Universidad Católica Santa María

Facultad de Ciencias e Ingenieras Físicas y Formales

Escuela Profesional de Ingeniera de Sistemas

DESARROLLO DE SISTEMA DE GESTIÓN PARA PLANTACION Y LOCALIZACION DE ARBOLES CON FRAMEWORK MVC.NET Y ARQUITECTURA ORIENTADO A SERVICIOS

Tesis presentada por el Bachiller:

Delgado Ballón, Gerson Américo

Para optar por el título profesional de:

Ingeniero de Sistemas

Especialidad en Ingeniería del Software

Asesor:

Mg. Zúñiga Carnero, Manuel

Arequipa – Perú

2020

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA DE SISTEMAS

DICTAMEN APROBACIÓN DE BORRADOR DE TESIS

Arequipa, 26 de Septiembre del 2020

Dictamen: 001664-C-EPIS-2020

Visto el borrador de tesis del expediente 001664, presentado por:

2009601571 - DELGADO BALLON GERSON AMERICO

Titulado:

DESARROLLO DE SISTEMA DE GESTIÓN DE PLANTACION Y LOCALIZACION DE ARBOLES CON FRAMEWORK MVC.NET Y ARQUITECTURA ORIENTADO A SERVICIOS

Nuestro dictamen es:

APROBADO

1220 - ZUÑIGA CARNERO MANUEL MARIANO
DICTAMINADOR

1564 - CORRALES DELGADO CARLO JOSE LUIS
DICTAMINADOR

PRESENTACION

Sr. Director de la Escuela Profesional de Ingeniería de Sistemas.

Sres. Miembros del Jurado.

De conformidad con las disposiciones del Reglamento de Grados y Títulos de la Escuela Profesional de Ingeniería de Sistemas, ponemos a vuestra consideración el presente trabajo de investigación titulado: "**DESARROLLO DE SISTEMA DE GESTIÓN PARA PLANTACION Y LOCALIZACION DE ARBOLES CON FRAMEWORK MVC.NET Y ARQUITECTURA ORIENTADO A SERVICIOS**", el mismo que de ser aprobado nos permitirá optar por el Título Profesional de Ingeniero de Sistemas.

DELGADO BALLÓN, GERSON AMERICO.

DEDICATORIA

Gerson Américo Delgado Ballón

Dedico este trabajo a mi familia por ser un gran apoyo a lo largo de mi carrera. A mi madre, por ser la persona que me ha acompañado durante todo mi trayecto de vida y quien ha sabido formarme con buenos sentimientos, hábitos y valores. A mi padre, quien con sus consejos ha sabido guiarme para culminar mi carrera profesional y lograr mis metas. A mis hermanos, que siempre han estado para mí brindándome su apoyo y especialmente a mi esposa Gabriela y mi pequeña hija Emilia, que son la familia que forme y velan por mi desarrollo profesional y bienestar, por creer en mí y alentarme en todo momento.

AGRADECIMIENTO

A mi asesor de Tesis, Ing. Manuel Zúñiga Carnero, le agradezco por su valioso tiempo, ya que gracias a sus valiosos consejos y enseñanzas pude lograr culminar este proyecto, por su paciencia, por su valiosa dirección y apoyo para seguir este camino y llegar a la culminación del trabajo de tesis.

A mi familia y a ustedes, mi mayor reconocimiento y gratitud.

RESUMEN

Este trabajo consiste en el diseño, desarrollo e implementación de una aplicación informática. El Objetivo es permitir una mejor gestión y seguimiento sobre el proceso de plantación y localización de árboles, para lograr que el área de Responsabilidad Social de la Universidad Católica de Santa María, tenga un mejor control del alumnado que realiza esta actividad.

Para llevar a cabo el proyecto se realizó un análisis de los requerimientos básicos, las funcionalidades deseadas, la información previa y los intereses del área responsable, a través de reuniones y entregables basados en la metodología SCRUM.

Posteriormente se efectuó un proceso de selección de tecnología, con el fin de identificar la más adecuada para el presente desarrollo. Para la solución planteada, se ha requerido implementar un servidor web, un gestor de base de datos y un entorno de desarrollo, para lo cual se optó por tecnologías que sean gratuitas, confiabilidad y respaldo que tienen con su amplio uso y prestigio de las mismas.

Una vez que se eligió todo el entorno de desarrollo, se planifico el proyecto de la siguiente manera:

Primero, se realizó el análisis de los requisitos, clasificándolos como funcionales y no funcionales, para así tener una mayor idea de lo que se espera del sistema a desarrollar.

Segundo, se realizó el diseño de la Base de Datos, que en base a los requerimientos se organizó las entidades necesarias a intervenir en el proyecto, con dichas entidades se manejara la persistencia de Datos, con la cual se almacenará la información deseada.

Tercero, se realizó el Análisis y Diseño de la arquitectura de Software, para que se tengan mapeados todos los módulos y componentes del software, así posteriormente realizar el diseño de la web, contemplando todos los artefactos generados.

Para brindar una mayor escalabilidad, tener una mejor independencia en el desarrollo del sistema y mejorar el desempeño, se optó por realizar una Arquitectura Orientada a Servicios (SOA), con el cual se tendrán separadas las diferentes funcionalidades del sistema, logrando así una menor dependencia entre los mismos y tener un desarrollo más de-centralizado, permitiendo al sistema ser más escalable y multiplataforma, ya que al tener el Core funcional del sistema en los diferentes servicios web, se logrará interactuar con diferentes plataformas.

Finalmente, con el Sistema de Gestión y Localización de Árboles, le permitirá a la rea de Responsabilidad Social, tener un mejor control de los alumnos que realicen la actividad y ver mejores resultados del mismo.

Palabras claves: Servicios Web, Aplicación Web, SOA, Visual Studio, SQL Server.

ABSTRACT

This work consists in the design, development and implementation of a computer application using Web technology.

The objective is to allow a better management and monitoring of tree planting and their location, to achieve the area of Social Responsibility of the Catholic University of Santa Maria, to have a better control of the students that perform this activity.

To carry out a project, an analysis of the basic requirements, the specific functionalities, the previous information and the interests of the responsible area can be carried out, through the meetings and the deliverables based on the information.

