	color:#000;
	border:none;
	-webkit-border-radius: 5px;
	-moz-border-radius: 5px;
	border-radius: 5px;
}
```

```
/* Lista de Campos*/
.list_reset, .list_reset li {
      margin: 0;
      padding: 5px;
      list-style: none;
      border: none;
}

.clearfix{
      display:block
}

#contCampos fieldset{
      border: none;
}

.error{
      color: Red;
      display: inline;
}

.label{
      width:100px;
      position:relative;
      float:left;
      font-size: 12px;
      font-family: Arial;
}

.input{
      width:305px;
      position:relative;
      float:left;
}

.inputChk{
      width:100px;
      position:relative;
      float:left;
}

.span{
      width:150px;
      position:relative;
      float:left;
}
```

```
/* Botones */

.btnBuscar{
      height:30px;
      padding: 5px;
      background: #94B52C;
      color: #FFFFFF;
      border: 1px solid #719C27;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
}

/* Grilla */

#filtros{
      padding: 45px 45px 0 45px;
}

#filBusqueda{
      width: 850px;
      margin-left: 200px;
      border: 1px solid #F0F1F3;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
      box-shadow: 1px 1px 1px rgba(0, 0, 0, 0.4);
}

.gridview{
    margin:5px;
    border:none;
    background-color:#F0F1F3;
}

.gridview tr {
      text-align: center;
    border-bottom: solid;
    border-color:#FFFFFF;
    border-width:2px;
    font-family: Tahoma;
      font-size: 13px;
}

.gridview th{
      height:30px;
    border:none;
```

```
    background: url(../Img/table_header.jpg) repeat-x;
    border-left: 1px solid #373737;
    text-align:center;
    display: table-cell;
     vertical-align: inherit;
     font-family: Tahoma;
     font-size: 13px;
     font-weight: bold;
     line-height: 14px;
     margin: 0 0 0 10px;
     padding: 0 10px 0 0;
}

.gridview td{
     height:30px;
    border:none;
}

.over{
     background: #FBD3B1;
}

#footer {
     color: #818181;
     font-size: 11px;
     line-height: 11px;
     padding: 15px 0 15px 25px;
     margin: 0 20px;
     margin-top:150px;
     border-top: 1px solid #dbdbdb;
     border-bottom: 1px solid #dbdbdb;
     text-align: center;
}

.corte {
     clear: both;
     padding-top: 20px;
}

/*-- Datos requeridos --*/

.spanalert{
     background-color: #FBD3B1;
     margin-left:10px;
     font-size: 12px;
     color: #CE2700;
     font-family: Tahoma;
```

```
      font-weight: bold;
      border: solid 1px #CE2700;
      padding:3px;
      -webkit-border-radius: 3px;
      -moz-border-radius: 3px;
      border-radius: 3px;
}
```

- **CSSMenu.css:** Estilos para el menú de las páginas.

```
* { margin: 0; padding: 0; }

a:hover { color: white; }

.nav-wrap {
      font-family: Tahoma;
      font-size: 13px;
      background: url(../Img/menu_fondo.jpg) repeat-x;
}

.group:after { visibility: hidden; display: block; content: "";
clear: both; height: 0; }
*:first-child+html .group { zoom: 1; } /* IE7 */

#example-one {
      margin: 0 45px;
      list-style: none;
      position: relative;
      width: 100%;
}
#example-one li {
      display: inline-block;
}
#example-one a {
      color: #bbb;
      font-size: 14px;
      float: left;
      padding: 6px 10px 4px 10px;
      text-decoration: none;
      text-transform: uppercase;
}
#example-one a:hover {
      color: white;
}
#magic-line {
      position: absolute;
```

```
      bottom: -1px;
      left: 0;
      width: 90px;
      height: 1px;
      background: #A0D50E;
}
.current_page_item a {
      color: white !important;
}
.ie6 #example-one li, .ie7 #example-one li {
      display: inline;
}
.ie6 #magic-line {
      bottom: -3px;
}
```

- **CSSVModal.css:** Estilos para las ventanas modales (pop ups) para la creación de nuevos objetos.

```
#titulo
{
      text-align:center;
      padding: 5px;
      background: #94B52C;
      color: #FFFFFF;
      border: 1px solid #719C27;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
      font-family: Tahoma;
      font-size: 13px;
}

.btnGuardar
{
      height:30px;
      padding: 5px;
      background: #94B52C;
      color: #FFFFFF;
      border: 1px solid #719C27;
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      border-radius: 5px;
}
```

```
.btnAddRecord
{
      width:32px;
      height:32px;
      background-image:url(../Img/addrecord.png);
      border:none;
}

.tooltip
{
      color: #fff;
     background:#1d1d1d;
     padding:10px;
     position:absolute;
     z-index:1000;
     font-family: Tahoma;
      font-size: 13px;
      width:300px;
     box-shadow: 0px 0px 20px rgba(0,0,0,0.9);
     border:1px solid black;

      opacity:0.9;
      filter:alpha(opacity=90); /* For IE8 and earlier */

     -webkit-border-radius: 5px;
     -moz-border-radius: 5px;
     border-radius: 5px;
}

/* ProgressBar */

.meter {
      height: 20px;  /* Can be anything */
      position: relative;
      background: #FFFFFF;
      -moz-border-radius: 10px;
      -webkit-border-radius: 10px;
      border-radius: 10px;
      /*padding: 10px;*/
      -webkit-box-shadow: inset 0 -1px 1px rgba(255,255,255,0.3);
      -moz-box-shadow   : inset 0 -1px 1px rgba(255,255,255,0.3);
      box-shadow        : inset 0 -1px 1px rgba(255,255,255,0.3);
}

.meter > span {
      display: block;
      height: 100%;
```

```
            -webkit-border-top-right-radius: 8px;
        -webkit-border-bottom-right-radius: 8px;
                -moz-border-radius-topright: 8px;
            -moz-border-radius-bottomright: 8px;
                    border-top-right-radius: 8px;
                border-bottom-right-radius: 8px;
            -webkit-border-top-left-radius: 8px;
         -webkit-border-bottom-left-radius: 8px;
                -moz-border-radius-topleft: 8px;
             -moz-border-radius-bottomleft: 8px;
                    border-top-left-radius: 8px;
                border-bottom-left-radius: 8px;
        background-color: rgb(43,194,83);
        background-image: -webkit-gradient(
          linear,
          left bottom,
          left top,
          color-stop(0, rgb(43,194,83)),
          color-stop(1, rgb(84,240,84))
         );
        background-image: -moz-linear-gradient(
          center bottom,
          rgb(43,194,83) 37%,
          rgb(84,240,84) 69%
         );
        -webkit-box-shadow:
          inset 0 2px 9px  rgba(255,255,255,0.3),
          inset 0 -2px 6px rgba(0,0,0,0.4);
        -moz-box-shadow:
          inset 0 2px 9px  rgba(255,255,255,0.3),
          inset 0 -2px 6px rgba(0,0,0,0.4);
        box-shadow:
          inset 0 2px 9px  rgba(255,255,255,0.3),
          inset 0 -2px 6px rgba(0,0,0,0.4);
        position: relative;
        overflow: hidden;
}

.meter > span:after, .animate > span > span {
        content: "";
        position: absolute;
        top: 0; left: 0; bottom: 0; right: 0;
        background-image:
           -webkit-gradient(linear, 0 0, 100% 100%,
              color-stop(.25, rgba(255, 255, 255, .2)),
             color-stop(.25, transparent), color-stop(.5,
             transparent),
```

```
            color-stop(.5, rgba(255, 255, 255, .2)),
            color-stop(.75, rgba(255, 255, 255, .2)),
            color-stop(.75, transparent), to(transparent)
        );
    background-image:
          -moz-linear-gradient(
            -45deg,
            rgba(255, 255, 255, .2) 25%,
            transparent 25%,
            transparent 50%,
            rgba(255, 255, 255, .2) 50%,
            rgba(255, 255, 255, .2) 75%,
            transparent 75%,
            transparent
        );
    z-index: 1;
    -webkit-background-size: 50px 50px;
    -moz-background-size: 50px 50px;
    background-size: 50px 50px;
    -webkit-animation: move 2s linear infinite;
    -moz-animation: move 2s linear infinite;
       -webkit-border-top-right-radius: 8px;
    -webkit-border-bottom-right-radius: 8px;
         -moz-border-radius-topright: 8px;
       -moz-border-radius-bottomright: 8px;
             border-top-right-radius: 8px;
          border-bottom-right-radius: 8px;
      -webkit-border-top-left-radius: 20px;
     -webkit-border-bottom-left-radius: 20px;
           -moz-border-radius-topleft: 20px;
        -moz-border-radius-bottomleft: 20px;
              border-top-left-radius: 20px;
           border-bottom-left-radius: 20px;
    overflow: hidden;
}

.animate > span:after {
    display: none;
}
@-webkit-keyframes move {
    0% {
       background-position: 0 0;
    }
    100% {
       background-position: 50px 50px;
    }
}
```

```
@-moz-keyframes move {
    0% {
       background-position: 0 0;
    }
    100% {
       background-position: 50px 50px;
    }
}

.green > span {
     background-color: #B1D149;
     background-image: -moz-linear-gradient(top, #B1D149,
#D9E96C);
     background-image: -webkit-gradient(linear,left top,left
bottom,color-stop(0, #B1D149),color-stop(1, #D9E96C));
     background-image: -webkit-linear-gradient(#B1D149,
#D9E96C);
}

.nostripes > span > span, .nostripes > span:after {
     -webkit-animation: none;
     -moz-animation: none;
     background-image: none;
}
```

2. MÓDULO MÓVIL

- **Carpeta src**

  - **ActMenu.java:** Esta clase se encarga de la creación dinámica de los componentes del menú por medio de un control de grilla.

```
package com.tesis.app;
public class ActMenu extends Activity {
// Código (Métodos, funciones, etc.).
}
```

➢ Atributos de la clase:

```
String tag = ActMenu.class.getSimpleName();
CForm theForm = new CForm();
Context context = this;
GridView grdMenu;
boolean signalOn;
```

- Método onCreate: Establece las acciones a realizar al ser presionado cada elemento del menú.

```
public void onCreate(Bundle savedInstanceState){
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_act_menu);

          try{
          SubSetControles();
          grdMenu.setOnItemClickListener(
           new OnItemClickListener()
           {
                @Override
                public void onItemClick(AdapterView<?> parent,
                View v, int position, long id)
                {
                     switch (position) {
                     case 0: //Sincronizar
                          theForm = new CForm();
                          signalOn =
                          Utilitario.fnVerSignal(context);
                          if (signalOn == true)
                          {
                               Intent newSincForm = new
                          Intent(ActMenu.this,ActTipoSinc.class
                          );
                               startActivity(newSincForm);
                          } else{
                               Toast.makeText(ActMenu.this, "No
                          hay señal",
                          Toast.LENGTH_SHORT).show();
                        }
                          break;
                     case 1: //Evaluar
                          Intent newRunForm = new
                          Intent(ActMenu.this,ActTipoEv.class);

                     newRunForm.addFlags(Intent.FLAG_ACTIVITY_N
                     O_HISTORY);
                          startActivity(newRunForm);
                          break;
                     case 2: //Mostrar resultados
                          theForm = new CForm();
                          Intent newResForm = new
                     Intent(ActMenu.this,ActTipoRes.class);
```

```
                        newResForm.addFlags(Intent.FLAG_ACTIVITY_N
                        O_HISTORY);
                              startActivity(newResForm);
                              return;
                        }
                  }
             }
            );
            }catch(Exception e)
            {
             Log.i(tag,e.toString());
            }
      }
```

- Método SubSetControles: Sitúa los controles en una Vista Grid para que puedan ser correctamente distribuidos en la pantalla del dispositivo móvil.

```
public void SubSetControles(){
            grdMenu = (GridView)findViewById(R.id.GridView01);
            grdMenu.setAdapter(new ImageAdapter(this));
}
```

- Clase Anidada ImageAdapter: Sitúa en la vista las imágenes y textos pertenecientes al menú.

```
public class ImageAdapter extends BaseAdapter{
      Context mContext;
      public static final int ACTIVITY_CREATE = 10;

      public ImageAdapter(Context c){
            mContext = c;
      }
      @Override
      public int getCount() {
            return Configuracion.LstOpcMenu.length;
      }
      @Override
      public   View   getView(int   position,   View   convertView,
      ViewGroup parent) {
          View v;
      AlphaAnimation  blinkanimation=  new  AlphaAnimation(0,  1);
      //Cambia la imagen de estado invisible a visible
            blinkanimation.setDuration(1500);
          if(convertView == null){
```

```
            LayoutInflater li = getLayoutInflater();
            v = li.inflate(R.layout.icon, null);
                  //Textos
                  TextView                    tv                    =
                  (TextView)v.findViewById(R.id.icon_text);
                  tv.setTextColor(Color.WHITE);
                  tv.setText(Configuracion.LstOpcMenu[position]);
                  //Iconos
                  ImageView                    iv                    =
                  (ImageView)v.findViewById(R.id.icon_image);

            iv.setImageResource(Configuracion.LstIcoMenu[position
            ]);
                  iv.startAnimation(blinkanimation);
                  tv.startAnimation(blinkanimation);
            }
            else{
            v = convertView;
            }
            return v;
      }
      @Override
      public Object getItem(int position) {
            // TODO Auto-generated method stub
            return null;
      }
      @Override
      public long getItemId(int position) {
            // TODO Auto-generated method stub
            return 0;
      }
}
```

- ActTipoSinc.java: Clase que sitúa al fichero layout "activity_act_tipsinc" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

```
package com.tesis.app;
public class ActTipoSinc extends Activity {
// Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
final String tag = ActTipoSinc.class.getSimpleName();
Context context = this;
```

```
ProgressDialog progressDialog;
Handler progressHandler;
```

- Método onCreate: Al momento de crearse la actividad se establece el funcionamiento de los controles.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_act_tipsinc);

                  Button       btnSincForm       =       (Button)
      this.findViewById(R.id.btnSincForm);
      btnSincForm.setOnClickListener(new Button.OnClickListener()
      {
            public void onClick(View v)
            {
                  EditText     formSincNumber     =     (EditText)
                  findViewById(R.id.edtSincCodigo);
                  if
                  (!formSincNumber.getText().toString().isEmpty())
                  {
                  SincForm(formSincNumber.getText().toString());
                  }else
                  {    Toast.makeText(ActTipoSinc.this,   "Ingrese
                  nombre              de              Formulario",
                  Toast.LENGTH_LONG).show();
                  }
            }
      });

      final       ImageView       imgQR       =       (ImageView)
      findViewById(R.id.imageViewQR);
      imgQR.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            imgQR.setImageResource(R.drawable.draw_qrin);
            Intent            intent            =            new
            Intent("com.google.zxing.client.android.SCAN");
               intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
               startActivityForResult(intent, 0);
          }
      });
 }
```

- Método onActivityResult: Este método es llamado cuando la actividad para obtener el código QR termina, procediendo a situar el texto leído por el escáner en el control EditText.

```
public void onActivityResult(int requestCode, int resultCode,
Intent intent) {
     EditText        formSincNumber        =        (EditText)
     findViewById(R.id.edtSincCodigo);
      ImageView          imgQR          =          (ImageView)
     findViewById(R.id.imageViewQR);
      imgQR.setImageResource(R.drawable.draw_qr);
        if (requestCode == 0) {
           if (resultCode == RESULT_OK) {
                       String           contents           =
         intent.getStringExtra("SCAN_RESULT");
             formSincNumber.setText(contents);
           } else if (resultCode == RESULT_CANCELED) {
           Log.i(tag,"Escaneo cancelado");
           }
        }
     }
```

- Método SincForm: Método que se dedica exclusivamente a la sincronización del formulario ingresado mediante su identificador.

```
public void SincForm(String formSincCode) {
     new HttpSincronizar(formSincCode,this).execute();
}
```