Subsequently, a technology selection process was carried out in order to identify the most appropriate to the requirements of the need. For the proposed solution, it is required to implement a web server, a database manager and a development environment, for which we opted for technologies that are free and the reliability and support they have with its wide use and prestige. The same.

Once the entire development environment was chosen, the project was planned as follows.

First, the analysis of the requirements was carried out, classifying the functional and non-functional requirements, in order to have a better idea of what is expected of the system to be developed.

Second, the design of the Database was carried out, based on the requirements, the entities needed to intervene in the project were organized, with these entities the persistence of Data will be managed, with which the desired information will be obtained.

Third, the Analysis and Design of the Software architecture was carried out, so that all the modules and components of the software are mapped and subsequently the Web Design is done, contemplating all the artifacts generated by the architecture and thus ensuring the efficiency and system efficiency.

To provide greater scalability, have a better independence in the development of the and improve performance, we opted for a Service Oriented Architecture (SOA), with the different functionalities of the system will be separated, thus achieving a lower dependency between them and have a more de-centralized development, allowing the system to be more scalable and multiplatform since having the functional core of the in the different web services, it will be possible to interact with different platforms.

Finally, with the System of Management and Location of Trees, it will allow the area of Social Responsibility, to have a better control of the students who carry out the activity and see better results of it.

Keywords: Web Services, Web Application, SOA, Visual Studio, SQL Server.

INTRODUCCIÓN

Actualmente se tiene un gran problema con la contaminación, el Perú es el tercer país más vulnerable al cambio climático, la contaminación ambiental en el Perú está alcanzando cifras alarmantes. Los problemas aumentan tras el continuo incremento de la población, su concentración en grandes centros urbanos y las actividades ilegales, como la minería la quema de basura y la gran cantidad de gases tóxicos que emanan los automóviles Figueroa, D, 2019). Precisamente en la ciudad de Arequipa, el mayor incremento del automotor y la poca planificación en el crecimiento urbano, hace que cada vez exista áreas verdes en la ciudad, esto ocasiona a que se tenga mayor porcentaje de contaminación menos aire puro que respirar.

Frente al presente problema social, la Universidad Católica de Santa María, siempre preocupándose por el bienestar, desarrollo social y ambiental de la población arequipeña, es que lanzó una campaña de Plantación de Árboles, mediante la cual desea contribuir con el medio ambiente, desarrollando un plan para mejorar y aumentar las áreas verdes de la ciudad, para ello, el área de Responsabilidad Social, implemento un plan, el cual consiste en brindar un árbol a cada nuevo alumno de la universidad (cachimbos), para que puedan plantar y realizar el cuidado del mismo, contribuyendo así en la mejora del ambiente y así también hacer partícipe a los alumnos mejorando su calidad y desarrollo de valores.

Para tener un mejor control y seguimiento de lo planteado anteriormente, es que se propone desarrollar un sistema, para que ayude al área de Responsabilidad Social a tener la información sobre la localización de los árboles que estén plantando los alumnos, fotos del plantado y cuidado del árbol, con dicha información tener un mejor control de la actividad realizada, con ello se podrá brindar beneficios universitarios a los alumnos que realicen el adecuadamente la actividad.

El presente trabajo, presenta la documentación del proceso de ingeniería sobre análisis, diseño e implementación de la aplicación web planteada para la solución de la antes descrita y se organiza de la siguiente manera: El capítulo 1 trata sobre el teórico relacionado con el problema de investigación. En el capítulo 2 se presenta el marco teórico correspondiente, en el capítulo 3 se presenta el levantamiento de requerimientos su respectivo análisis y definición, para que en el capítulo 4 se pueda proceder al diseño, desarrollo e implementación de la solución web, basados en los requerimientos anteriormente definidos, en el capítulo 5 se lleva a cabo la ejecución del sistema,

se presentan los resultados, las conclusiones y recomendaciones (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

CAPITULO I

PLANTEAMIENTO TEÓRICO

1.1. Título del Proyecto

DESARROLLO DE SISTEMA DE GESTIÓN PARA PLANTACION Y LOCALIZACION DE ARBOLES CON FRAMEWORK MVC.NET Y ARQUITECTURA ORIENTADO A SERVICIOS.

1.2. Descripción del problema

La Universidad Católica de Santa María, desarrolló una campaña para aumentar las áreas verdes en la ciudad de Arequipa, la cual consiste en entregar un árbol a cada nuevo ingresante de la universidad y que este pueda realizar el plantado y cuidado del mismo para velar por el crecimiento y así contribuir con el desarrollo de áreas verdes de la ciudad.

Actualmente los encargados de la campaña de plantación de árboles, recae en el área de Responsabilidad Social, la cual no tiene una herramienta con la que pueda tener el control del proceso y cuidado de los árboles, tampoco tiene un repositorio donde puedan observar la localización de los árboles, haciendo difícil su labor para velar por el correcto cumplimiento de la actividad y con ello otorgar los beneficios que la universidad entrega a los alumnos que realicen correctamente la actividad indicada.

1.3. Delimitaciones y definición del problema

1.3.1. Delimitaciones

a. Delimitación espacial

El presente trabajo se lleva a cabo en la ciudad de Arequipa para el área de Responsabilidad Social de la Universidad Católica Santa María.

b. Delimitación Temporal

El trabajo se inicia en Febrero de 2019 y culminara en Octubre de 2020.

c. Delimitación Social

Está orientado a mejorar el proceso de seguimiento y control del área Responsabilidad Social, para así contribuir con su proyecto social de mejorar las áreas verdes en la ciudad de Arequipa.

d. Delimitación conceptual

Diseño, desarrollo e implementación de una solución web. Calidad en el desarrollo de la solución.

1.3.2. Definición del Problema

El área de Responsabilidad Social, no cuenta con una herramienta que le ayude con el control, localización y seguimiento del proceso de plantación de árboles de los alumnos, por lo cual no sabe con exactitud si se viene realizando de manera adecuada la actividad.

1.4. Formulación del Problema

1.4.1. Problema Principal

No existe una herramienta que ayude al área de Responsabilidad Social a tener la información sobre la localización del árbol y evidencia del proceso del cuidado.