- **ActTipoEv.java:** Clase que sitúa al fichero layout "activity_act_tipev" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

```
package com.tesis.app;
public class ActTipoEv extends Activity {
     // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
final String tag = ActTipoEv.class.getSimpleName();
List<String> listValues = new ArrayList<String>();
List<String> listLabel = new ArrayList<String>();
Context context = this;
```

```
String itemSeleccionado="";
long seleccionado;
String[][] list = null;
```

- onCreate: Al momento de crearse la actividad obtiene los elementos que deben ser situados en el control spinner. También establece la manera en que el control botón funcionará, enviando el identificador del formulario seleccionado a otra clase.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_act_tipev);
      Spinner sp = (Spinner) findViewById(R.id.spinner1);

      //Carga lista de formularios en memoria
      DBHandler dbh = new DBHandler (context);
     list = dbh.actTipoEv_getList();
     dbh.close();
      if( list != null)
      {
            if(list.length>=1)
            {
                  for (int i = 0; i < list.length; i++) {
                        //Almacena en control spinner el label de
                        la lista
                        listLabel.add(list[i][0]);
                       //Almacena el identificador del archivo
                        listValues.add(list[i][1]);
                  }
            }else{
                  Toast.makeText(ActTipoEv.this,     "No     posee
                  formularios", Toast.LENGTH_LONG).show();
                  finish();
            }
      }
      else {
            Toast.makeText(ActTipoEv.this,        "No        posee
            formularios", Toast.LENGTH_LONG).show();
            finish();
      }
     //Arreglo de nombres es insertado en el control spinner
     ArrayAdapter<String>          adapter          =          new
ArrayAdapter<String>(this,
             android.R.layout.simple_spinner_item, listLabel);
```

```
adapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
sp.setAdapter(adapter);
sp.setOnItemSelectedListener(new OnItemSelectedListener() {
public void onItemSelected(AdapterView<?> parentView, View
selectedItemView,
      int position, long id) {
           seleccionado                                    =
           parentView.getItemIdAtPosition(position);
           itemSeleccionado = listValues.get(position);
       }
      @Override
      public void onNothingSelected(AdapterView<?> arg0) {
           // TODO Auto-generated method stub
      }
   });

 Button          btnRunForm          =          (Button)
 this.findViewById(R.id.btnRunForm);
 btnRunForm.setOnClickListener(new Button.OnClickListener()
 {
      public void onClick(View v)
      {
           if(itemSeleccionado == ""){
                Toast.makeText(ActTipoEv.this,  "Seleccione
                un item", Toast.LENGTH_LONG).show();
           }
           else
           {
                int                index                =
                itemSeleccionado.lastIndexOf('.');
                if (index > 0)
 itemSeleccionado=itemSeleccionado.substring(0,index);
 Intent            newRunForm          =             new
 Intent(ActTipoEv.this,RunForm.class);
 //Finaliza cuando se llame a otra activity
 newRunForm.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
 Bundle extras = new Bundle();
extras.putString("formNumber",itemSeleccionado);
newRunForm.putExtras(extras);
 startActivity(newRunForm);
 finish();
           }
      }
 });
}
```

- **ActTipoRes.java:** Clase que sitúa al fichero layout "activity_act_tipres" como vista contenedora por medio del cuál podemos acceder a los diferentes controles mostrados al usuario y poder capturar el ingreso del identificador del formulario a sincronizar.

```
package com.tesis.app;
public class ActTipoRes extends Activity {
      // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
final String tag = ActTipoRes.class.getSimpleName();
List<String> listName = new ArrayList<String>();
final Context context = this;
long seleccionado;
String selForm="";
String selVersion="";
```

- onCreate: Al momento de crearse la actividad obtiene los elementos que deben ser situados en el control spinner.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_act_tipres);
      Spinner sp = (Spinner) findViewById(R.id.spinner1);
      final List<String> listForm = new ArrayList<String>();
      final List<String> listVersion = new ArrayList<String>();

      //Obtiene la lista de respuestas de formularios
     DBHandler dbh = new DBHandler (context);
     String[][] lista = dbh.actTipoRes_getList();
      dbh.close();

      if(lista.length>=1)
      {
            for (int i = 0; i < lista.length; i++) {
                listForm.add(lista[i][0]);
                listVersion.add(lista[i][1]);

      listName.add(lista[i][2]!=null?lista[i][2]:"FormSN");
            }
      }else{
```

```
            Toast.makeText(ActTipoRes.this, "No posee
            formularios", Toast.LENGTH_LONG).show();
            finish();
      }

     ArrayAdapter<String>          adapter          =          new
ArrayAdapter<String>(this,
             android.R.layout.simple_spinner_item, listName);

      adapter.setDropDownViewResource(android.R.layout.simple_spi
      nner_dropdown_item);
      sp.setAdapter(adapter);
      sp.setOnItemSelectedListener(new OnItemSelectedListener() {
      public void onItemSelected(AdapterView<?> parentView, View
      selectedItemView,
                  int position, long id) {
            seleccionado                                    =
      parentView.getItemIdAtPosition(position);
            selForm = listForm.get(position);
            selVersion = listVersion.get(position);
          }

          @Override
          public void onNothingSelected(AdapterView<?> arg0) {
                // TODO Auto-generated method stub
          }
        });

      Button          btnResForm          =          (Button)
      this.findViewById(R.id.btnResForm);
      btnResForm.setOnClickListener(new Button.OnClickListener()
      {
            public void onClick(View v)
            {
                  if(selForm == ""){
                        Toast.makeText(ActTipoRes.this,
                  "Seleccione un item", Toast.LENGTH_LONG).show();
                  }
                  else
                  {
                        Intent newRunForm = new
            Intent(ActTipoRes.this,ActResultD.class);

            newRunForm.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
                        Bundle extras = new Bundle();

extras.putInt("formID",Integer.valueOf(selForm));
```

```
            extras.putInt("resVersion",Integer.valueOf(selVersion
            ));
                  newRunForm.putExtras(extras);
                        startActivity(newRunForm);
                  }
            }
      });
}
```

- **CForm. java:** Esta clase representa al objeto Form, almacenará sus atributos y métodos. Implementa la interface Parcelable que permite que las instancias del objeto puedan ser escritas y restauradas en la clase Parcel. La clase Parcel es un contenedor de datos y referencias de objetos que pueden ser enviados.

```
package com.tesis.app.clases;
public class CForm implements Parcelable  {
      // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase: Estos atributos cuentan con métodos setters y getters implementados por defecto.

```
private Integer FRMID;
private String FRMIDT;
private String FRMNAME;
private String FRMURL;
private Integer FRMVERSI;
public List<CItem> items;
public List<CItemData> itemsData;
public List<CStrategy> strategy;
public List<CResult> result;
```

- CForm: Constructor de la clase, inicializa los objetos contenidos en esta.

```
public CForm()
{
      FRMID = 0;
      FRMIDT = "";
      FRMNAME = "";
      FRMURL = "";
      this.items = new ArrayList<CItem>(); //Lista de controles
      this.itemsData  =  new  ArrayList<CItemData>();  //Lista  de
      datos ingresados
```

```
      this.strategy = new ArrayList<CStrategy>(); //Lista de
      estrategias
      this.result = new ArrayList<CResult>(); //Lista de
resultados
}
```

- CForm: Constructor de la clase sobrecargado para recibir un objeto Parcel, para luego enviarlo al método “readFromParcel”.

```
public CForm(Parcel in) {
      FRMID = 0;
      FRMIDT = "";
      FRMNAME = "";
      FRMURL = "";
      this.items = new ArrayList<CItem>();
      this.itemsData = new ArrayList<CItemData>();
      this.strategy = new ArrayList<CStrategy>();
      this.result = new ArrayList<CResult>();
      readFromParcel(in);
}
```

- describeContents: Describe contenidos de la clase Parcelable.

```
@Override
public int describeContents() {
      return 0;
}
```

- writeToParcel: Realiza una especie de compresión de los objetos en la clase contenedora Parcel.

```
@Override
public void writeToParcel(Parcel dest, int flags) {
      dest.writeInt(FRMID);
      dest.writeString(FRMIDT);
      dest.writeString(FRMNAME);
      dest.writeString(FRMURL);
      dest.writeTypedList(items);
      dest.writeTypedList(itemsData);
      dest.writeTypedList(strategy);
      dest.writeTypedList(result);
}
```

- readFromParcel: Realiza la lectura (descompresión) de los objetos desde la clase Parcel.

```
private void readFromParcel(Parcel in) {
      FRMID = in.readInt();
      FRMIDT = in.readString();
      FRMNAME = in.readString();
      FRMURL = in.readString();
      in.readTypedList(items, CItem.CREATOR);
      in.readTypedList(itemsData, CItemData.CREATOR);
      in.readTypedList(strategy, CStrategy.CREATOR);
      in.readTypedList(result, CResult.CREATOR);
}
```

- **CMapper.java:** Esta clase se encarga de la conversión de un objeto desde y hacia el formato Json.

```
package com.tesis.app.clases;
public class CMapper {
      // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
private static ObjectMapper m = new ObjectMapper();
private static JsonFactory jf = new JsonFactory();
```

- fromJson: Recibe una cadena Json y lo convierte en un Objeto.

```
public static <T> Object fromJson(String jsonAsString, Class<T>
BeanClass)
{
        try
        {
                return m.readValue(jsonAsString, BeanClass);
           }
        catch (Exception e)
           {
                return null;
           }
}
```

- toJson: Convierte el objeto ingresado a una cadena Json y si la variable prettyPrint es “true”, la cadena se imprime con saltos de línea e indentaciones que agregan facilidad al momento de ser leído el Json por el usuario.

```
public static String toJson(Object Bean, boolean prettyPrint)
{
        StringWriter sw = new StringWriter();
        try
        {
           JsonGenerator jg = jf.createJsonGenerator(sw);
            if (prettyPrint)
            {
                jg.useDefaultPrettyPrinter();
            }
            m.writeValue(jg, Bean);
      }
        catch (Exception e)
      {
           sw.append(e.getMessage());
      }
        return sw.toString();
}
```

- **DBHandler.java:** Extiende de la clase SQLiteOpenHelper la cual nos permite crear la base de datos y actualizar la estructura de las tablas y manejar datos dentro de ella.

```
package com.tesis.app.dal;
public class DBHandler extends SQLiteOpenHelper {
     // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
private static final int DATABASE_VERSION = 12;
private static final String DATABASE_NAME = "DB_DATILEv1.db";
final String tag = DBHandler.class.getSimpleName();
Context setcontext =null;
String PATH = "";
```

- DBHandler: Es el constructor de la clase DBHandler, sitúa el nombre de la base de datos y su versión.

```
super(context, DATABASE_NAME, null, DATABASE_VERSION);
PATH =
context.getDatabasePath(DBHandler.DATABASE_NAME).toString();
setcontext = context;
```

- onCreate: Se le llama cuando la base de datos se crea por primera vez. Aquí es donde se define la estructura de las tablas y se cargan eventualmente los datos iniciales.

```
public void onCreate(SQLiteDatabase db) {
      String script ="";
      SQLiteDatabase checkDB = null;
      checkDB = SQLiteDatabase.openDatabase(PATH, null,
      SQLiteDatabase.OPEN_READONLY);

      try {
            BufferedReader in =
            HttpSincronizar.fnSincronizar(setcontext,
            EnumUrl.SINCRONIZAR,"");
            String inputLine;
            while ((inputLine = in.readLine()) != null)
            {     script = inputLine;
                  db.execSQL(script);
            }
            in.close();
      }
      catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }
         checkDB.close();
}
```

- onUpgrade: Recibe el nombre de la base de datos y llama al método onCreate para volver a crear la base de datos.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        onCreate(db);
}
```

- sincForm_setForm: Inserta los datos del formulario sincronizado en la base de datos.

```
public boolean sincForm_setForm(String formSincCode)
{
      SQLiteDatabase db = this.getWritableDatabase();
      String script ="";
      try{
            BufferedReader in =
            HttpSincronizar.fnSincronizar(setcontext,
            EnumUrl.SINCRONIZARFORM,formSincCode);
            String inputLine;
            while ((inputLine = in.readLine()) != null)
            {     script = inputLine;
                  db.execSQL(script);
            }
            in.close();
     }catch (Exception e)
      {
            Log.i(tag,e.toString());
            return false;
      }
        db.close();
      return true;
}
```

- actTipoEv_getList: Selecciona todos los formularios que estén activos en la base de datos del dispositivo móvil.

```
public String[][] actTipoEv_getList()
{
      SQLiteDatabase db = this.getWritableDatabase();
      int i=0;
      String[][] list = null;
      try{
      Cursor cur = db.rawQuery("SELECT FRMNAME, FRMIDT FROM
      TBM_FORM WHERE FRMACTIV='T';", null);

      if (cur != null && cur.moveToFirst()) {
            list = new String [cur.getCount()][2];
                  do {
                        list[i][0]=cur.getString(0);
                        list[i][1]=cur.getString(1);
                        i++;
                 } while(cur.moveToNext());
      }
     }catch (Exception e)
      {
            Log.i(tag,e.toString());
```

```
            return list;
      }
      db.close();
      return list;
 }
```

- runForm_getForm: Recupera registros de la base de datos para almacenarlo en la clase CForm.

```
public CForm runForm_getForm(CForm form, String formCode)
 {
      SQLiteDatabase db = this.getWritableDatabase();
      Integer frmversi=0;
      Integer frmid=0;
      CForm theForm = new CForm();
      try{

      //TBM_FORM
            Cursor c = db.rawQuery("SELECT FRMID, FRMNAME,
      FRMURL, FRMVERSI FROM TBM_FORM WHERE FRMIDT='"+formCode+"'
      AND FRMACTIV='T';", null);
      //Nos aseguramos de que existe al menos un registro
      if (c != null && c.moveToFirst()) {
            theForm.setFRMID(c.getInt(0));
            frmid =c.getInt(0);
            theForm.setFRMIDT(formCode);
            theForm.setFRMNAME(c.getString(1));
            theForm.setFRMURL(c.getString(2));
            theForm.setFRMVERSI(c.getInt(3));
            frmversi = c.getInt(3);
      }

      int itmid=0;
        int i=0;
        int cont=0;
        //TBM_ITEM
      Cursor cur = db.rawQuery("SELECT ITMID, ITMNAME, ITMLABEL,
      CTRLID, ITMREQUI, ITMOPT, ITMSOLVE, " +
                  "TSOLID, ITMVAL, ITMPNT FROM TBM_ITEM WHERE
                  FRMID="+frmid+";", null);
      if (cur != null && cur.moveToFirst()) {
            //Recorremos el cursor hasta que no haya más
registros
           do {
                  CItem tempItem =  new CItem();
                  tempItem.setITMID(cur.getInt(0));
```

```
      itmid = cur.getInt(0);
      tempItem.setITMNAME(cur.getString(1));
      tempItem.setITMLABEL(cur.getString(2));
      tempItem.setCTRLID(cur.getInt(3));
      tempItem.setITMREQUI(cur.getString(4));
      tempItem.setITMSOLVE(cur.getString(6));
      tempItem.setTSOLID(cur.getInt(7));

      theForm.getItems().add(tempItem);

      i=0;
      String itmopt="";
                              Cursor cur1 =
      db.rawQuery("SELECT OPTID, OPTINDEX, OPTNAME
      FROM TBM_ITMOPT WHERE ITMID="+itmid+";", null);
if (cur1 != null && cur1.moveToFirst()) {
            //Recorremos el cursor hasta que no haya
            más registros
      do {
                  CItemOpt tempItemOpt =  new
                  CItemOpt();
                  tempItemOpt.setOPTID(cur1.getInt(0));
                  tempItemOpt.setITMID(itmid);
                                             v
      tempItemOpt.setOPTINDEX(cur1.getInt(1));
      tempItemOpt.setOPTNAME(cur1.getString(2));

theForm.items.get(cont).getOpt().add(tempItemOpt);
                  if(i==0)
                        itmopt += cur1.getString(2);
                  else
                        itmopt += "|" +
                        cur1.getString(2);

                  i++;
     } while(cur1.moveToNext());
}
//Para llenar array del choice
theForm.items.get(cont).setITMOPT(itmopt);

i=0;
Cursor cur2 = db.rawQuery("SELECT VALID, VALINDEX,
VALNAME FROM TBM_ITMVAL WHERE ITMID="+itmid+";",
null);
if (cur2 != null && cur2.moveToFirst()) {
           //Recorremos el cursor hasta que no haya
           más registros
```

```
            do {
                        CItemVal tempItemVal =  new
                        CItemVal();
                        tempItemVal.setVALID(cur2.getInt(0));
                        tempItemVal.setITMID(itmid);

                tempItemVal.setVALINDEX(cur2.getInt(1));
                tempItemVal.setVALNAME(cur2.getString(2));

          theForm.items.get(cont).getVal().add(tempItemVal);
                        i++;
                } while(cur2.moveToNext());
          }

          i=0;
                Cursor cur3 = db.rawQuery("SELECT PNTID,
                PNTINDEX, PNTNAME FROM TBM_ITMPNT WHERE
                ITMID="+itmid+";", null);
          if (cur3 != null && cur3.moveToFirst()) {
                    //Recorremos el cursor hasta que no haya
                más registros
                do {
                        CItemPnt tempItemPnt =  new
                        CItemPnt();
                        tempItemPnt.setPNTID(cur3.getInt(0));
                        tempItemPnt.setITMID(itmid);

          tempItemPnt.setPNTINDEX(cur3.getInt(1));

          tempItemPnt.setPNTNAME(cur3.getString(2));
          theForm.items.get(cont).getPnt().add(tempItemPnt);
                        i++;
                } while(cur3.moveToNext());
          }

        cont++;
         } while(cur.moveToNext());
}

int version = 0;
//TBD_ITEM
Cursor ver = db.rawQuery("SELECT VERSION FROM TBD_ITEM
WHERE FRMID="+frmid+" ORDER BY VERSION DESC LIMIT 1;",
null);
if (ver != null && ver.moveToFirst()) {
     version = (int) ver.getInt(0);
 }
```

```
      Cursor curso = db.rawQuery("SELECT DITMID, FRMID, VERSION,
      ITMID, DITMVAL, DITMPNT, DITMGOOD " +
                  " FROM TBD_ITEM WHERE FRMID="+frmid+" AND
      VERSION="+version+";", null);

      if (curso != null && curso.moveToFirst()) {
            //Recorremos el cursor hasta que no haya más
registros
           do {
                 CItemData tempItemD =  new CItemData();
                 tempItemD.setDITMID(curso.getInt(0));
                 tempItemD.setFRMID(curso.getInt(1));
                 tempItemD.setVERSION(curso.getInt(2));
                 tempItemD.setITMID(curso.getInt(3));
                 tempItemD.setDITMVAL(curso.getString(4));
                 tempItemD.setDITMPNT(curso.getString(5));
                 tempItemD.setDITMGOOD(curso.getString(6));
                 theForm.getItemsData().add(tempItemD);
           } while(curso.moveToNext());
      }

      //TBM_STRATEGY
      Cursor curs = db.rawQuery("SELECT
      STRID,FRMID,STRNAME,STRLABEL,STRTSOL,STRVAL FROM
      TBM_STRATEGY " +
                  "WHERE FRMID="+frmid+";", null);

      if (curs != null && curs.moveToFirst()) {
            //Recorremos el cursor hasta que no haya más
registros
           do {
                 CStrategy tempStrategy =  new CStrategy();
                 tempStrategy.setSTRID(curs.getInt(0));
                 tempStrategy.setFRMID(curs.getInt(1));
                 tempStrategy.setSTRNAME(curs.getString(2));
                 tempStrategy.setSTRLABEL(curs.getString(3));
                 tempStrategy.setSTRTSOL(curs.getString(4));
                 tempStrategy.setSTRVAL(curs.getString(5));
                 theForm.getStrategy().add(tempStrategy);

           } while(curs.moveToNext());
      }
      }catch (Exception e)
      {
            Log.i(tag,e.toString());
      }
```

```
      db.close();
      return theForm;
}
```

- resolForm_setItemsData: Inserta en la base de datos los datos del formulario ingresados por el usuario.

```
public void resolForm_setItemsData(CForm theForm)
{   SQLiteDatabase db = this.getWritableDatabase();
      Integer frmid=0;
      Integer itmid=0;
      Integer version=0;
      String ditmval="";
      String ditmpnt="";
      String ditmgood="";
      try{
          //El contador de filas me dice cuántas veces corrí el
          formulario
        Cursor c = db.rawQuery("SELECT COUNT(*) FROM TBD_RESULT
        WHERE FRMID="+theForm.getFRMID()+";",null);
        if (c != null && c.moveToFirst()) {
           version = c.getInt(0);
        }
      version = version + 1;

      for (int i=0;i<theForm.items.size();i++) {
          frmid = theForm.itemsData.get(i).getFRMID();
          itmid = theForm.itemsData.get(i).getITMID();
          ditmval = theForm.itemsData.get(i).getDITMVAL();
          ditmpnt = theForm.itemsData.get(i).getDITMPNT();
          ditmgood = theForm.itemsData.get(i).getDITMGOOD();

          Log.i(tag,"INSERT TBD_ITEM:"+version+"/"+ditmval);
                        String INSERT_TBD_ITEM = "INSERT INTO
          TBD_ITEM (FRMID, VERSION, ITMID, DITMVAL, DITMPNT,
          DITMGOOD) values ("+
                     frmid +","+
                     version +","+
                     itmid +",'"+
                     ditmval +"','"+
                     ditmpnt +"','"+
                     ditmgood +"');";
          db.execSQL(INSERT_TBD_ITEM);
      }
     }catch (Exception e)
      {
```

```
            Log.i(tag,e.toString());
      }
         db.close();
    }
```

- resolForm_setResult: Inserta el resultado final de la evaluación en la base de datos.

```
public void resolForm_setResult(CForm theForm ,Float result,
String strname)
    {
      Integer idStrategy =0;
      Integer version=0;
      SQLiteDatabase db = this.getWritableDatabase();
      try{
      Cursor cur = db.rawQuery("SELECT STRID FROM TBM_STRATEGY
      WHERE " +
                "FRMID="+theForm.getFRMID()+" AND " +
                "STRNAME='"+strname+"';" ,null);
        if (cur != null && cur.moveToFirst()) {
               idStrategy = (int) cur.getLong(0);
        }

            //El contador de filas me dice cuántas veces corrí el
            formulario
        Cursor c = db.rawQuery("SELECT COUNT(*) FROM TBD_RESULT
        WHERE FRMID="+theForm.getFRMID()+";",null);
        if (c != null && c.moveToFirst()) {
            version = c.getInt(0);
        }
      version = version + 1;

         String INSERT_TBD_RESULT = "INSERT INTO TBD_RESULT
         (FRMID, VERSION, STRID, RESVAL) values ("+
                theForm.getFRMID() +","+
                      version +","+
                      idStrategy +",'"+
                      result +"');";
      db.execSQL(INSERT_TBD_RESULT);
    }catch (Exception e)
      {
            Log.i(tag,e.toString());
      }
      db.close();
}
```

- actResult_getStrlabel: Obtiene el texto que fue parametrizado como resultado en el módulo web.

```
public String actResult_getStrlabel(CForm theForm)
{
      SQLiteDatabase db = this.getWritableDatabase();
      Integer strid = 0;
      String strlabel = "";
      try{
                  Cursor c = db.rawQuery("SELECT STRID FROM
      TBD_RESULT WHERE RESID=" +
                        "(SELECT RESID FROM TBD_RESULT WHERE
      FRMID="+theForm.getFRMID()+" ORDER BY VERSION DESC LIMIT
      1);", null);
      //Nos aseguramos de que existe al menos un registro
      if (c != null && c.moveToFirst()) {
          strid =c.getInt(0);
      }

                 Cursor cu = db.rawQuery("SELECT STRLABEL FROM
      TBM_STRATEGY WHERE FRMID="+theForm.getFRMID()+" AND
      STRID="+strid+";", null);
      //Nos aseguramos de que existe al menos un registro
      if (cu != null && cu.moveToFirst()) {
          strlabel =cu.getString(0);
      }
     }catch (Exception e)
      {
           Log.i(tag,e.toString());
           return "";
      }
      db.close();
      return strlabel;
}
```

- actResult_getResval: Obtiene el valor del resultado guardado anteriormente en la base de datos.

```
public Float actResult_getResval(CForm theForm)
{
      SQLiteDatabase db = this.getWritableDatabase();
      Float resval = (float) 0;
      try{
      Cursor c = db.rawQuery("SELECT RESVAL FROM TBD_RESULT WHERE
      RESID=" +
```

```
      "(SELECT RESID FROM TBD_RESULT WHERE
      FRMID="+theForm.getFRMID()+" ORDER BY VERSION DESC LIMIT
      1);", null);
      //Nos aseguramos de que existe al menos un registro
      if (c != null && c.moveToFirst()) {
            resval =c.getFloat(0);
      }
     }catch (Exception e)
      {
            Log.i(tag,e.toString());
      }
      db.close();
      return resval;
}
```

- actResult_setName: Inserta el nombre que se le da a una resolución determinada de un formulario.

```
public boolean actResult_setName(CForm theForm, String formName)
{
      SQLiteDatabase db = this.getWritableDatabase();
      try{
            String UPDATE_TBD_RESULT = "UPDATE TBD_RESULT SET
            RESNAME ='"+formName+"' WHERE RESID=" +
                        "(SELECT RESID FROM TBD_RESULT WHERE
            FRMID="+theForm.getFRMID()+" ORDER BY VERSION DESC
            LIMIT 1);";
             db.execSQL(UPDATE_TBD_RESULT);

             //Llena TBD_RESULT
                        Cursor curs = db.rawQuery("SELECT RESID,
            FRMID, STRID, RESVAL, VERSION, RESNAME FROM
            TBD_RESULT WHERE RESID=" +
                        "(SELECT RESID FROM TBD_RESULT WHERE
            FRMID="+theForm.getFRMID()+" ORDER BY VERSION DESC
            LIMIT 1);", null);

            if (curs != null && curs.moveToFirst()) {
                  //Recorremos el cursor hasta que no haya más
                  registros
                  do {
                        CResult tempResult =  new CResult();
                        tempResult.setRESID(curs.getInt(0));
                        tempResult.setFRMID(curs.getInt(1));
                        tempResult.setSTRID(curs.getInt(2));
                        tempResult.setRESVAL(curs.getFloat(3));
```

```
                    tempResult.setVERSION(curs.getInt(4));
                    tempResult.setRESNAME(curs.getString(5));
                    theForm.getResult().add(tempResult);

                } while(curs.moveToNext());
            }

        }catch (Exception e) {
            Log.i(tag,e.toString());
            return false;
        }
        db.close();
        return true;
}
```

- actTipoRes_getList: Obtiene la lista de las resoluciones de formularios guardadas en el dispositivo móvil.

```
public String[][] actTipoRes_getList()
{
        SQLiteDatabase db = this.getWritableDatabase();
        int count=0;
            Cursor c1 = db.rawQuery("SELECT COUNT(*) FROM
        TBD_RESULT;", null);
            //Nos aseguramos de que existe al menos un registro
        if (c1 != null && c1.moveToFirst()) {
            count =c1.getInt(0);
        }
        String[][] resName = new String[count][3];
        try{
             Cursor c = db.rawQuery("SELECT FRMID, VERSION,
            RESNAME FROM TBD_RESULT;", null);

             //Nos aseguramos de que existe al menos un registro
             int i=0;
        if (c != null && c.moveToFirst()) {
             do {
                  resName[i][0] =String.valueOf(c.getInt(0));
                  resName[i][1] =String.valueOf(c.getInt(1));
                  resName[i][2] = c.getString(2);
                  i++;
             } while(c.moveToNext());
        }
        }catch (Exception e) {
            Log.i(tag,e.toString());
            return resName;
```

```
        }
        db.close();
        return resName;
}
```

- actResultD_getForm: Obtiene el formulario según su identificador y su versión.

```
public CForm actResultD_getForm(CForm theForm, int formID, int
version)
{
        SQLiteDatabase db = this.getWritableDatabase();

        try{

        //TBM_FORM
              Cursor c = db.rawQuery("SELECT FRMID,  FRMNAME,
        FRMURL, FRMIDT FROM TBM_FORM WHERE FRMID="+formID+";",
        null);
        //Nos aseguramos de que existe al menos un registro
        if (c != null && c.moveToFirst()) {
              theForm.setFRMID(c.getInt(0));
              theForm.setFRMNAME(c.getString(1));
              theForm.setFRMURL(c.getString(2));
              theForm.setFRMIDT(c.getString(3));
        }

        int itmid=0;
          int i=0;
          int cont=0;
          //TBM_ITEM
        Cursor cur = db.rawQuery("SELECT ITMID, ITMNAME, ITMLABEL,
        CTRLID, ITMREQUI, ITMOPT, ITMSOLVE, " +
                    "TSOLID, ITMVAL, ITMPNT FROM TBM_ITEM WHERE
                    FRMID="+formID+";", null);
        if (cur != null && cur.moveToFirst()) {
              //Recorremos el cursor hasta que no haya más
              registros
              do {
                    CItem tempItem =  new CItem();
                    tempItem.setITMID(cur.getInt(0));
                    itmid = cur.getInt(0);
                    tempItem.setITMNAME(cur.getString(1));
                    tempItem.setITMLABEL(cur.getString(2));
                    tempItem.setCTRLID(cur.getInt(3));
                    tempItem.setITMREQUI(cur.getString(4));
```

```
      tempItem.setITMSOLVE(cur.getString(6));
      tempItem.setTSOLID(cur.getInt(7));

      theForm.getItems().add(tempItem);

      i=0;
      String itmopt="";
                              Cursor cur1 =
      db.rawQuery("SELECT OPTID, OPTINDEX, OPTNAME
      FROM TBM_ITMOPT WHERE ITMID="+itmid+";", null);
if (cur1 != null && cur1.moveToFirst()) {
            //Recorremos el cursor hasta que no haya
            más registros
      do {
                  CItemOpt tempItemOpt =  new
                  CItemOpt();
                  tempItemOpt.setOPTID(cur1.getInt(0));
                  tempItemOpt.setITMID(itmid);

tempItemOpt.setOPTINDEX(cur1.getInt(1));
tempItemOpt.setOPTNAME(cur1.getString(2));
theForm.items.get(cont).getOpt().add(tempItemOpt);
                  if(i==0)
                        itmopt += cur1.getString(2);
                  else
                        itmopt += "|" +
                        cur1.getString(2);

                  i++;
      } while(cur1.moveToNext());
}
//Para llenar array del control que lo solicite
theForm.items.get(cont).setITMOPT(itmopt);

i=0;
      Cursor cur2 = db.rawQuery("SELECT VALID,
VALINDEX, VALNAME FROM TBM_ITMVAL WHERE
ITMID="+itmid+";", null);
if (cur2 != null && cur2.moveToFirst()) {
            //Recorremos el cursor hasta que no haya
      más registros
      do {
                  CItemVal tempItemVal =  new
                  CItemVal();
                  tempItemVal.setVALID(cur2.getInt(0));
                  tempItemVal.setITMID(itmid);
```

```
tempItemVal.setVALINDEX(cur2.getInt(1));
            tempItemVal.setVALNAME(cur2.getString(2));
            theForm.items.get(cont).getVal().add(tempItemVal);
                              i++;
                  } while(cur2.moveToNext());
            }

            i=0;
                  Cursor cur3 = db.rawQuery("SELECT PNTID,
                  PNTINDEX, PNTNAME FROM TBM_ITMPNT WHERE
                  ITMID="+itmid+";", null);
            if (cur3 != null && cur3.moveToFirst()) {
                        //Recorremos el cursor hasta que no haya
                        más registros
                  do {
                              CItemPnt tempItemPnt =  new
                              CItemPnt();
                              tempItemPnt.setPNTID(cur3.getInt(0));
                              tempItemPnt.setITMID(itmid);

            tempItemPnt.setPNTINDEX(cur3.getInt(1));
            tempItemPnt.setPNTNAME(cur3.getString(2));
            theForm.items.get(cont).getPnt().add(tempItemPnt);
                              i++;
                  } while(cur3.moveToNext());
            }

          cont++;
           } while(cur.moveToNext());
       }

       //TBD_ITEM
      //Llama desde ActResult directamente de botón Resultado
       Cursor curso = db.rawQuery("SELECT DITMID, FRMID, VERSION,
       ITMID, DITMVAL, DITMPNT, DITMGOOD " +
                  " FROM TBD_ITEM WHERE FRMID="+formID+" AND
            VERSION="+version+";", null);

       if (curso != null && curso.moveToFirst()) {
            //Recorremos el cursor hasta que no haya más
            registros
           do {
                  CItemData tempItemD =  new CItemData();
                  tempItemD.setDITMID(curso.getInt(0));
                  tempItemD.setFRMID(curso.getInt(1));
                  tempItemD.setVERSION(curso.getInt(2));
```

```
            tempItemD.setITMID(curso.getInt(3));
            tempItemD.setDITMVAL(curso.getString(4));
            tempItemD.setDITMPNT(curso.getString(5));
            tempItemD.setDITMGOOD(curso.getString(6));
            theForm.getItemsData().add(tempItemD);
      } while(curso.moveToNext());
}

 // TBD_RESULT
            Cursor cursor = db.rawQuery("SELECT RESID,
      FRMID, STRID, RESVAL, VERSION, RESNAME FROM
      TBD_RESULT " +
            "WHERE FRMID="+theForm.getFRMID()+" AND
      VERSION="+version+";", null);

if (cursor != null && cursor.moveToFirst()) {
      //Recorremos el cursor hasta que no haya más
      registros
      do {
            CResult tempResult =  new CResult();
            tempResult.setRESID(cursor.getInt(0));
            tempResult.setFRMID(cursor.getInt(1));
            tempResult.setSTRID(cursor.getInt(2));
            tempResult.setRESVAL(cursor.getFloat(3));
            tempResult.setVERSION(cursor.getInt(4));
            tempResult.setRESNAME(cursor.getString(5));
            theForm.getResult().add(tempResult);

      } while(cursor.moveToNext());
}

//TBM_STRATEGY
            Cursor curs = db.rawQuery("SELECT
      STRID,FRMID,STRNAME,STRLABEL,STRTSOL,STRVAL FROM
      TBM_STRATEGY " +
            "WHERE FRMID="+formID+";", null);

if (curs != null && curs.moveToFirst()) {
      //Recorremos el cursor hasta que no haya más
      registros
      do {
            CStrategy tempStrategy =  new CStrategy();
            tempStrategy.setSTRID(curs.getInt(0));
            tempStrategy.setFRMID(curs.getInt(1));
            tempStrategy.setSTRNAME(curs.getString(2));
            tempStrategy.setSTRLABEL(curs.getString(3));
            tempStrategy.setSTRTSOL(curs.getString(4));
```

```
                tempStrategy.setSTRVAL(curs.getString(5));
                theForm.getStrategy().add(tempStrategy);

            } while(curs.moveToNext());
        }
        }catch (Exception e)
        {
            Log.i(tag,e.toString());
        }
        db.close();
        return theForm;
}
```