1.5. Objetivos de la Investigación

1.5.1. Objetivo General

Diseñar, desarrollar e implementar una solución web, con el uso de Tecnologías de Información y Comunicación, que permita ver la localización y evidencias (fotos) del cuidado del árbol, para así tener un mejor control del proceso.

1.5.2. Objetivos Específicos

1. Establecer un prototipo para la solución web, lo que incluye, interfaces, estructuras de desarrollo y la selección de un servidor web que esté de acuerdo a los requerimientos del proyecto a desarrollar.
2. Mejorar el proceso de plantación de árboles que tiene el área de Responsabilidad Social, permitiendo a los alumnos ingresar la localización y las evidencias (fotos) de sus árboles mediante la web.
3. Mostrar la información que proporcionaron los alumnos (localización, fotos, fecha de plantación), para que Responsabilidad Social realice el seguimiento y control para ver si corresponde los beneficios que ofrece la universidad al alumno.
4. El área de responsabilidad Social, pueda enviar mensajes a correo universitario y personal, con el fin de que el alumno tenga alertas en sus dispositivos móviles sobre algún inconveniente que el área detecte.
5. Validar la eficiencia y completitud del sistema.

1.6. Viabilidad de la Investigación

1.6.1. Económica

Los recursos económicos necesarios para solventar el presente trabajo de investigación, son asumidos por el tesista.

1.6.2. Técnica

Se cuenta con la capacidad académica y los conceptos adecuados para resolver el problema.

1.6.3. Operativa

Medios bibliográficos, internet, bibliotecas y otros centros de estudio con infraestructura, laboratorios, entre otros.

1.7. Justificación e Importancia de la Investigación

1.7.1. Justificación

Existe la necesidad urgente de construir una solución web, para que el área de Responsabilidad Social, tenga la información adecuada de la actividad de plantación de árboles (localización, fotos, fechas), para así tener un mejor control.

Así mismo, al ser una aplicación web, el alumno podrá realizar desde la comodidad de su hogar el ingreso de la información que requiera, para así evidenciar la conformidad de su actividad y acceder a los beneficios universitarios.

1.7.2. Importancia

Permitir que el área de responsabilidad social, logre tener un adecuado seguimiento y control sobre la realización de la actividad, para poder lograr ofrecer el beneficio que ofrece la universidad a quienes realicen adecuadamente la labor y con ello contribuir con el medio ambiente de la ciudad.

1.8. Limitaciones de la Investigación

Actualmente existen aplicaciones de control de tala de árboles, aplicaciones de geo-localización, pero, son escasas las aplicaciones relacionadas al proyecto descrito, no se tienen antecedentes que puedan servir como base del desarrollo, así mismo, la solución web surge debido a que la actividad a realizar, es un proyecto propio de la Universidad Católica Santa María, para el desarrollo y mejora del medio ambiente de la población.

1.9. Área, línea, tipo y nivel de la investigación

1.9.1. Área de investigación

El área de investigación es la Ingeniería de Software.

1.9.2. Línea de investigación

Ingeniería del Software.

1.9.3. Tipo de investigación

Aplicada.

1.9.4. Nivel de investigación

Experimental o evaluativa, porque se propone mejorar el desempeño de las actividades que realiza el área de Responsabilidad Social con su proyecto de Plantación de Árboles.

1.10. Método y diseño de investigación

1.10.1. Método de investigación

Investigación adaptiva, empleando la metodología ágil SCRUM.

1.10.2. Forma de tratamiento de los datos.

Para el proceso de los datos que el alumno ingrese a la web se utilizará.

a) Api de geo-localización de Google Maps

Para poder acceder a la localización del alumno que ingrese a la web, se utilizara el api de google para obtener la localización en tiempo real del mismo.

b) Soporte de Carga de Fotos

Con el fin de guardar en Base de Datos las evidencias que otorguen los alumnos de su proceso realizado, la web permitirá cargar fotos que el alumno tome y posteriormente almacenarlo en la Base de Datos.

c) Visualización de Información

Responsabilidad Social, podrá ver los datos ingresados por los alumnos, para así realizar el control y evaluación de los datos proporcionados.

d) Envío de Mensajes

Mediante la web, se podrían enviar mensajes al correo alumno, el cual llegaran a su dispositivo móvil, para que este esté alertado en todo momento de cualquier observación y pueda subsanar la misma.

1.11. Cobertura del Estudio

1.11.1. Universo

Alumnos ingresantes y personal de Responsabilidad Social de la Universidad Católica Santa María – Arequipa, Perú.

1.11.2. Muestra

Alumnos de Primer Año de la Facultad de Ingeniería de Sistemas de la UCSM, con los cuales se realizó el proceso de prueba de la web.

CAPITULO II
MARCO TEORICO

2.1. Estado del Arte

Existen pocos sistemas que puedan ser tomados como base para el presente proyecto, no se tienen sistemas parecidos, por lo cual, lo sistemas más parecidos, son sistemas de localización (SIG), los que se tomaron como referencia, son los siguientes:

Primero, existe un trabajo de la universidad “Francisco José de Caldas”, presentada por Daniel Rivas Torres, el cual consiste en almacenar e identificar los datos de información geográfica de las zonas urbanas de su localidad, tomando como referencia datos cartográficos y de localización mediante GPS y con ello administrar las zonas que tengan mayor cantidad de árboles en las zonas urbanas. Este trabajo es un acercamiento al desarrollo de un SIG para los arboles urbanos, que requiere el trabajo conjunto de los temáticos de la Dasonomía Urbana, la Arboricultura y los especialistas en SIG (Rivas Torres, 2000).

Segundo, existe un sistema de identificación y geolocalización de áreas verdes y jardines, el cual está basado en Sistemas de Información Geográfica, estos se basan en datos satelitales y cartográficos, con lo cual tienen una base de datos de áreas verdes y llevar la administración de las mismas. Los SIG se pueden utilizar para rastrear muchos otros documentos y análisis de planificación que los gestores desarrollan y archivan, incluidas encuestas de población, mapas de vegetación y arbolado urbano (Bonells, 2020).