- **XmlGuiEditText.java:** Esta clase contiene al control EditText el cuál será creado dinámicamente, la clase extiende de LinearLayout, que facilita su posición dentro de la vista que se mostrará al usuario.

```
package com.tesis.app.widget;
public class XmlGuiEditText extends LinearLayout  {
     // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
TextView label;
EditText txtBox;
```

- XmlGuiEditText: Constructor de la clase. Establece las propiedades del control Label y EditText.

```
public XmlGuiEditText(Context context,String labelText,String
initialText) {
          super(context);
          this.setOrientation(LinearLayout.VERTICAL);

          label = new TextView(context);
          label.setTextSize(16);
          label.setText(labelText);
          txtBox = new EditText(context);
          txtBox.setText(initialText);
          txtBox.setLayoutParams(new
          LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,View
          Group.LayoutParams.WRAP_CONTENT));
          txtBox.setBackgroundResource(R.xml.custom_edittext);
          this.addView(label);
```

```
            this.addView(txtBox);
}
```

- makeNumeric: Hace que la información ingresada en el control EditText sea mediante teclado numérico.

```
public void makeNumeric()
{
      DigitsKeyListener dkl = new DigitsKeyListener(true,true);
      txtBox.setKeyListener(dkl);
}
```

- getValue: Obtiene el texto del control EditText.

```
public String getValue()
{
      return txtBox.getText().toString();
}
```

- setValue: Establece el String entrante como valor del texto del EditText.

```
public void setValue(String v)
{
      txtBox.setText(v);
}
```

- **XmlGuiChoice.java:** Contiene el control Spinner, el cuál se creará dinámicamente.

```
package com.tesis.app.widget;
public class XmlGuiChoice extends LinearLayout  {
// Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
String tag = XmlGuiChoice.class.getName();
TextView label;
ArrayAdapter<String> aa;
Spinner spinner;
```

- XmlGuiChoice: Constructor de la clase. Establece las propiedades del control Label y Spinner.

```
public XmlGuiChoice(Context context,String labelText,String
options) {
          super(context);
          this.setOrientation(LinearLayout.VERTICAL);

          label = new TextView(context);
          label.setText(labelText);
          label.setTextSize(16);
          spinner = new Spinner(context);
          String []opts = options.split("\\|");
          aa = new ArrayAdapter<String>( context,
          android.R.layout.simple_spinner_item,opts);
          spinner.setAdapter(aa);
          spinner.setBackgroundResource(R.xml.custom_spinner);

          spinner.setPopupBackgroundResource(R.xml.custom_spinn
          erlist);
          this.addView(label);
          this.addView(spinner);
     }
```

- getValue: Obtiene el texto del item seleccionado.

```
public String getValue()
{
     return (String) spinner.getSelectedItem().toString();
}
```

- getPosition: Obtiene el índice de posición del item seleccionado.

```
public String getPosition()
{
     return String.valueOf(spinner.getSelectedItemPosition());
}
```

- **RunForm.java:** Esta clase carga los datos del formulario sincronizado y se encarga de situar los controles dinámicamente en la vista para ser mostrada en el dispositivo móvil.

```
package com.tesis.app.form;
public class RunForm extends Activity {
     // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
String tag = RunForm.class.getSimpleName(); //Solo para log
public static Float resultEval=(float) 0;
ProgressDialog progressDialog;
static CForm theForm;
Handler progressHandler;
String formCode = "";
Context context = this;
```

- onCreate: Constructor de la clase. Extrae los datos del formulario electo.

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        theForm = new CForm();
        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        formCode = extras.getString("formNumber");

            //Carga datos del formulario
            DBHandler dbh = new DBHandler (context);
            theForm = dbh.runForm_getForm(theForm,formCode);
                  dbh.close();

     DisplayForm();
}
```

- DisplayForm: Sitúa dinámicamente los controles en la vista.

```
private boolean DisplayForm()
{
      try
      {
            ScrollView sv = new ScrollView(this);
            final LinearLayout ll = new LinearLayout(this);
        sv.addView(ll);
        ll.setOrientation(android.widget.LinearLayout.VERTICAL);

        // Crea dinámicamente los controles en la vista
        int i;
        for (i=0;i<theForm.items.size();i++) {
           Log.i(tag, theForm.items.get(i).getITMREQUI());
           if (theForm.items.get(i).getCTRLID().equals(1)) {
```

```
            theForm.items.get(i).obj = new
        XmlGuiEditText(this,
        theForm.items.get(i).getITMLABEL() +
        (theForm.items.get(i).getITMREQUI().equals("T") ? "*"
        : "") ,"");
            ll.addView((View) theForm.items.get(i).obj);
        }
        if (theForm.items.get(i).getCTRLID().equals(2)) {
            theForm.items.get(i).obj = new
        XmlGuiEditText(this,
        theForm.items.get(i).getITMLABEL() +
        (theForm.items.get(i).getITMREQUI().equals("T") ? "*"
        : ""),"");
            (

(XmlGuiEditText)theForm.items.get(i).obj).makeNumeric();
            ll.addView((View) theForm.items.get(i).obj);
        }
        if (theForm.items.get(i).getCTRLID().equals(3)) {
            theForm.items.get(i).obj = new
        XmlGuiChoice(this, theForm.items.get(i).getITMLABEL()
        + (theForm.items.get(i).getITMREQUI().equals("T") ?
        "*" : ""),theForm.items.get(i).getITMOPT());
            ll.addView((View) theForm.items.get(i).obj);
        }
        if (theForm.items.get(i).getCTRLID().equals(4)) {
            theForm.items.get(i).obj = new XmlGuiLabel(this,
        theForm.items.get(i).getITMLABEL() +
        (theForm.items.get(i).getITMREQUI().equals("T") ? "*"
        : ""));
            ll.addView((View) theForm.items.get(i).obj);
        }

    Button btn = new Button(this);
    btn.setBackgroundResource(R.xml.custom_button_green);
    LayoutParams params = new LayoutParams(
            LayoutParams.MATCH_PARENT,
            LayoutParams.MATCH_PARENT
    );
    ll.addView(btn);
    params.setMargins(0,20,0,0);
    btn.setLayoutParams(params);

    btn.setText("Submit");
    btn.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
```

```
        // Chequea si los campos requeridos no están
        vacios
        if (!CheckForm())
        {
              AlertDialog.Builder bd = new
              AlertDialog.Builder(ll.getContext());
              AlertDialog ad = bd.create();
              ad.setTitle("Error");
              ad.setMessage("Ingresa los campos (*)");
              ad.show();
              return;

        }
        //Si hay URL para enviar datos
        if (!theForm.getFRMURL().equals("")) {
              //Guarda datos ingresados
              if (!SaveDataForm()) {
                    AlertDialog.Builder bd = new
              AlertDialog.Builder(ll.getContext());
                    AlertDialog ad = bd.create();
                    ad.setTitle("Error");
                    ad.setMessage("Error en guardar datos
                    ingresados");
                    ad.show();
                    return;
              }
        //Llama al resolutor
        if (!ResolForm.fnResolutor(theForm,context)) {
                    AlertDialog.Builder bd = new
              AlertDialog.Builder(ll.getContext());
                    AlertDialog ad = bd.create();
                    ad.setTitle("Error");
                          ad.setMessage("Error en
              resolución de datos por resolutor");
                    ad.show();
                    return;
              }
        }
        Intent newResult = new
Intent(RunForm.this,ActResult.class);

newResult.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
        Bundle extras = new Bundle();
        extras.putString("formResultCode",formCode);
        extras.putInt("version",0);
        extras.putParcelable("theForm",theForm);
        newResult.putExtras(extras);
```

```
                        startActivity(newResult);
                    finish();

                }
            } );
            setContentView(sv);
            setTitle(theForm.getFRMNAME());
            return true;
        } catch (Exception e) {
            Log.e(tag,"Error mostrando el
            formulario:"+e.toString());
            return false;
        }
}
```

- CheckForm: Revisa si los campos requeridos están vacíos o no.

```
private boolean CheckForm()
{
      try {
            int i;
            boolean good = true;

            for (i=0;i<theForm.items.size();i++) {
                  String itemValue = (String)
                  theForm.items.get(i).getData();

                  if
                  (theForm.items.get(i).getITMREQUI().equals("T"))
                  {
                        if (itemValue == null) {
                              good = false;
                        } else {
                              if (itemValue.trim().length() == 0) {
                                    good = false;
                              }
                        }
                  }
            }
            return good;
      } catch(Exception e) {
            Log.e(tag,"Error en el checkform" + e.getMessage());
            e.printStackTrace();
            return false;
      }
}
```

- SaveDataForm: Guarda datos ingresados en la clase CItemData.

```
private boolean SaveDataForm()
{
      int i;
      theForm.itemsData.clear() ;
      for (i=0;i<theForm.items.size();i++) {
            CItemData tempItemData =  new CItemData();
            tempItemData.setFRMID(theForm.getFRMID());

      tempItemData.setITMID(theForm.items.get(i).getITMID());
            tempItemData.setDITMVAL((String)
            theForm.items.get(i).getData());
            theForm.getItemsData().add(tempItemData);
        }
      return true;
}
```

- **ResolForm.java:** Esta clase se encarga de la resolución de las respuestas ingresadas por el usuario.

```
package com.tesis.app.form;
public class ResolForm {
      // Código (Métodos, funciones, etc.).
}
```

- fnResolutor: Constructor de la clase. Establece los tipos de resolución con que se tratarán los datos.

```
public static boolean fnResolutor(CForm theForm,Context context)
{
      float result = 0;
      String itemDatPnt="";
      Integer tsolid=0;
      Integer ctrlid=0;
      Integer ctrlNumeric=2;
      Integer ctrlChoice=3;
    String  type="";
    String strname="";
      int i;

        for (i=0;i<theForm.items.size();i++) {
           if(theForm.items.get(i).getITMSOLVE().equals("T"))
           {
                 tsolid=theForm.items.get(i).getTSOLID();
```

```
                    ctrlid = theForm.items.get(i).getCTRLID();

                    //Tipo de resolución
                    if(tsolid.equals(1)) //R. Simple
                    {
                         if(ctrlid.equals(ctrlNumeric))  type="SN";
                         if(ctrlid.equals(ctrlChoice))   type="SC";
                    }
                    if(tsolid.equals(2)) //R. Límites
                    {
                         if(ctrlid.equals(ctrlNumeric))  type="LN";
                    }

                    itemDatPnt=fnResolutor(type,theForm, i);
                    if(itemDatPnt.equals("0"))
                         theForm.itemsData.get(i).setDITMGOOD("F");
                    else
                         theForm.itemsData.get(i).setDITMGOOD("T");
                    //Agrega resultado a la clase contenedora de
                    datos ingresados
                    theForm.itemsData.get(i).setDITMPNT(itemDatPnt);
                    result = result + Float.valueOf(itemDatPnt);
               }
          }
               strname =
               fnGenerateResults(theForm,String.valueOf(result)); //
               Resultado segun las estrategias
          //Guarda Resultado en BD
          DBHandler dbh1 = new DBHandler (context);
          dbh1.resolForm_setItemsData(theForm);
     dbh1.resolForm_setResult(theForm, result, strname);
     dbh1.backupDB(context);
     dbh1.close();
          RunForm.resultEval = result;
          return true;
}
```

- fnResolutor: Resuelve y extrae el puntaje correspondiente.

```
public static String fnResolutor(String type,CForm theForm, int
i)
{
      String
      itemDatVal=theForm.itemsData.get(i).getDITMVAL().trim();
     String itemPnt="";
     int count=0;
```

```
        for( int j=0;j<theForm.items.get(i).val.size();j++ )
        //Tamaño de valores
       {
        String s = theForm.items.get(i).val.get(j).getVALNAME();
        if(type.equals("SN"))
              {
              if(itemDatVal.isEmpty())
                    return "0";
                    if(s.equals(itemDatVal))
                    return
             theForm.items.get(i).pnt.get(0).getPNTNAME();
              }

              if(type.equals("SC"))
              {
                    if(s.equals(itemDatVal))
                    {
                          int svalue = Integer.valueOf(s);
                          if(theForm.items.get(i).pnt.size()>1)
                              //Extrae el puntaje correspondiente
                        a la posición
                              itemPnt =
                    theForm.items.get(i).pnt.get(svalue).getPNTNAME(
                    );
                          else
                          itemPnt =
                    theForm.items.get(i).pnt.get(0).getPNTNAME();
                          return itemPnt;
                    }
               }

               if(type.equals("LN"))
               {
                    if(itemDatVal.isEmpty())
                          return "0";
                     if(fnEvalLimits(s,itemDatVal))
                     {
                           itemPnt =
                        theForm.items.get(i).pnt.get(count).getPNTNAME()
                        ;
                           return itemPnt;
                     }
                     count++;
               }
              }
        return "0";
}
```

- fnEvalLimits: Evalúa si el valor de ambos ítems está dentro de los límites establecidos.

```
private static boolean fnEvalLimits(String itemVal, String
itemDatVal)
{
      boolean dentroDeLimites=false;
       List<String> limits = Arrays.asList(itemVal.split("-"));
       float varx = Float.valueOf(limits.get(0));
       float vary = Float.valueOf(limits.get(1));
      float itemDatValf = Float.valueOf(itemDatVal); //Dato
      ingresado
       if(varx <= itemDatValf && itemDatValf <= vary)
       {
            dentroDeLimites = true;
       }
      return dentroDeLimites;
}
```

- fnGenerateResults: Evalúa el resultado final según estrategias.

```
private static String fnGenerateResults(CForm theForm ,String
result)
{
     for (int i=0;i<theForm.strategy.size();i++) {
          if(theForm.strategy.get(i).getSTRTSOL().equals("S"))
          {

if(theForm.strategy.get(i).getSTRVAL().equals(result))
                    return
theForm.strategy.get(i).getSTRNAME();
          }
          if(theForm.strategy.get(i).getSTRTSOL().equals("L"))
          {
          if(fnEvalLimits(theForm.strategy.get(i).getSTRVAL(),r
          esult))
                     return
theForm.strategy.get(i).getSTRNAME();
          }
        }
      return "";
}
```

- **SendForm.java:** Esta clase se carga los enviar los datos del formulario resuelto a la dirección especificada dentro del formulario.

```
package com.tesis.app.form;
public class SendForm extends Activity {
      // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
static String tag = SendForm.class.getSimpleName();
ProgressDialog progressDialog;
Handler progressHandler;
CForm theForm;
Context context = this;
ObjectMapper m = new ObjectMapper();
```

- onCreate: Constructor de la clase. Crea el diálogo de espera mientras transmite datos.

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_act_send);

      theForm = new CForm();
     Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        theForm = extras.getParcelable("theForm");

      final TextView text = (TextView)
      findViewById(R.id.textViewSend);

      try {
            //Dialogo de espera
             this.progressDialog = ProgressDialog.show(context,
            theForm.getFRMNAME(), "Saving Form Data",
            true,false);
             //Clase Handler es para compartir datos con
            diferentes hilos.
             this.progressHandler = new Handler() {

                  //Crea la clase handler para usarla luego
                 @Override
                 public void handleMessage(Message msg) {
                     switch (msg.what) {
                         case 0:
```

```
                progressDialog.setMessage("" + (String)
                msg.obj);
                            break;
                       case 1:
                            progressDialog.cancel();
                     text.setText("DATOS ENVIADOS
                     CORRECTAMENTE!");
                            break;
                       case 2:
                          progressDialog.cancel();
                          text.setText("INTENTE DE NUEVO.");
                          break;
                    }
                    super.handleMessage(msg);
                }

            };
            //----INSTANCIA UN THREAD PARA TRANSMITIR DATA
                Thread workthread = new Thread(new
                TransmitFormData(theForm));
            workthread.start();

      } catch (Exception e) {
           Log.e(tag,"Error in SubmitForm()::" +
e.getMessage());
            e.printStackTrace();
             Message msg = new Message();
             msg.what = 1;
             this.progressHandler.sendMessage(msg);
      }
}
```

- TransmitFormData: Conecta a la aplicación con la dirección del servidor descrito en el campo URL del formulario.

```
private class TransmitFormData implements Runnable
{
        CForm theForm = new CForm();
        Message msg;
        TransmitFormData(CForm form) {
            this.theForm = form;
        }

        //Clase que corre cuando se instanció un hilo
        public void run() {

            try {
```

```
String numEquipo =
Utilitario.fnNumEquipo(context);

msg = new Message();
msg.what = 0;
msg.obj = ("Conectando al Servidor");
progressHandler.sendMessage(msg);

URL url = new URL(theForm.getFRMURL());
URLConnection conn = url.openConnection();
conn.setDoOutput(true);

DataOutputStream dos = new
DataOutputStream(conn.getOutputStream());

dos.writeBytes(theForm.getFRMIDT()+"_"+theForm.g
etFRMID().toString()+"_"+theForm.result.get(0).g
etVERSION().toString()+":"+CMapper.toJson(theFor
m,true));
dos.flush();
msg = new Message();
msg.what = 0;
msg.obj = ("Data enviada");
progressHandler.sendMessage(msg);
InputStream is = conn.getInputStream();
// RECIBE RESPUESTA DEL SERVIDOR
int ch;

StringBuffer b =new StringBuffer();
while( ( ch = is.read() ) != -1 ){ b.append(
(char)ch ); }
String s = b.toString();

Boolean bSuccess = false;
if (s.indexOf("SUCCESS") != -1) {
    bSuccess = true;
}

dos.close();

if (bSuccess) {
    msg = new Message();
    msg.what = 0;
    msg.obj = ("Formulario enviado
      correctamente");
    progressHandler.sendMessage(msg);
```

```
                    msg = new Message();
                    msg.what = 1;
                    progressHandler.sendMessage(msg);
                    return;
                }
             } catch (Exception e) {
             Log.d(tag, "Falla al enviar data: " +
            e.getMessage());
                 msg = new Message();
                 msg.what = 0;
                 msg.obj = ("Error enviando datos");
                 progressHandler.sendMessage(msg);
             }
             msg = new Message();
             msg.what = 2;
             progressHandler.sendMessage(msg);
        }
}
```

- **HttpSincronizar.java:** Esta clase sirve para hacer la conexión con el servidor al sincronizar un archivo.

```
package com.tesis.app.http;
public class HttpSincronizar extends HttpConexion{
     // Código (Métodos, funciones, etc.).
}
```

- SubConHttp:Llama al manejador de la base de datos para sincronizar el formulario.

```
private void SubConHttp()
{
     if(!Utilitario.fnVerSignal(ioContext))//OffLine
     {
          ((ActGeneral)ioContext).setHttpResultado("No hay
     señal.");
          ((ActGeneral)ioContext).setIdHttpResultado(-1);
     }
     try {
               DBHandler dbh = new DBHandler (ioContext);
          boolean guardadoExitoso =
          dbh.sincForm_setForm(formIDT);
         dbh.close();
         if(guardadoExitoso){
          ((ActGeneral)ioContext).setHttpResultado(
          "Sincronización y guardado exitosos!");
```

```
                ((ActGeneral)ioContext).setIdHttpResultado(1);
                return;
            }
            else
            {
             ((ActGeneral)ioContext).setHttpResultado("No se puede
             sincronizar el formulario.");
                ((ActGeneral)ioContext).setIdHttpResultado(-1);
            }
            return;
        } catch (Exception e) {
             ((ActGeneral)ioContext).setHttpResultado("No se
            puede sincronizar el formulario.");
                ((ActGeneral)ioContext).setIdHttpResultado(-1);
           }
}
```

- fnSincronizar: Hace la conexión para la sincronización.

```
public static BufferedReader fnSincronizar(Context setcontext,
EnumUrl poEnumUrl, String formSincCode){
      URL url;
      BufferedReader in = null ;
      try {
            url = new URL(Configuracion.fnUrl(setcontext,
            poEnumUrl, formSincCode));
            in = new BufferedReader(new
            InputStreamReader(url.openStream()));
      } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }
      return in;
}
```

- **Configuracion.java:** Comprende más que todo las rutas de archivos que podremos configurar más adelante.

```
package com.tesis.app.utils;
public class Configuracion {
      // Código (Métodos, funciones, etc.).
}
```

- Atributos de la clase:

```
//-- ENUM URL
public static enum EnumUrl {LOGIN,SINCRONIZAR,SINCRONIZARFORM};
//-- LISTA OPCIONES MENU
public static Integer[] LstOpcMenu = {
          R.string.actMenu_Sincronizar,
          R.string.actMenu_Evaluar,
          R.string.actMenu_Resultados
};
//-- LISTA ICONOS MENU: Deben corresponder en orden con las opc.
de menu
public static Integer[] LstIcoMenu = {
          //R.drawable.ico_menu_sincro,
          R.drawable.draw_refresh,
          R.drawable.draw_evaluate,
        R.drawable.draw_result
};
```

- fnUrl: Asigna las rutas del servidor con el que se hará conexión y sus archivos.

```
public static String fnUrl(Context poContext, EnumUrl poEnumUrl
, String formSincCode)
{
      String lsUrl = "";
      String lsRuta = "http://" +
      poContext.getString(R.string.app_urlserver) +
      Constantes.RUTA;
      String lsRuta1 = "http://" +
      poContext.getString(R.string.app_urlserver) +
      Constantes.RUTA1;
      if(poEnumUrl == EnumUrl.LOGIN)
      {
            lsUrl = lsRuta + Constantes.LOGIN ;
      }
      else if(poEnumUrl == EnumUrl.SINCRONIZAR)
      {
            lsUrl = lsRuta1 + Constantes.SINCRONIZAR;
      }
      else if(poEnumUrl == EnumUrl.SINCRONIZARFORM)
      {
            lsUrl = lsRuta1 + formSincCode+ ".txt";
      }
      return lsUrl;
}
```

- **Utilitario.java:** En esta clase se encuentran los métodos utilitarios listos para ser utilizados.

```
package com.tesis.app.utils;
public class Utilitario {
      // Código (Métodos, funciones, etc.).
}
```

- fnVerSignal: Verifica si el dispositivo móvil cuenta con una señal wi-fi o 3g activa.

```
public static boolean fnVerSignal(Context poContext)
{
      boolean lbResultado;
      boolean lbCon3g=false;
      boolean lbconData=false;
      boolean lbWifi=false;

      TelephonyManager loTelephonyManager; //Para ver redes 3G
      ConnectivityManager loConnectivityManager; //Para ver
      conectividad en general
      NetworkInfo loNetworkInfo;  //Para ver la informacion de la
      red
      loConnectivityManager = (ConnectivityManager)
      poContext.getSystemService(Context.CONNECTIVITY_SERVICE);
      loTelephonyManager = (TelephonyManager)
      poContext.getSystemService(Context.TELEPHONY_SERVICE);

      if (loTelephonyManager.getDataState() ==
      TelephonyManager.DATA_CONNECTED) {
            lbCon3g=true;

      } else if (loTelephonyManager.getDataState() ==
      TelephonyManager.DATA_DISCONNECTED) {
            lbCon3g=false;
      }
      try
      {
            //Verifica data
            loNetworkInfo =
            loConnectivityManager.getActiveNetworkInfo();
            if (loNetworkInfo.isConnected()) {
                  lbconData=true;
            } else {
                  lbconData=false;
            }
```

```
        } catch (SecurityException e) {
              lbconData=false;
        } catch (NullPointerException e){
              lbconData=false;
        }
        try
        {
              //Verifica conexión Wifi
              NetworkInfo loNetworkInfo2 =
              loConnectivityManager.getNetworkInfo(ConnectivityMana
              ger.TYPE_WIFI);
              if (loNetworkInfo2.isConnected()) {
                    lbWifi=true;
              } else {
                    lbWifi=false;
              }
        } catch (SecurityException e) {
              lbWifi=false;
        } catch (NullPointerException e){
              lbWifi=false;
        }
        lbResultado=lbCon3g || lbconData || lbWifi;
        return lbResultado;
}
```

- fnVerStorage: Verifica si el dispositivo móvil posee memoria externa que tenga permisos para que se pueda leer y escribir en ella.
- fnNumEquipo: Obtiene el número de celular del dispositivo móvil.

```
public static String fnNumEquipo(Context poContext)
{
      String lsRespuesta;
      TelephonyManager loTelephonyManager
      =(TelephonyManager)poContext.getSystemService(Context.TELEP
      HONY_SERVICE);
      lsRespuesta = loTelephonyManager.getLine1Number();
      return lsRespuesta;
}
```

- fnRedondeo: Devuelve el valor ingresado redondeando a este a la precisión deseada.

```
public static double fnRedondeo(double pdValor, int piPrecision)
{
    BigDecimal bd = new BigDecimal(pdValor);
     BigDecimal rounded = bd.setScale(piPrecision,
     BigDecimal.ROUND_HALF_UP);
    return rounded.doubleValue();
}
```

- fnFechaFormato: Le da un formato definido a la fecha ingresada.

```
public static String fnFechaFormato(String psFecha)
{
     StringBuffer loSB=new StringBuffer();
     loSB.append(psFecha.substring(6,8));
     loSB.append("/");
     loSB.append(psFecha.substring(4,6));
     loSB.append("/");
     loSB.append(psFecha.substring(0,4));
     return loSB.toString();
}
```

- **Carpeta res/layout:** Un layout define la estructura visual de para determinada interfaz de usuario como una activity.
  - **activity_act_menu.xml:** Este layout está compuesto por una vista Grid que hace que los elementos cargados en el menú puedan ser correctamente distribuidos a lo largo de la pantalla del dispositivo móvil.

```
<?xml version="1.0" encoding="utf-8"?>
<GridView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/GridView01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/draw_bckcarbon"
    android:columnWidth="90dp"
    android:gravity="center"
    android:horizontalSpacing="10dp"
    android:numColumns="auto_fit"
    android:padding="10dp"
    android:stretchMode="columnWidth"
    android:verticalSpacing="10dp" >
</GridView>
```

- **activity_act_tipsinc.xml:** Este layout muestra un control EditText en el cuál se podrá ingresar el identificador del formulario que se quiere sincronizar.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    tools:context=".ActTipoRes" >

     <LinearLayout
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_centerInParent="true"
         android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:layout_marginLeft="10dp"
        android:text="Ingresa el código de Formulario:"

android:textAppearance="?android:attr/textAppearanceMedium" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal" >

        <EditText
            android:id="@+id/edtSincCodigo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:background="@xml/custom_edittext"
            android:inputType="text" >

        </EditText>
```

```
        <ImageView
            android:id="@+id/imageViewQR"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_gravity="center"
            android:layout_marginRight="10dp"
            android:src="@drawable/draw_qr" />

    </LinearLayout>

    <Button
        android:id="@+id/btnSincForm"
        style="@style/buttonstyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:background="@xml/custom_button_green"
        android:text="Obtener Formulario" />

     </LinearLayout>

</RelativeLayout>
```

- **activity_act_tipev.xml:** Permite elegir el tipo de formulario que queremos resolver.
- **activity_act_tipres.xml:** Nos muestra en una lista las resoluciones que ingresamos de todos los formularios y nos permite revisar estas respuestas en detalle.
- **activity_act_result.xml:** Este layout muestra los items evaluados indicando si la respuesta que ingresamos es correcta o no.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp" >

    <ImageView
        android:id="@+id/resultlogo"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginLeft="5px"
```

```
        android:layout_marginRight="20px"
        android:layout_marginTop="5px"
        android:src="@drawable/graph" >
    </ImageView>

    <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
         <TextView
             android:id="@+id/ritmlabel"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="Respuesta"
             android:textSize="20dp" >
         </TextView>

        <TextView
             android:id="@+id/ritmval"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="Respuesta"
             android:textSize="15dp" >
         </TextView>

        <TextView
             android:id="@+id/rditmval"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="Respuesta"
             android:textSize="15dp" >
         </TextView>

    </LinearLayout>
</LinearLayout>
```

- **activity_act_resultmain.xml:** Muestra el resultado final de determinada resolución de determinado formulario, incorpora en su vista la lista generada en “activity_act_result” dando la opción de desplazar arriba y abajo esta lista.
- **activity_act_send.xml:** Este layout nos indica si la resolución de formulario enviado al servidor obtuvo una respuesta favorable.