Tercero, en el siguiente proyecto de investigación, utilizan los sistemas SIG (Sistemas de Información Geográfica), dichos sistemas son basados en data satelital y cartográfica, almacenando los datos de las zonas verdes de las localidades, dichos sistemas son utilizados para la silvicultura. Como una fase piloto, para adquirir experiencia, definir metodologías y determinar las ventajas de los Sistemas de Información Geográfica (SIG) al ser aplicados en la Silvicultura Urbana, se realizó el inventario, diagnóstico, propuesta de manejo y valoración económica del Bosque Urbano del barrio La Magnolia del Municipio de Envigado, Departamento de Antioquia, Colombia (Otaya Burbano, L. A., Sánhez Zapata, R. D. J., Morales Soto, L., & Botero Fernández, V, 2006).

En 2013, el Quinto Informe de Evaluación (AR5) del Grupo Intergubernamental de expertos sobre el Cambio Climático (IPCC) concluyó que "es extremadamente probable que la influencia humana ha sido la causa dominante del calentamiento observado desde la mitad del siglo xx". La mayor influencia humana ha sido la emisión de gases de efecto invernadero como el dióxido de carbono, metano y óxidos de nitrógeno. Las proyecciones de modelos climáticos resumidos en el AR5 indicaron que durante el presente siglo la temperatura superficial global subirá probablemente 0,3 a 1,7 °C para su escenario de emisiones más bajas usando mitigación estricta y 2,6 a 4,8 °C para las mayores. Estas conclusiones han sido respaldadas por las academias nacionales de ciencia de los principales países industrializados y no son disputadas por ninguna organización científica de prestigio nacional o internacional (IPCC, Climate Change, 2013).

2.2. Ingeniería de Software

La Ingeniería de Software o proceso de ingeniería de software se define como: "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad" (Pressman, R. S, 2006).

Este proceso también suele ser llamado, Ciclo de vida del software el cual comprende de las siguientes fases Levantamiento de requerimientos, Análisis y Diseño, Construcción, Pruebas e Implementación.

- Levantamiento de Requerimientos: Define el alance del proyecto y las funciones que tendría el mismo.
- Análisis y Diseño: Etapa donde se elabora el plan del proyecto, define y elabora la arquitectura.
- Construcción: Donde se realiza el desarrollo del software.
- Pruebas: Unitarias e Integrales, para verificar el funcionamiento y cumplimiento con los requisitos.
- Implementación: Instalación y estabilización en el entorno de producción del software, producto final (Pressman, R. S, 2006).

En la práctica casi todos los proyectos no cubren todos los requisitos en su primera versión, por lo que una vez realizada una entrega se vuelve a abordar una versión posterior que incluye nuevas funcionalidades. Asumir que esto es así desde un

principio es lo que plantea el ciclo de vida iterativo: que una vez realizada la integración o entrega de una versión, se volverá a la fase de análisis para seguir avanzando en el desarrollo, es así como surgió el modelo iterativo incremental, mejorando algunas deficiencias del modelo primitivo en cascada (IEDGE Business School, 2020).

Modelo Iterativo Incremental: Surge en base al modelo en cascada, mejorando algunas deficiencias como entregables por fases, asociándose a un modelo ágil, para un mejor control de versiones y cumplimiento adecuado de los requisitos, desarrollando el software por etapas y realizando entregables pequeños para que el usuario valide.

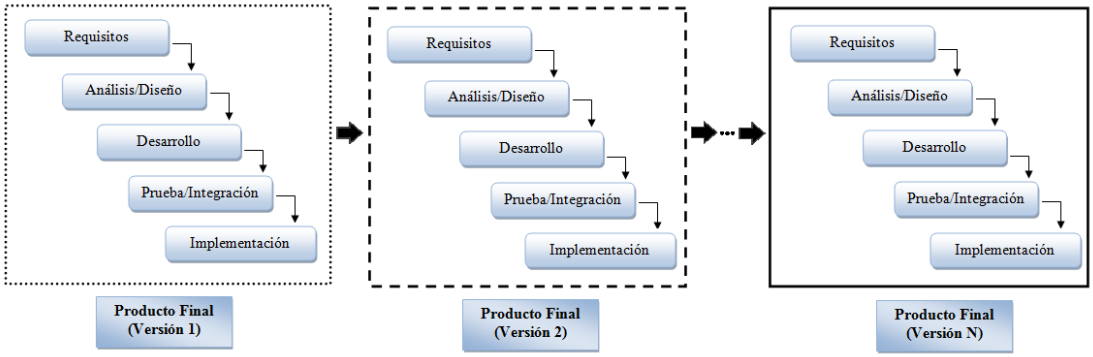

Figura 2.1 El "modelo iterativo incremental"

Fuente: Jacobson, I., Booch, G. y Rumbaugh, J, (2000).

2.3. Lenguaje Unificado de Modelado UML.

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. Este lenguaje nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo (Hernández Orallo, 2002).

UML, es un lenguaje utilizado para el modelamiento de los componentes del sistema, como también para diagramas de casos de uso, secuencia, etc., logrando representar la lógica del sistema de manera visual y a alto nivel.

2.3.1. Objetivos del lenguaje unificado de modelado

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión (Hernández Orallo, 2002).

2.3.2. Utilidad del lenguaje unificado de modelado

Se utiliza el lenguaje unificado, para representar de manera visual los componentes y lógica a alto nivel del sistema, representando el comportamiento de este y viendo los productos a desarrollar, para que así se llegue y cumpla con una mejor calidad en el desarrollo de software, se elaboran dichos modelos, para tener una mayor comprensión de lo que se debe realizar (Hernández Orallo, 2002).

Entre las utilidades del modelado se cuentan:

- Visualizar cómo es o queremos que sea el sistema
- Especificar la estructura y comportamiento del sistema
- Proporcionan plantillas que guían la construcción del sistema
- Documentan las decisiones (James R, Ivar J, Grody B, 2002).

2.3.3. Fases del ciclo de desarrollo que soporta UML

Cada diagrama puede ser usado con énfasis distinto en las fase de desarrollo: análisis, diseño e implementación, un diagrama cualquiera en una fase de tendrá un estudio lógico, cabe aclarar que aunque UML es orientado a objetos preferentemente, esto es útil en cualquier modelo tecnológico ya que es independiente de lenguajes de programación o tecnología determinada (Jacobson, I., Booch, G. y Rumbaugh, J, 2000).