- **Carpeta res/values**

  - **strings.xml:** El fichero strings.xml permite definir y almacenar cadenas de texto (strings) utilizando el formato XML para su definición que pueden ser utilizadas en cualquier fichero xml o código android.
  - **styles.xml:** Fichero donde se define el formato y vista de la interfaz de usuario, puede ser aplicada a una vista individual desde un archivo layout o a la actividad entera y aplicación desde el Android Manifest.

```
<resources>

    <!--
        Base application theme, dependent on API level. This
theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer
devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">
        <!--
            Theme customizations available in newer API levels
can go in
            res/values-vXX/styles.xml,   while   customizations
related to
            backward-compatibility can go here.
        -->
    </style>

    <!-- Application theme. -->
    <style name="AppTheme" parent="AppBaseTheme">
        <!--  All  customizations  that  are  NOT  specific  to  a
particular API-level can go here. -->
    </style>

      <style name="buttonstyle">
      <item name="android:textStyle">bold</item>
      <item name="android:textColor">#ffffff</item>
      <item name="android:textSize">16sp</item>
      <item name="android:shadowColor" >#000000</item>
    <item name="android:shadowDx" >1</item>
    <item name="android:shadowDy" >1</item>
    <item name="android:shadowRadius" >2</item>
     </style>

</resources>
```

- **Carpeta res/xml**

  - **custom_button_green.xml:** En este archivo están definidos los estilos de un botón conforme al color de fondo, de línea y forma de éste.

```
<?xml version="1.0" encoding="utf-8" ?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_pressed="true" >
    <shape>
      <solid
          android:color="#94B52C" />
      <stroke
          android:width="1dp"
          android:color="#54661C" />
      <corners
          android:radius="6dp" />
      <padding
          android:left="10dp"
          android:top="10dp"
          android:right="10dp"
          android:bottom="10dp" />
    </shape>
  </item>
  <item>
    <shape>
      <gradient
          android:startColor="#94B52C"
          android:endColor="#54661C"
          android:angle="270" />
      <stroke
          android:width="1dp"
          android:color="#54661C" />
      <corners
          android:radius="6dp" />
      <padding
          android:left="10dp"
          android:top="10dp"
          android:right="10dp"
          android:bottom="10dp" />
    </shape>
  </item>
</selector>
```

- **custom_button_red.xml :** Tiene definidos los estilos de un botón conforme al color de fondo, de línea y forma de éste.
- **custom_edittext.xml:** Define los estilos de un control EditText.

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#EAF1D2"/>
    <stroke android:width="1dp" android:color="#54661C"/>
    <corners android:radius="3dp" />
    <padding android:left="10dp" android:top="10dp"
        android:right="10dp" android:bottom="10dp" />
</shape>
```

- **custom_spinner.xml:** Están establecidos los estilos de el control Spinner.

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <item><layer-list>
          <item><shape>
                              <gradient       android:angle="90"
          android:endColor="#EAF1D2"
          android:startColor="#7B8A4A" android:type="linear" />
                    <stroke                   android:width="1dp"
               android:color="#54661C" />
                    <corners android:radius="4dp" />
                              <padding       android:left="10dp"
          android:top="10dp"
        android:right="10dp" android:bottom="10dp" />
               </shape></item>
          <item     ><bitmap     android:gravity="bottom|right"
          android:src="@drawable/arrow_down" />
           </item>
        </layer-list></item>
</selector>
```

- **custom_spinnerlist.xml:** Define el estilo visual que se le podrá aplicar a la lista de un control Spinner.

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
```

```
    <solid android:color="#7B8A4A" />
     <corners
     android:bottomRightRadius="15dp"
     android:bottomLeftRadius="15dp"
  android:topLeftRadius="0dp"
  android:topRightRadius="0dp"/>
      <padding android:left="10dp" android:top="10dp"
        android:right="10dp" android:bottom="10dp" />
       <stroke android:width="1dp" android:color="#54661C" />
</shape>
```

ANEXO C

TABLAS Y PROCEDIMIENTOS ALMACENADOS DEL FRAMEWORK

1. INTRODUCCIÓN

El almacenamiento y acceso a datos durante el uso del framework es muy importante, es por ello que para el módulo web se utilizará SQL Server 2005 y para la aplicación móvil será SQLite, que es parte del Sistema Operativo Android.

2. NOMENCLATURA

- **Tablas de parametrización:** Las tablas que contengan datos de parametrización tendrán la nomenclatura TBM_[Entidad].
- **Tablas de datos:** Las tablas que contengan datos resultantes del uso final de la aplicación móvil se llamarán TBD_[Entidad]
- **Procedimientos Almacenados:** Para el caso de los procedimientos almacenados se tiene: SP[Web/Movil]_[Proceso al que pertenecen][Entidad][Operación].
- **Funciones:** Las funciones seguirán la siguiente nomenclatura: FX_[Operación].

3. TABLAS DEL MÓDULO WEB

TBM_USER: Permite almacenar y controlar usuarios que tendrán acceso al módulo web o a la aplicación móvil en caso de ser necesario, por defecto se registra a uno como administrador.

```
CREATE TABLE [dbo].[TBM_USER](
      [USRID] [bigint] IDENTITY(1,1) NOT NULL,
      [USRCOD] [varchar](10) NOT NULL,
      [USRPWD] [varchar](20) NOT NULL,
      [USRNAME] [varchar](100) NOT NULL,
      [USRPROF] [bigint] NOT NULL,
      [USRSTAT] [varchar](1) NOT NULL,
CONSTRAINT [PK_TBM_USER] PRIMARY KEY CLUSTERED
(
      [USRID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

INSERT INTO TBM_USER VALUES('ADMIN','1','ADMINISTRADOR','1','A')
```

TBM_PROFILE: Contiene los perfiles que puede tener un usuario, por defecto: "Administrador" o "Usuario Móvil".

```
CREATE TABLE [dbo].[TBM_PROFILE](
      [PRFID] [bigint] IDENTITY(1,1) NOT NULL,
      [PRFABR] [varchar](5) NOT NULL,
      [PRFNAME] [varchar](20) NOT NULL,
CONSTRAINT [PK_TBM_PROFILE] PRIMARY KEY CLUSTERED
(
      [PRFID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

INSERT INTO TBM_PROFILE VALUES('A','ADMINISTRADOR')
INSERT INTO TBM_PROFILE VALUES('U','USUARIO MOVIL')
```

TBM_FORM: Permite almacenar los formularios parametrizados por el usuario administrador.

```
CREATE TABLE [dbo].[TBM_FORM](
      [FRMID] [bigint] IDENTITY(1,1) NOT NULL,
      [FRMIDT] [varchar](50) NOT NULL,
      [FRMNAME] [varchar](50) NULL,
      [FRMURL] [varchar](100) NOT NULL,
      [FRMSTAT] [char](1) NOT NULL,
CONSTRAINT [PK_TBM_FORM] PRIMARY KEY CLUSTERED
(
      [FRMID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBM_ITEM: Permite almacenar los ítems pertenecientes a cada formulario.

```
CREATE TABLE [dbo].[TBM_ITEM](
      [ITMID] [bigint] IDENTITY(1,1) NOT NULL,
      [FRMID] [bigint] NOT NULL,
      [CTRLID] [bigint] NOT NULL,
      [ITMNAME] [varchar](20) NOT NULL,
      [ITMLABEL] [varchar](50) NULL,
      [ITMREQUI] [char](1) NOT NULL,
      [ITMSOLVE] [char](1) NOT NULL,
      [TSOLID] [bigint] NOT NULL,
      [ITMSTAT] [char](1) NOT NULL,
CONSTRAINT [PK_TBM_ITEM] PRIMARY KEY CLUSTERED
(
      [ITMID] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBM_ITMOPT: Permite almacenar la(s) opción(es) pertenecientes a cada item.

```
CREATE TABLE [dbo].[TBM_ITMOPT](
      [OPTID] [bigint] IDENTITY(1,1) NOT NULL,
      [ITMID] [bigint] NOT NULL,
      [OPTINDEX] [varchar](10) NULL,
      [OPTNAME] [varchar](50) NULL,
CONSTRAINT [PK_TBM_ITMOPT] PRIMARY KEY CLUSTERED
(
      [OPTID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBM_ITMVAL: Permite almacenar el/los valor(es) asociados a cada opción.

```
CREATE TABLE [dbo].[TBM_ITMVAL](
      [VALID] [bigint] IDENTITY(1,1) NOT NULL,
      [ITMID] [bigint] NOT NULL,
      [VALINDEX] [varchar](10) NULL,
      [VALNAME] [varchar](50) NULL,
CONSTRAINT [PK_TBM_ITMVAL] PRIMARY KEY CLUSTERED
(
      [VALID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBM_ITMPNT: Permite almacenar los puntajes asociados a cada opción.

```
CREATE TABLE [dbo].[TBM_ITMPNT](
      [PNTID] [bigint] IDENTITY(1,1) NOT NULL,
      [ITMID] [bigint] NOT NULL,
      [PNTINDEX] [varchar](10) NULL,
      [PNTNAME] [varchar](50) NULL,
CONSTRAINT [PK_TBM_ITMPNT] PRIMARY KEY CLUSTERED
(
      [PNTID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBM_CTRLTYPE: Contiene los nombres e identificadores de los controles gráficos que pueden asociarse a cada item, por defecto se tiene: Text, Numeric, Choice, Label.

```
CREATE TABLE [dbo].[TBM_CTRLTYPE](
      [CTRLID] [bigint] IDENTITY(1,1) NOT NULL,
      [CTRLNAME] [varchar](50) NOT NULL,
CONSTRAINT [PK_TBM_CTRLTYPE] PRIMARY KEY CLUSTERED
(
      [CTRLID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

INSERT INTO TBM_CTRLTYPE VALUES('TEXT')
INSERT INTO TBM_CTRLTYPE VALUES('NUMERIC')
INSERT INTO TBM_CTRLTYPE VALUES('CHOICE')
INSERT INTO TBM_CTRLTYPE VALUES('LABEL')
INSERT INTO TBM_CTRLTYPE VALUES('GPS')
```

TBM_TYSOL: Contiene los tipos de soluciones disponibles, por defecto se tiene: Simple y Límites.

```
CREATE TABLE [dbo].[TBM_TYSOL](
      [TSOLID] [bigint] IDENTITY(1,1) NOT NULL,
      [TSOLABR] [varchar](5) NULL,
      [TSOLNAME] [varchar](20) NULL,
CONSTRAINT [PK_TBM_TYSOL] PRIMARY KEY CLUSTERED
(
      [TSOLID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

INSERT INTO TBM_TYSOL VALUES('S', 'SIMPLE')
INSERT INTO TBM_TYSOL VALUES('L', 'LIMITES')
```

TBM_STRATEGY: Permite almacenar las estrategias asociadas a cada formulario, según el resultado que se obtenga.

```
CREATE TABLE [dbo].[TBM_STRATEGY](
      [STRID] [bigint] IDENTITY(1,1) NOT NULL,
      [FRMID] [bigint] NOT NULL,
      [STRNAME] [varchar](50) NULL,
      [STRLABEL] [varchar](200) NULL,
      [TSOLID] [bigint] NOT NULL,
      [STRVAL] [varchar](50) NULL,
```

```
      [STRSTAT] [char](1) NULL,
CONSTRAINT [PK_TBM_STRATEGY] PRIMARY KEY CLUSTERED
(
      [STRID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBD_ITEM: Permite almacenar las respuestas pertenecientes a cada ítem, estas respuestas son ingresadas por el usuario a través del dispositivo móvil.

```
CREATE TABLE [dbo].[TBD_ITEM](
      [DITMID] [bigint] IDENTITY(1,1) NOT NULL,
      [USRID] [bigint] NOT NULL,
      [FRMID] [bigint] NOT NULL,
      [ITMID] [bigint] NOT NULL,
      [DITMVAL] [varchar](50) NOT NULL,
      [DITMPNT] [varchar](50) NOT NULL,
CONSTRAINT [PK_TBD_ITEM] PRIMARY KEY CLUSTERED
(
      [DITMID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

TBD_RESULT: Permite almacenar los resultados generados por el dispositivo móvil en base a las respuestas ingresadas por el usuario.

```
CREATE TABLE [dbo].[TBD_RESULT](
      [RESID] [bigint] IDENTITY(1,1) NOT NULL,
      [USRID] [bigint] NOT NULL,
      [FRMID] [bigint] NOT NULL,
      [STRID] [bigint] NOT NULL,
      [RESVAL] [varchar](50) NULL,
CONSTRAINT [PK_TBD_RESULT] PRIMARY KEY CLUSTERED
(
      [RESID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

4. TABLAS DEL MÓDULO MÓVIL

TBM_USER: Almacena y controla a los usuarios que tendrán acceso al módulo móvil.

```
CREATE TABLE TBM_USER (USR_ID INTEGER PRIMARY KEY  AUTOINCREMENT
NOT NULL  UNIQUE , USRNAME VARCHAR, USRPWD VARCHAR, USRLASTS
DATETIME DEFAULT CURRENT_TIMESTAMP);
```

TBM_FORM: Permite almacenar los formularios sincronizados mediante la aplicación móvil.

```
CREATE TABLE TBM_FORM (FRMID INTEGER PRIMARY KEY  AUTOINCREMENT
NOT NULL  UNIQUE , FRMIDT VARCHAR, FRMVERSI INTEGER , FRMUSRID
VARCHAR, FRMNAME VARCHAR, FRMURL VARCHAR, FRMACTIV VARCHAR);
```

TBM_ITEM: Permite almacenar los ítems pertenecientes a cada formulario.

```
CREATE TABLE TBM_ITEM (ITMID INTEGER PRIMARY KEY  AUTOINCREMENT
NOT NULL  UNIQUE , FRMID INTEGER,CTRLID INTEGER, TSOLID INTEGER,
ITMNAME VARCHAR, ITMLABEL VARCHAR, ITMREQUI VARCHAR, ITMOPT
VARCHAR, ITMSOLVE VARCHAR, ITMVAL VARCHAR, ITMPNT VARCHAR);
```

TBM_CTRLTYPE: Contiene los nombres e identificadores de los controles gráficos que pueden asociarse a cada item, por defecto se tiene: Text, Numeric, Choice, Label.

```
CREATE TABLE TBM_CTRLTYPE (CTRLID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , CTRLNAME VARCHAR);
INSERT INTO TBM_CTRLTYPE VALUES(1,'text');
INSERT INTO TBM_CTRLTYPE VALUES(2,'numeric');
INSERT INTO TBM_CTRLTYPE VALUES(3,'choice');
INSERT INTO TBM_CTRLTYPE VALUES(4,'label');
INSERT INTO TBM_CTRLTYPE VALUES(5,'gps');
```

TBM_ITMOPT: Permite almacenar la(s) opción(es) pertenecientes a cada item.

```
CREATE TABLE TBM_ITMOPT (OPTID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , ITMID INTEGER, OPTINDEX
INTEGER,OPTNAME VARCHAR);
```

TBM_ITMVAL: Permite almacenar el/los valor(es) asociados a cada opción.

```
CREATE TABLE TBM_ITMVAL (VALID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , ITMID INTEGER, VALINDEX
INTEGER,VALNAME VARCHAR);
```

TBM_ITMPNT: Permite almacenar los puntajes asociados a cada opción.

```
CREATE TABLE TBM_ITMPNT (PNTID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , ITMID INTEGER, PNTINDEX
INTEGER, PNTNAME VARCHAR);
```

TBM_TYSOL: Contiene los tipos de soluciones disponibles, por defecto se tiene: Simple y Límites.

```
CREATE TABLE TBM_TYSOL (TSOLID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , TSOLABR VARCHAR, TSOLNAME
VARCHAR);
INSERT INTO TBM_TYSOL VALUES(1,'S','simple');
INSERT INTO TBM_TYSOL VALUES(2,'L','limites');
```

TBM_STRATEGY: Permite almacenar las estrategias con las cuales la aplicación móvil generará resultados finales.

```
CREATE TABLE TBM_STRATEGY (STRID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , FRMID INTEGER, STRNAME
VARCHAR, STRLABEL VARCHAR, STRTSOL VARCHAR, STRVAL VARCHAR);
```

TBD_ITEM: Permite almacenar las respuestas pertenecientes a cada ítem, estas respuestas son ingresadas por el usuario a través del dispositivo móvil.

```
CREATE TABLE TBD_ITEM (DITMID INTEGER PRIMARY KEY  AUTOINCREMENT
NOT NULL  UNIQUE , USRID INTEGER, FRMID INTEGER, VERSION INTEGER
, ITMID INTEGER, DITMVAL VARCHAR, DITMPNT VARCHAR, DITMGOOD
VARCHAR);
```

TBD_RESULT: Permite almacenar los resultados generados por el dispositivo móvil en base a las respuestas ingresadas por el usuario.