Con ello se tiene una mejor documentación y trazabilidad, para tener un mejor control de que se realizó y que se debe hacer por cada fase de la construcción del software.

2.3.4. Diagramas que ofrece el UML

2.3.4.1. Diagrama de Casos de Uso

Un caso de uso es una unidad coherente de funcionalidad, externamente visible proporcionada por una unidad del sistema y expresada por secuencias de mensajes intercambiados por la unidad de sistemas y uno o más actores (Ejemplo Figura 2.2). El propósito de un caso de uso es definir una pieza de comportamiento coherente, sin revelar la estructura interna del sistema. La definición de un caso de uso incluye todo el comportamiento

que implica: Las líneas principales, las diferentes variaciones sobre el comportamiento normal, y todas las condiciones excepcionales, que pueden ocurrir con tal comportamiento, junto con la respuesta deseada. Desde el punto de vista de los usuarios estás pueden situaciones anormales. Desde el punto de vista de los sistemas son las variaciones adicionales que deben ser descritas y manejadas (James R, Ivar J, Grody B, 2002).

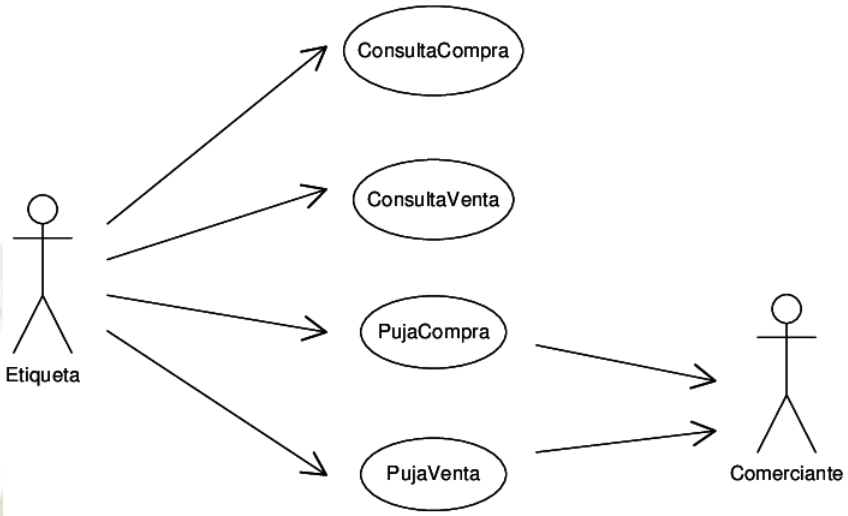

Figura 2.2 Ejemplo de Diagrama de Caso de Uso

Fuente: Propia.

2.3.4.2. Diagrama de Secuencia

Un diagrama de secuencia representa los pasos que sigue el sistema para realizar una determinada función, indicando los componentes con los que interactúa. Cada componente está representado por una línea vertical, que representa la línea de vida de dicho objeto en el proceso evaluado.

Se muestra un mensaje como una flecha desde la línea de vida de un objeto a la del otro. Las flechas se organizan en el diagrama en orden cronológico hacia abajo (Figura 2.3).

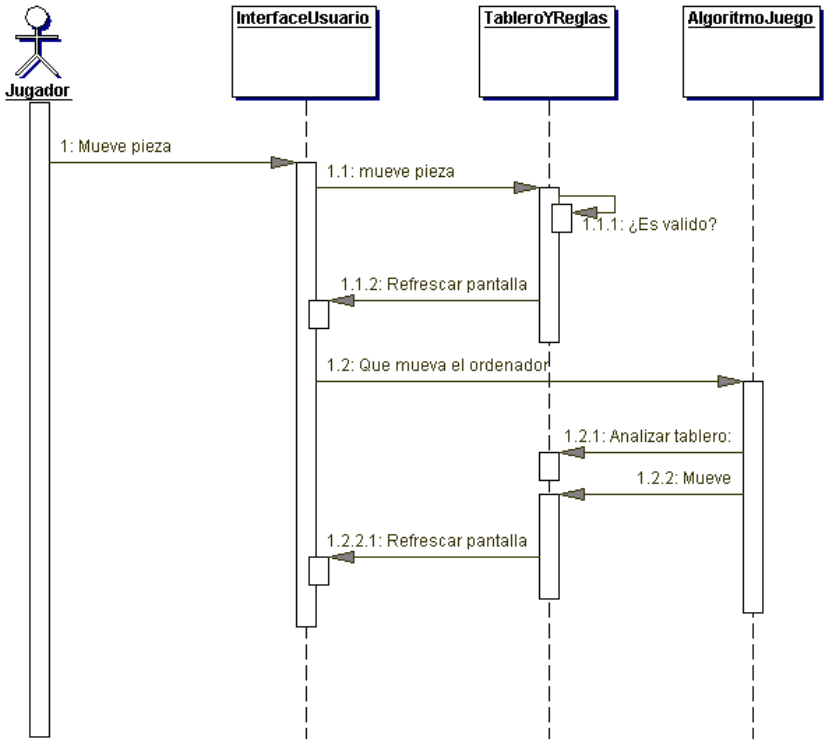

Figura 2.3 Ejemplo de Diagrama de Secuencia

Fuente: Propia.

2.3.4.3. Diagrama de Actividades

Un diagrama de actividades es la notación para un gráfico de actividades. Incluye algunos símbolos especiales abreviados por conveniencia como nodos de acción, nodos de objeto y nodos de control (Figura 2.4). Estos símbolos pueden usarse en cualquier diagrama de estados, aunque mezclar la notación puede ser molesto (James R, Ivar J, Grody B, 2002).

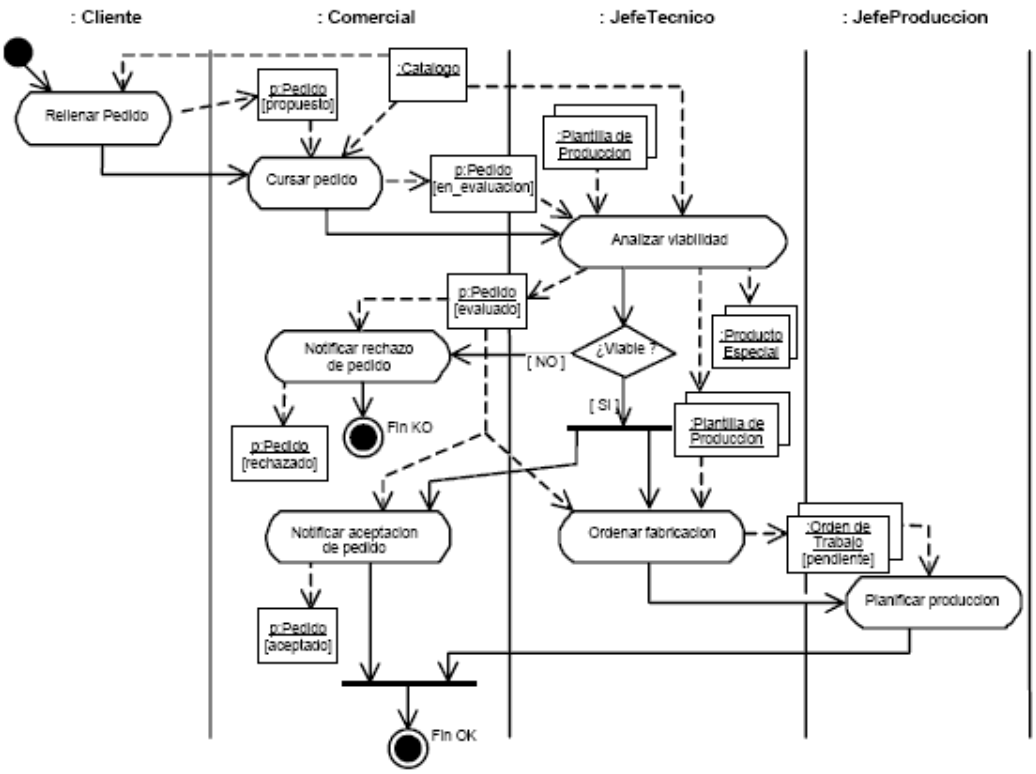

Figura 2.4 Símbolos para diagramas de actividades.

Fuente: propia

2.4. Web Services - Web Api

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma (Alpañez Soler , 2012).

Los web Services usan los protocolos Http o Soap, para la interacción con los sistemas, con los web services se pueden separar las funcionalidades del sistema por módulos, para que puedan ser desarrollados de forma independiente y más rápida, nos permite realizar integración con diferentes lenguajes de programación, permitiendo a un sistema ser multiplataforma.

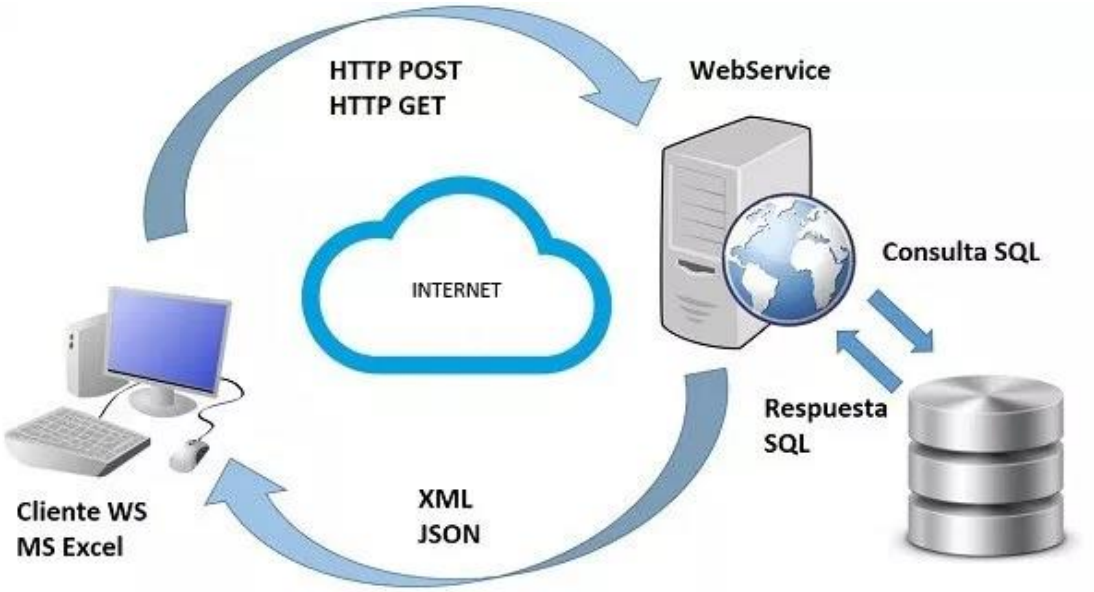

Figura 2.5 Diagrama Web Service.

Fuente: propia

2.5. Programación orientada a objetos

La programación Orientada a Objetos es una metodología que basa la estructura de los programas en torno a los objetos. Los lenguajes de POO ofrecen medios y herramientas para describir los objetos manipulados por un programa. Más que describir cada objeto individualmente, estos lenguajes proveen una construcción (Clase) que describe a un conjunto de objetos que poseen las mismas propiedades (Carballo Yusneyi, 2007).

La POO sirve para manejar de manera más sencilla los datos, logrando encapsulamiento de los mismos y con ello tener mayor integridad de los mismos, permite también el manejo de herencia y con ello poder realizar el desacoplamiento del código, separando la parte visual, con el funcionamiento del negocio, logrando obtener en una biblioteca de clases la funcionalidad separada de la parte visual, logrando programación en n capas.