```
CREATE TABLE TBD_RESULT (RESID INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL  UNIQUE , USRID INTEGER, FRMID INTEGER,
VERSION INTEGER ,STRID INTEGER, RESVAL FLOAT, RESSAVE VARCHAR,
RESNAME VARCHAR);
```

5. PROCEDIMIENTOS ALMACENADOS DEL MÓDULO WEB

El usar procedimientos almacenados en una aplicación hace que esta sea más rápida y eficiente, así mismo pueden ser llamados varias veces desde la aplicación. Tienen mayor rendimiento que realizar las sentencias de manejo de datos directamente desde la aplicación y disminuye la escritura de código fuente ya que sólo se llama al procedimiento en unas pocas líneas y/o se le envían parámetros. Por consiguiente se puede ver las principales ventajas de utilizar procedimientos almacenados:

- **Gestión simplificada:** Los Procedimientos almacenados permiten que gran parte de la lógica del negocio se encuentre en la base de datos, simplificando la gestión de datos y reduciendo la necesidad de codificar la lógica en el resto de los programas cliente. Así mismo los programadores de una aplicación pueden ejecutar la llamada a un procedimiento almacenado conociendo los parámetros que éste requiere sin necesidad de conocer la lógica de dicho procedimiento.

- **Seguridad:** Facilitan algunas tareas de administración de seguridad y asignación de permisos.

- **Centralización de la definición:** Al formar parte de la base de datos los procedimientos almacenados están en un lugar centralizado y pueden ser ejecutados por cualquier aplicación que tenga acceso a la misma. Si un determinado procedimiento almacenado es programado en una aplicación, es posible que no esté disponible en todos los lugares que se lo necesite y si se modifican no habrá necesidad de hacerlo en todas las aplicaciones que accedan a él.

- **Reducción del tráfico de red:** Una sentencia formada por muchas líneas de código SQL puede escribirse como un procedimiento almacenado en el servidor y ejecutarse simplemente mediante el nombre de dicho procedimiento, en lugar de enviar todas las líneas de código por la red desde el cliente hasta el servidor. [GUL03]

Procedimiento SPS_LoginWeb: Lista a un usuario según su identificador y password. Funciona para el ingreso al módulo web.

```
CREATE PROCEDURE [dbo].[SPS_LoginWeb]
@IN_USER AS VARCHAR(20),
@IN_PASS AS VARCHAR(100)
AS

BEGIN
SET NOCOUNT ON

SELECT USRCOD,
        USRNAME AS USRDSC,
        USRPROF AS USRPERFIL
FROM TBM_USER
```

```
WHERE USRCOD = @IN_USER
AND USRPWD = @IN_PASS
AND USRPROF = 1
END
```

Procedimiento SPM_Login: Lista a un usuario móvil según su identificador y password. Funciona para el ingreso a la aplicación móvil en caso de ser necesario.

```
CREATE PROCEDURE [dbo].[SPM_Login]
@PAR_USRCOD AS VARCHAR(20),
@PAR_USRPWD AS VARCHAR(100)
AS

BEGIN
SET NOCOUNT ON

DECLARE @RPTA INT
DECLARE @EXISTE INT

SELECT @EXISTE = COUNT(*)
FROM TBM_USER
WHERE USRCOD = @PAR_USRCOD
AND USRPWD = @PAR_USRPWD

IF (@EXISTE = 0)
BEGIN
      SELECT @EXISTE = COUNT(*)
      FROM TBM_USER
      WHERE USRCOD = @PAR_USRCOD

      IF(@EXISTE = 0)
            SET @RPTA = -1
      ELSE
            SET @RPTA = 0
END

ELSE
BEGIN
      SET @RPTA = 1
      SELECT @RPTA AS RPTA,
                  USRID,
                  USRCOD,
                  USRNAME AS USRDSC,
                  USRPWD
      FROM TBM_USER
      WHERE USRCOD = @PAR_USRCOD
      AND USRPWD = @PAR_USRPWD
END
IF (@RPTA <= 0)
BEGIN
     SELECT @RPTA AS RPTA
END
END
```

Procedimiento SPS_MTUserSel: Lista usuarios de acuerdo a los parámetros enviados.

```
CREATE PROCEDURE [dbo].[SPS_MTUserSel]
(
      @USR_COD VARCHAR(100),
      @USR_NAME VARCHAR(100),
      @USR_PROF VARCHAR(20),
      @USR_STAT VARCHAR(20)
)
AS
DECLARE @TAMTOTAL INT
BEGIN
SET NOCOUNT ON

DECLARE @TMPLISTADO TABLE(
      USRID VARCHAR(100) ,
      USRCOD VARCHAR(20),
      USRNAME VARCHAR(100),
      USRPROF VARCHAR(20),
      ROW_NUMBER VARCHAR(5),
      NORDEN INT IDENTITY(1,1)
)

INSERT INTO @TMPLISTADO
SELECT      USRID, USRCOD,
            USRNAME,
            P.PRFNAME,
            '0' AS ROW_NUMBER
FROM TBM_USER U
INNER JOIN TBM_PROFILE P ON U.USRPROF = P.PRFID
WHERE ((@USR_COD = '') OR (@USR_COD <> '' AND U.USRCOD =
@USR_COD))
AND (U.USRNAME LIKE '%' + @USR_NAME + '%')
AND ((@USR_PROF = '') OR (@USR_PROF <> '' AND U.USRPROF =
@USR_PROF))
AND U.USRSTAT = @USR_STAT
ORDER BY U.USRID ASC;

SET @TAMTOTAL = @@ROWCOUNT

SELECT      USRID,
            USRCOD,
            USRNAME,
            USRPROF,
            ROW_NUMBER,
            @TAMTOTAL AS TAMANIOTOTAL
FROM @TMPLISTADO
END
```

Procedimiento SPS_MTUserSelObj: Lista un usuario específico utilizando su identificador único.

```
CREATE PROCEDURE [dbo].[SPS_MTUserSelObj]
(
      @USR_ID INT
)
AS
BEGIN
      SELECT * FROM TBM_USER
      WHERE USRID = @USR_ID;
END
```

Procedimiento SPS_MTUserInsUpd: Inserta o Actualiza los datos de los usuarios.

```
CREATE PROCEDURE [dbo].[SPS_MTUserInsUpd]
(
      @USR_ID VARCHAR(20),
      @USR_COD VARCHAR(20),
      @USR_NAME VARCHAR(30),
      @USR_PWD VARCHAR(10),
      @USR_PROF VARCHAR(10),
      @USR_STAT VARCHAR(20)
)
AS
BEGIN

DECLARE @EXISTE INT
SET @EXISTE = 0
SET NOCOUNT ON

IF(@USR_ID = '' OR @USR_ID = '0')
BEGIN
  SELECT @EXISTE = COUNT(*) FROM TBM_USER WHERE USRCOD =@USR_COD

  IF @EXISTE > 0 BEGIN RAISERROR('LOGIN REPETIDO',18,1) END

  IF @EXISTE <= 0
  BEGIN
      INSERT INTO TBM_USER
(USRCOD,USRNAME,USRPWD,USRPROF,USRSTAT)
      VALUES(@USR_COD, @USR_NAME, @USR_PWD, @USR_PROF,
@USR_STAT);
  END
END

ELSE
BEGIN
  UPDATE TBM_USER
  SET USRCOD = @USR_COD,
       USRNAME = @USR_NAME,
        USRPWD = @USR_PWD,
```

```
      USRPROF = @USR_PROF,
      USRSTAT = @USR_STAT
  WHERE USRID = @USR_ID;
END
END
```

Procedimiento SPS_MTUserDel: Inactiva usuarios.

```
CREATE PROCEDURE [dbo].[SPS_MTUserDel]
(
     @IDS VARCHAR(200)
)
AS
BEGIN

UPDATE TBM_USER
SET USRSTAT = 'I'
WHERE USRID IN (SELECT SPLITVALUE FROM DBO.FX_Split(@IDS,'|') )
;

END
```

Procedimiento SPS_MTProfileSel: Lista todos los perfiles.

```
CREATE  PROCEDURE [dbo].[SPS_MTProfileSel]
AS
BEGIN

SELECT  PRFID as PER_ABREVIA,
        PRFNAME as PER_NOMBRE
FROM TBM_PROFILE
ORDER BY PRFID

END
```

Procedimiento SPS_MTFormSel: Lista formularios según los parámetros enviados.

```
CREATE PROCEDURE [dbo].[SPS_MTFormSel]
(
     @FRM_IDT VARCHAR(100),
     @FRM_NOMBRE VARCHAR(100),
     @FRM_URL VARCHAR(100),
     @FRM_STAT VARCHAR(100)
)
AS

DECLARE @TAMTOTAL INT

BEGIN

SET NOCOUNT ON
```

```
DECLARE @TMPLISTADO TABLE(
      FRMID VARCHAR(100),
      FRMIDT VARCHAR(100) ,
      FRMNAME VARCHAR(100),
      FRMURL VARCHAR(100),
      ROW_NUMBER VARCHAR(5),
      NORDEN INT IDENTITY(1,1)
)

INSERT INTO @TMPLISTADO

SELECT     FRMID,
           FRMIDT,
           FRMNAME,
           FRMURL,
           '0' AS ROW_NUMBER
FROM TBM_FORM F
WHERE (F.FRMIDT LIKE '%' + @FRM_IDT + '%')
AND   (F.FRMNAME LIKE '%' + @FRM_NOMBRE + '%')
AND   (F.FRMURL LIKE '%' + @FRM_URL + '%')
AND    (F.FRMSTAT = @FRM_STAT)
ORDER BY F.FRMID ASC;

SET @TAMTOTAL = @@ROWCOUNT

SELECT     FRMID,
           FRMIDT,
           FRMNAME,
           FRMURL,
           ROW_NUMBER,
           @TAMTOTAL AS TAMANIOTOTAL
FROM @TMPLISTADO

END
```

Procedimiento SPS_MTFormSelObj: Lista un formulario específico utilizando su identificador único.

```
CREATE PROCEDURE [dbo].[SPS_MTFormSelObj]
(
      @FRM_ID INT
)
AS
BEGIN
      SELECT * FROM TBM_FORM
      WHERE FRMID = @FRM_ID;
END
```

Procedimiento SPS_MTFormInsUpd: Inserta o Actualiza los datos de los formularios.

```
CREATE PROCEDURE [dbo].[SPS_MTFormInsUpd]
```

```
(
  @FRM_ID VARCHAR(10),
  @FRM_IDT VARCHAR(50),
  @FRM_NOMBRE VARCHAR(50),
  @FRM_URL VARCHAR(100),
  @FRM_STAT VARCHAR(50)
)
AS
BEGIN

SET NOCOUNT ON

IF(@FRM_ID = '' OR @FRM_ID = '0')
BEGIN
      INSERT INTO TBM_FORM(FRMIDT, FRMNAME, FRMURL, FRMSTAT)
      VALUES(@FRM_IDT, @FRM_NOMBRE, @FRM_URL, @FRM_STAT);
END

ELSE
BEGIN
      UPDATE TBM_FORM
      SET FRMNAME = @FRM_NOMBRE,
            FRMURL = @FRM_URL,
            FRMSTAT = @FRM_STAT
      WHERE FRMID = @FRM_ID
      AND FRMIDT = @FRM_IDT;
END

END
```

Procedimiento SPS_MTFormDel: Inactiva formularios.

```
CREATE PROCEDURE [dbo].[SPS_MTFormDel]
(
      @IDS VARCHAR(200)
)
AS
BEGIN

UPDATE TBM_FORM
SET FRMSTAT = 'I'
WHERE FRMID IN (SELECT SPLITVALUE FROM DBO.FX_Split(@IDS,'|') )
;

END
```

Procedimiento SPS_MTFormLstSel: Lista todos los formularios.

```
CREATE  PROCEDURE [dbo].[SPS_MTFormLstSel]
AS
BEGIN

SELECT  FRMID,
```

```
          FRMNAME
FROM TBM_FORM
ORDER BY FRMID

END
```

Procedimiento SPS_MTItemSel: Lista ítems según los parámetros enviados.

```
CREATE PROCEDURE [dbo].[SPS_MTItemSel]
(
     @ITM_NAME VARCHAR(100),
     @ITM_LABEL VARCHAR(100),
     @ITM_FORM VARCHAR(100),
     @ITM_CTRL VARCHAR(100),
     @ITM_TSOL VARCHAR(100),
     @ITM_REQ VARCHAR(100),
     @ITM_SOLVE VARCHAR(100),
     @ITM_STAT VARCHAR(100)
)
AS

DECLARE @TAMTOTAL INT

BEGIN

SET NOCOUNT ON

DECLARE @TMPLISTADO TABLE(
     ITMID VARCHAR(100),
     FRMNAME VARCHAR(100) ,
     CTRLNAME VARCHAR(100),
     ITMNAME VARCHAR(100),
     ITMLABEL VARCHAR(100),
     ITMSOLVE VARCHAR(100),
     TSOLNAME VARCHAR(100),
     ROW_NUMBER VARCHAR(5),
     NORDEN INT IDENTITY(1,1)
)

INSERT INTO @TMPLISTADO

SELECT     I.ITMID,
           F.FRMNAME,
           C.CTRLNAME,
           ITMNAME,
           ITMLABEL,
           ITMSOLVE,
           T.TSOLNAME,
           '0' AS ROW_NUMBER
FROM TBM_ITEM I
INNER JOIN TBM_FORM F ON F.FRMID = I.FRMID
INNER JOIN TBM_CTRLTYPE C ON C.CTRLID = I.CTRLID
LEFT JOIN TBM_TYSOL T ON T.TSOLID = I.TSOLID
```

```
WHERE (I.ITMNAME LIKE '%' + @ITM_NAME + '%')
AND (I.ITMLABEL LIKE '%' + @ITM_LABEL + '%')
AND ((@ITM_FORM = '') OR (@ITM_FORM <> '' AND F.FRMID =
@ITM_FORM))
AND ((@ITM_CTRL = '') OR (@ITM_CTRL <> '' AND C.CTRLID =
@ITM_CTRL))
AND ((@ITM_TSOL = '') OR (@ITM_TSOL <> '' AND T.TSOLID =
@ITM_TSOL))
AND (I.ITMREQUI = @ITM_REQ)
AND (I.ITMSOLVE = @ITM_SOLVE)
AND (I.ITMSTAT = @ITM_STAT)
ORDER BY F.FRMID ASC;

SET @TAMTOTAL = @@ROWCOUNT

SELECT     ITMID,
           FRMNAME,
           CTRLNAME,
           ITMNAME,
           ITMLABEL,
           ITMSOLVE,
           ISNULL(TSOLNAME,'NINGUNA') AS TSOLNAME,
           ROW_NUMBER,
           @TAMTOTAL AS TAMANIOTOTAL
FROM @TMPLISTADO

END
```

Procedimiento SPS_MTItemSelObj: Lista un ítem específico utilizando su identificador único.

```
CREATE PROCEDURE [dbo].[SPS_MTItemSelObj]
(
      @ITM_ID INT
)
AS
BEGIN
      SELECT * FROM TBM_ITEM
      WHERE ITMID = @ITM_ID;
END
```

Procedimiento SPS_MTItemInsUpd: Inserta o Actualiza los datos de los ítems.