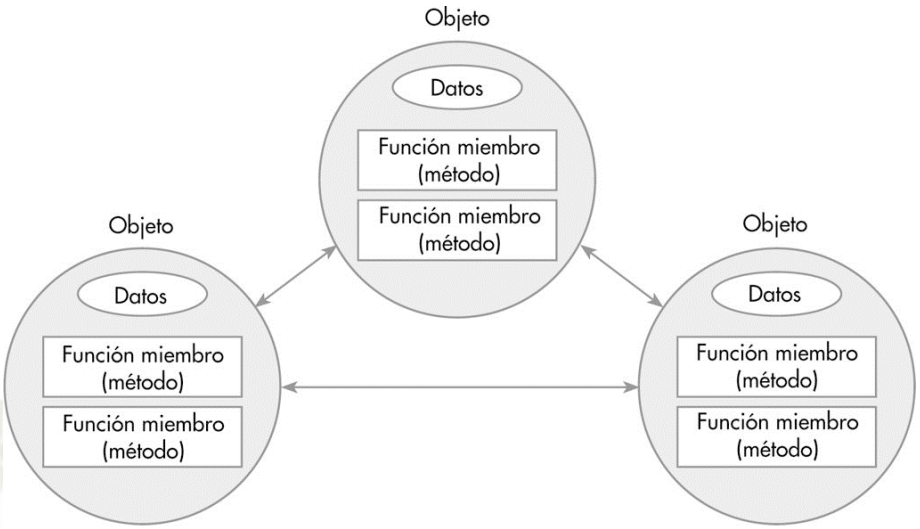

Figura 2.6 Programación Orientada a Objetos

Fuente: (Carballo Yusneyi, 2007).

2.6. Modelo Cliente – Servidor

Cliente/Servidor es una arquitectura que separa el procesamiento entre clientes y servidores en una red. Con la arquitectura Cliente/Servidor, logra separar los procesos del lado del Cliente (usuarios pc) y Servidor (usualmente donde reside la BD), logrando una mayor rapidez de procesamiento, debido a que normalmente el cliente envía los datos para ser procesados en el servidor.

Con la llegada de Internet, se ha separado la parte visual con el proceso interno de la aplicación, lo cual permitirá a un servidor de aplicación ofrecer servicios al browser de la Web el cual es independiente de la aplicación, con ello se tiene mayor seguridad, para no enviar data importante mediante el browser (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

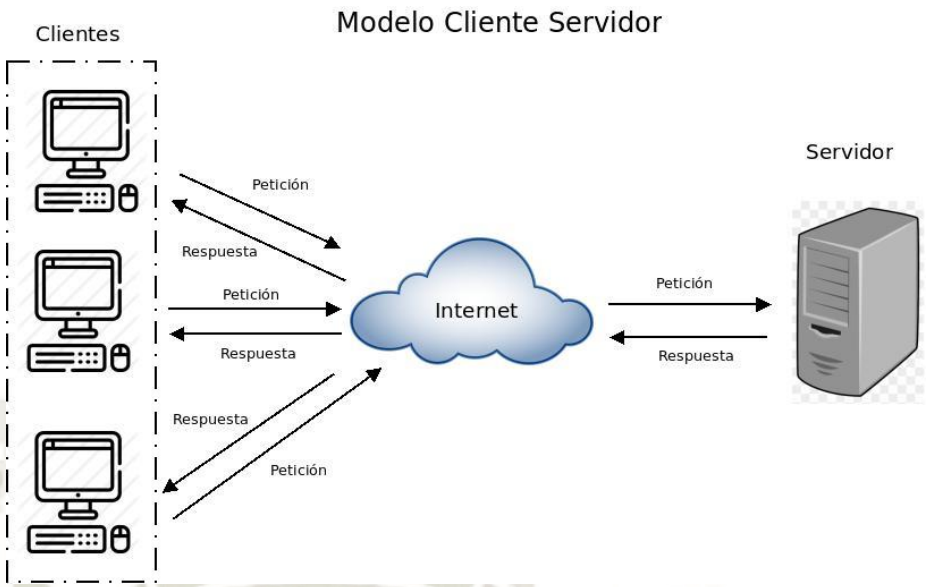

Figura 2.7 Modelo Cliente-Servidor

Fuente: Propia.

2.6.1. Cliente

El cliente es la parte visual del software, donde se encuentran las interfaces que interactúan con el usuario, mediante la cual el usuario envía peticiones para ser procesadas en el servidor, interpreta los datos obtenidos por el servidor, mostrando de una manera amigable al usuario, conocida como front-end.

2.6.2. Servidor

Es el proceso encargado de recibir las peticiones enviadas por el usuario, registrando datos o mostrando según el requerimiento, normalmente es en el servidor donde se tienen las reglas de negocio que maneja el software, es conocida como back-end.

2.7. Tipo de Aplicaciones

A continuación, se muestra las diferencias que existen entre las principales aplicaciones nombradas anteriormente, señalando sus ventajas y desventajas.

Tabla 1

Tabla comparativa entre aplicación web y escritorio.

APLICACIONES WEB		APLICACIONES DE ESCRITORIO	
VENTAJAS	**DESVENTAJAS**	**VENTAJAS**	**DESVENTAJAS**
Normalmente no se necesita instalar nada para usar el Software.			El Software debe Instalarse siempre en el pc del usuario.
	El aprovechamiento de los recursos de la maquina cliente se ve limitada debido a que se ejecuta en el navegador.	Aprovecha mayormente los recursos de la maquina en la cual se ejecuta.	
Las actualizaciones son de forma inmediata, se modifica el código en el servidor y automáticament e todos los clientes cuentan con la nueva versión disponible.			Se debe utilizar algún sistema de control de versiones para actualizar cada estación de trabajo y en ocasiones actualizar de forma manual computador por computador.
Todo el código se ejecuta en el servidor devolviendo una respuesta en HTML			La mayoría del código se ejecuta en el pc del cliente y en muchas ocasiones ralentizando el rendimiento de este.
Una aplicación Web fácilmente puede ejecutarse			Una aplicación de escritorio no puede ejecutarse en

en cualquier dispositivo que cuente con un navegador.			cualquier dispositivo debido a su arquitectura.
Una aplicación Web es de forma automática multiplataforma , lo que permite su ejecución en varios sistemas operativos.			Para que una aplicación de escritorio será multiplataforma se debe tener en cuenta muchos factores, entre ellos el lenguaje de programación y las funciones que la aplicación valla a realizar.

Fuente: Propia

2.7.1. Análisis y Selección de aplicaciones.

Para el presente proyecto, se elige la plataforma web, debido a que se requiere tener la aplicación descentralizada y con acceso desde cualquier parte, también para que tenga mayor escalabilidad y pueda volverse multiplataforma al usar servicios.