```
CREATE PROCEDURE [dbo].[SPS_MTItemInsUpd]
(
      @ITM_ID VARCHAR(10),
      @FRM_ID VARCHAR(10),
      @CTRL_ID VARCHAR(10),
      @ITM_NAME VARCHAR(50),
      @ITM_LABEL VARCHAR(200),
      @ITM_REQUI VARCHAR(1),
      @ITM_SOLVE VARCHAR(1),
      @ITM_OPTIONS VARCHAR(1000),
```

```
      @ITM_VALUES VARCHAR(1000),
      @ITM_POINTS VARCHAR(1000),
      @TSOL_ID VARCHAR(10),
      @ITM_STAT VARCHAR(1)
)
AS
BEGIN
SET NOCOUNT ON
IF(@ITM_ID = '' OR @ITM_ID = '0')
BEGIN
      DECLARE @LST_ITM_ID INT

      INSERT INTO TBM_ITEM(FRMID, CTRLID, ITMNAME, ITMLABEL,
ITMREQUI, ITMSOLVE, TSOLID, ITMSTAT)
      VALUES(@FRM_ID, @CTRL_ID, @ITM_NAME, @ITM_LABEL,
@ITM_REQUI, @ITM_SOLVE, @TSOL_ID, @ITM_STAT);

      SET @LST_ITM_ID = (SELECT TOP(1) ITMID FROM TBM_ITEM ORDER
BY ITMID DESC)
      EXEC SPS_MTOptionInsUpd @LST_ITM_ID, @ITM_OPTIONS;
      EXEC SPS_MTValueInsUpd @LST_ITM_ID, @ITM_VALUES;
      EXEC SPS_MTPointInsUpd @LST_ITM_ID, @ITM_POINTS;
END
ELSE
BEGIN
      UPDATE TBM_ITEM
      SET FRMID = @FRM_ID,
            CTRLID = @CTRL_ID,
            ITMNAME = @ITM_NAME,
            ITMLABEL = @ITM_LABEL,
            ITMREQUI = @ITM_REQUI,
            ITMSOLVE = @ITM_SOLVE,
            TSOLID = @TSOL_ID,
            ITMSTAT = @ITM_STAT
      WHERE ITMID = @ITM_ID;
END
END
```

Procedimiento SPS_MTItemDel: Inactiva ítems.

```
CREATE PROCEDURE [dbo].[SPS_MTItemDel]
(
      @IDS VARCHAR(200)
)
AS
BEGIN

UPDATE TBM_ITEM
SET ITMSTAT = 'I'
WHERE ITMID IN (SELECT SPLITVALUE FROM DBO.FX_Split(@IDS,'|') )
;

END
```

Procedimiento SPS_MTOptionInsUpd: Inserta o Actualiza los datos de las opciones.

```
CREATE PROCEDURE [dbo].[SPS_MTOptionInsUpd]
(
      @ITM_ID VARCHAR(10),
      @ITM_OPTIONS VARCHAR(1000)
)
AS
BEGIN

SET NOCOUNT ON

INSERT INTO TBM_ITMOPT
SELECT @ITM_ID, (OCCURENCEID-1), SPLITVALUE
FROM DBO.FX_Split(@ITM_OPTIONS,'|')

END
```

Procedimiento SPS_MTValueInsUpd: Inserta o Actualiza los datos de los valores.

```
CREATE PROCEDURE [dbo].[SPS_MTValueInsUpd]
(
      @ITM_ID VARCHAR(10),
      @ITM_VALUES VARCHAR(1000)
)
AS
BEGIN
SET NOCOUNT ON

INSERT INTO TBM_ITMVAL
SELECT @ITM_ID, (OCCURENCEID-1), SPLITVALUE
FROM DBO.FX_Split(@ITM_VALUES,'|')

END
```

Procedimiento SPS_MTPointInsUpd: Inserta o Actualiza los datos de los puntajes.

```
CREATE PROCEDURE [dbo].[SPS_MTPointInsUpd]
(
      @ITM_ID VARCHAR(10),
      @ITM_POINTS VARCHAR(1000)
)
AS
BEGIN

SET NOCOUNT ON

INSERT INTO TBM_ITMPNT
SELECT @ITM_ID, (OCCURENCEID-1), SPLITVALUE
```

```
FROM DBO.FX_Split(@ITM_POINTS,'|')

END
```

Procedimiento SPS_MTStrategySel: Lista estrategias según los parámetros enviados.

```
CREATE PROCEDURE [dbo].[SPS_MTStrategySel]
(
      @STR_FORM VARCHAR(100),
      @STR_NAME VARCHAR(100),
      @STR_LABEL VARCHAR(100),
      @STR_TSOL VARCHAR(100),
      @STR_STAT VARCHAR(100)
)
AS

DECLARE @TAMTOTAL INT

BEGIN

SET NOCOUNT ON

DECLARE @TMPLISTADO TABLE(
      STRID VARCHAR(100),
      FRMNAME VARCHAR(100),
      STRNAME VARCHAR(100) ,
      STRLABEL VARCHAR(100),
      TSOLNAME VARCHAR(100),
      STRVAL VARCHAR(100),
      ROW_NUMBER VARCHAR(5),
      NORDEN INT IDENTITY(1,1)
)

INSERT INTO @TMPLISTADO

SELECT      STRID,
            FRMNAME,
            STRNAME,
            STRLABEL,
            TSOLNAME,
            STRVAL,
            '0' AS ROW_NUMBER
FROM TBM_STRATEGY S
INNER JOIN TBM_FORM F ON F.FRMID = S.FRMID
INNER JOIN TBM_TYSOL T ON T.TSOLID = S.TSOLID
WHERE ((@STR_FORM = '') OR (@STR_FORM <> '' AND S.FRMID =
@STR_FORM))
AND   ((@STR_TSOL = '') OR (@STR_TSOL <> '' AND S.TSOLID =
@STR_TSOL))
AND   (STRNAME LIKE '%' + @STR_NAME + '%')
AND   (STRLABEL LIKE '%' + @STR_LABEL + '%')
AND    (STRSTAT = @STR_STAT)
```

```
ORDER BY STRID ASC;

SET @TAMTOTAL = @@ROWCOUNT

SELECT     STRID,
           FRMNAME,
           STRNAME,
           STRLABEL,
           TSOLNAME,
           STRVAL,
           ROW_NUMBER,
           @TAMTOTAL AS TAMANIOTOTAL
FROM @TMPLISTADO

END
```

Procedimiento SPS_MTStrategySelObj: Lista una estrategia específica utilizando su identificador único.

```
CREATE PROCEDURE [dbo].[SPS_MTStrategySelObj]
(
     @STR_ID INT
)
AS
BEGIN
     SELECT * FROM TBM_STRATEGY
     WHERE STRID = @STR_ID;
END
```

Procedimiento SPS_MTStrategyInsUpd: Inserta o Actualiza los datos de los estrategias.

```
CREATE PROCEDURE [dbo].[SPS_MTStrategyInsUpd]
(
  @STR_ID VARCHAR(10),
  @STR_FORM VARCHAR(50),
  @STR_NAME VARCHAR(50),
  @STR_LABEL VARCHAR(100),
  @STR_TSOL VARCHAR(100),
  @STR_VAL VARCHAR(100),
  @STR_STAT VARCHAR(50)
)
AS
BEGIN
SET NOCOUNT ON
IF(@STR_ID = '' OR @STR_ID = '0')
BEGIN
     INSERT INTO TBM_STRATEGY(FRMID, STRNAME, STRLABEL, TSOLID,
STRVAL, STRSTAT)
     VALUES(@STR_FORM, @STR_NAME, @STR_LABEL, @STR_TSOL,
@STR_VAL, @STR_STAT);
END
ELSE
```

```
BEGIN
      UPDATE TBM_STRATEGY
      SET FRMID = @STR_FORM,
            STRNAME = @STR_NAME,
            STRLABEL = @STR_LABEL,
            TSOLID = @STR_TSOL,
            STRVAL = @STR_VAL,
            STRSTAT = @STR_STAT
      WHERE STRID = @STR_ID
END
END
```

Procedimiento SPS_MTStrategyDel: Inactiva estrategias.

```
CREATE PROCEDURE [dbo].[SPS_MTStrategyDel]
(
      @IDS VARCHAR(200)
)
AS
BEGIN

UPDATE TBM_STRATEGY
SET STRSTAT = 'I'
WHERE STRID IN (SELECT SPLITVALUE FROM DBO.FX_Split(@IDS,'|') )
;
END
```

Procedimiento SPS_MTCtrlTypeLstSel: Lista todos los controles disponibles.

```
CREATE  PROCEDURE [dbo].[SPS_MTCtrlTypeLstSel]
AS
BEGIN

SELECT  CTRLID,
            CTRLNAME
FROM TBM_CTRLTYPE
ORDER BY CTRLID

END
```

Procedimiento SPS_MTTypeSolLstSel: Lista todos los tipos de soluciones disponibles.

```
CREATE  PROCEDURE [dbo].[SPS_MTTypeSolLstSel]
AS
BEGIN
SELECT  TSOLID,
            TSOLNAME
FROM TBM_TYSOL
ORDER BY TSOLID

END
```

Procedimiento SPM_GenerateScript: Lista un formulario, sus items, opciones, valores, puntajes y estrategias respectivas a fin de ser ejecutadas en el dispositivo móvil.

```
CREATE PROCEDURE [dbo].[SPM_GenerateScript]
(
      @FRM_ID INT,
      @FRM_IDT VARCHAR(10)
)
AS
BEGIN

DECLARE @TB_TEMP TABLE (
      SCRIPT VARCHAR(5000),
      ITMID INT,
      ORDEN INT
)

INSERT INTO @TB_TEMP
SELECT 'UPDATE TBM_FORM SET FRMACTIV =''F'' WHERE FRMIDT=''' +
@FRM_IDT + '''' +
' AND FRMVERSI = (SELECT COUNT(*) FROM TBM_FORM WHERE FRMIDT =
''' + @FRM_IDT + '''' + ')' +
' AND 0 < (SELECT COUNT(*) FROM TBM_FORM WHERE FRMIDT = ''' +
@FRM_IDT + '''' + ');',
0 AS ITMID,
0 AS ORDEN

INSERT INTO @TB_TEMP
SELECT 'INSERT INTO TBM_FORM (FRMIDT, FRMVERSI,FRMNAME, FRMURL,
FRMACTIV) VALUES (' +
'''' + FRMIDT  + '''' + ',' +
'(SELECT COUNT(*) + 1 FROM TBM_FORM WHERE FRMIDT = ''' +
CONVERT(VARCHAR,@FRM_ID) + '''' + '),' +
'''' + FRMNAME + '''' + ',' +
'''' + FRMURL  + '''' + ',' +
'''' + FRMSTAT + '''' + ');',
0 AS ITMID,
0 AS ORDEN
FROM TBM_FORM
WHERE FRMID = @FRM_ID

INSERT INTO @TB_TEMP
SELECT 'INSERT INTO TBM_ITEM
(FRMID,CTRLID,TSOLID,ITMNAME,ITMLABEL,ITMREQUI,ITMSOLVE) VALUES
(' + '(SELECT FRMID FROM TBM_FORM ORDER BY FRMID DESC LIMIT 1),'
+
CONVERT(VARCHAR,CTRLID) + ',' +
CONVERT(VARCHAR,TSOLID) + ',' +
'''' + ITMNAME  + '''' + ',' +
'''' + ITMLABEL + '''' + ',' +
'''' + ITMREQUI + '''' + ',' +
```

```
'''' + ITMSOLVE + '''' + ');',
ITMID,
1 AS ORDEN
FROM TBM_ITEM
WHERE FRMID = @FRM_ID

INSERT INTO @TB_TEMP
SELECT 'INSERT INTO TBM_ITMOPT (ITMID, OPTINDEX, OPTNAME) VALUES
(' + '(SELECT ITMID FROM TBM_ITEM ORDER BY ITMID DESC LIMIT 1)'
+ ',' + '''' + OPTINDEX + '''' + ',' +
'''' + OPTNAME  + '''' + ');',
O.ITMID,
2 AS ORDEN
FROM TBM_ITMOPT O
INNER JOIN TBM_ITEM I ON I.ITMID = O.ITMID
WHERE I.FRMID = @FRM_ID



INSERT INTO @TB_TEMP
SELECT 'INSERT INTO TBM_ITMVAL (ITMID, VALINDEX, VALNAME) VALUES
(' + '(SELECT ITMID FROM TBM_ITEM ORDER BY ITMID DESC LIMIT 1)'
+ ',' + '''' + VALINDEX + '''' + ',' +
'''' + VALNAME  + '''' + ');',
V.ITMID,
3 AS ORDEN
FROM TBM_ITMVAL V
INNER JOIN TBM_ITEM I ON I.ITMID = V.ITMID
WHERE I.FRMID = @FRM_ID

INSERT INTO @TB_TEMP
SELECT 'INSERT INTO TBM_ITMPNT (ITMID, PNTINDEX, PNTNAME) VALUES
(' + '(SELECT ITMID FROM TBM_ITEM ORDER BY ITMID DESC LIMIT 1)'
+ ',' + '''' + PNTINDEX + '''' + ',' +
'''' + PNTNAME  + '''' + ');',
P.ITMID,
4 AS ORDEN
FROM TBM_ITMPNT P
INNER JOIN TBM_ITEM I ON I.ITMID = P.ITMID
WHERE I.FRMID = @FRM_ID

SELECT SCRIPT, ITMID, ORDEN FROM @TB_TEMP
ORDER BY ITMID, ORDEN ASC

END
```

6. FUNCIONES DEL MÓDULO WEB

Función FX_Split: Emula la función split, genera una tabla con los datos enviados, separados por el parámetro @Delimiter.

```
CREATE  FUNCTION [dbo].[FX_Split]
(
        @String VARCHAR(4000),
        @Delimiter VARCHAR(5)
)
RETURNS @SplittedValues TABLE
(
  OccurenceId SMALLINT IDENTITY(1,1),
  SplitValue VARCHAR(2000)
)
AS
BEGIN
DECLARE @SplitLength INT
WHILE LEN(@String) > 0
BEGIN
        SELECT @SplitLength = (CASE
CHARINDEX(@Delimiter,@String) WHEN 0 THEN LEN(@String)

                        ELSE CHARINDEX(@Delimiter,@String) -1
END)
        INSERT INTO @SplittedValues
        SELECT SUBSTRING(@String,1,@SplitLength)
        SELECT @String = (CASE (LEN(@String) - @SplitLength)
WHEN 0 THEN ''
ELSE RIGHT(@String, LEN(@String) - @SplitLength - 1) END)
END
RETURN
END
```

ANEXO D

CUESTIONARIO

Previo un atento saludo, le invitamos a responder la presente evaluación. Sus impresiones y opiniones servirán para evaluar y mejorar la propuesta.
Marque con una X su respuesta.

1. **¿El Framework propuesto facilita y agiliza la implementación de aplicaciones para adquisición de datos?**

Si		No	

2. **¿Cómo calificaría la organización de los paquetes y archivos del Framework?**

Mala		Regular		Buena	

3. **¿Considera que es difícil el manejo, entendimiento estructural y funcional del Framework?**

Si		No	

4. **¿Considera que los controles vistos en la aplicación móvil demuestran correctamente las propiedades de los controles parametrizados desde el módulo web del Framework?**

Si		No	

5. **Considera que integrar un nuevo control para la toma de datos es:**

Difícil		Regular		Fácil	

6. **Considera que el grado de reutilización de los controles existentes en el Framework es:**

Mala		Regular		Buena	

7. **¿Cómo calificaría la creación de nuevos controles en base al modelo de los controles existentes?**

Mala		Regular		Buena	

8. **Sabiendo que la creación dinámica de un componente gráfico, sin utilizar algún Framework, conlleva un determinado tiempo de implementación, ¿Cree usted que el tiempo invertido en crear un nuevo control utilizando el Framework propuesto, es óptimo?**

Si		No	

9. **¿Cómo calificaría usted el proceso de generación dinámica de componentes gráficos al momento de ejecución?**

Mala		Regular		Buena	

10. **¿Cree usted que el uso de Procedimientos Almacenados para las operaciones que realiza el Framework reducen el tiempo de mantenimiento y acceso a datos?**

Si		No	

11. **Le parece que la conexión y acceso a datos es:**

Mala		Regular		Buena	

12. **¿Cómo calificaría la aplicación instanciada en base al Framework?**

Mala		Regular		Buena	

13. Considera que dar mantenimiento a una aplicación instanciada por el Framework es:

Difícil		Regular		Fácil	

14. ¿Considera que el Framework propuesto podría modificarse fácilmente con el fin de proveer nuevas funcionalidades?

Si		No	