2.8. Servidor Aplicaciones

Un servidor de aplicaciones es el elemento (software) que es capaz de traducir las instrucciones y además comunicar con otros servidores (como por ejemplo los servidores de bases de datos) para extraer información de la empresa que se necesita para resolver la petición. Se puede tener un servidor local (hardware) y configurarlo para que este pueda alojar las aplicaciones web y que sean públicas, logrando tener un servidor físico donde puedas administrar tus aplicaciones y con ello mostrar la web en Internet (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

2.9. Gestores de Base de Datos

Tabla 2

Tabla comparativa de Gestores de Base de Datos.

SGBD	EMPRESA	LICENCIA	VENTAJAS	DESVENTAJAS	PLATAFORMAS	HERRAMIENTA
Oracle	Oracle Corporation	Privada	✓ Oracle es el motor de BD relacional más usado a nivel mundial ✓ Puede ejecutarse en todas las plataformas, desde un PC hasta en una supercomputadora ✓ Oracle es la BD con orientación hacia Internet. ✓ Soporte de transacciones ✓ Estabilidad ✓ Soporte multiplataforma	● El precio del producto y de sus licencias ● Un Oracle mal configurado es potencialmente lento e inestable	Microsoft Windows, Linux,Unix	Oracle Designer
MySQL	Sun Microsystem	Libre a nivel de usuario Privada para empresas	✓ Utiliza pocos recursos ✓ Fácil configuración e instalación ✓ Múltiples motores de almacenamiento ✓ Agrupación de transacciones ✓ Replicación segura	● No tiene soporte ● No permite el modo de autentificación local ● No sincroniza los datos con otras bases de datos	Microsoft Windows, Linux,Unix	MySQL Workbench
Microsoft SQL Server	Microsoft	Privada	✓ Gran facilidad de configuración e instalación ✓ Utiliza una extensión al SQL estándar, que se domina Transact SQL ✓ Seguridad: SQL permite administrar permisos a todos sus elementos ✓ Permisos a nivel de servidor, seguridad en tablas, permitir o denegar lectura, etc. ✓ Ofrece una potente forma de unir SQL e Internet ✓ Consultas Jerárquicas con select from	● Requiere gran memoria RAM para instalar y utilización de software ● La relación calidad-precio está considerada por debajo de otros SGBD	Microsoft Windows	Management Studio

Fuente: Propia

2.9.1. Análisis y Selección

Se realizó un análisis basado en el cuadro comparativo mostrado anteriormente, cabe resaltar que de todos los gestores mencionados, tienen su respectiva versión gratuita, debido a ello, se eligió el gestor SQL Server, debido a su gran potencia en el procesamiento de consultas e integración con las tecnologías .net en el cual se desarrollara el proyecto.

2.10. Lenguajes de Programación

Tabla 3

Tabla comparativa de Lenguajes de Programación

Lenguaje	Características	Fortalezas	Debilidades
PHP	• Utilizado para generar páginas web dinámicas • Se ejecuta en el servidor • Los usuarios no pueden ver el código PHP únicamente reciben en sus navegadores código HTML • Las páginas que genera son visibles para prácticamente cualquier navegador y computadora o dispositivos móviles que pueda interpretar el HTML. • No se necesita la instalación de PHP en el lado del cliente. • Versiones resiente permiten la POO • Lenguaje de alto nivel	• Su sintaxis es muy similar a otros lenguajes • Fácil • Es un lenguaje muy popular tiene una comunidad muy grande • Rápido • Multiplataforma • Maneja base de datos • Bastante documentado • Libre y gratuito. • Varias funciones • No requiere definición de variables	• Necesita un servidor para funcionar • La POO es deficiente para aplicaciones grandes • Todo el trabajo se realiza el en servidor y mucha información o solicitudes pueden ser ineficiente.
ASP.NET	• Sucesor de ASP • Creada por Microsoft • De paga • Orientado a objetos	• Controles de usuarios y personalizados • Fácil mantenimiento	• Mayor consumo de recursos

		• Incremento en velocidad • Mayor seguridad	
Python	• Permite la creación de todo tipo de programas incluso sitios web • No requiere de compilación es un código interpretado	• Libre y código fuente abierto • Lenguaje de propósito general • Multiplataforma • Orientado a objetos • Portable	• Los lenguajes interpretados suelen ser relativamente lentos
JavaScript	• Es un lenguaje interpretado • Es similar a java • Es orientado a objetos	• Los script tiene capacidad limitada por razones de seguridad • Se ejecuta del lado del cliente • Lenguaje de scripting seguro y fiable	
C++	• Orientado a objetos • Rápido	• Ideal para sistemas robustos • IDEs de desarrollo son DEV C++, BORLAND C, TURBO C • Es multiplaforma	• No soporta creación de aplicaciones web • Complejo visualmente
C#	• Está orientado a objetos • Esta estandarizado por Microsoft como parte de su plataforma net.	• Se desempeña de forma plena en los sistemas operativos Windows. • Sintaxis más en comparación con C y C++ • Posibilidad de realizar aplicaciones web, de escritorio y móviles.	• Requiere un mínimo de 4 gb para su instalación.
Java	• Es orientado a objetos • Multiplataforma	• Al ser orientado a objetos permite su modularización • Permite la creación de aplicaciones de escritorio Tiene soporte a desarrollo	• Es un lenguaje interpretado así que es relativamente lento en comparación

		de aplicaciones móviles y web.	con otros lenguajes

Fuente: Propia

2.10.1. Análisis y Elección

El lenguaje de programación debe ser universal, es decir, cualquier problema debe tener una solución que puede ser programada en el lenguaje y dicha solución ser implementada en cualquier computador. Este requisito es uno de los más fuertes y pocos lenguajes lo poseen. Uno de ellos es el lenguaje es C# que puede desempeñarse en diferentes plataformas, entre ellas .NET. En definitiva se trata de un lenguaje moderno, intuitivo y muy eficiente, que mejora la productividad en el desarrollo de software. Además es el lenguaje de la plataforma .NET con más y mejores ejemplos; es más, la propia plataforma fue desarrollada con este lenguaje. Fue por eso que se eligió C# como lenguaje de programación a usarse en el proyecto (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

2.11. Cuadro Comparativo de Entornos de Desarrollo C#

Tabla 4

Tabla comparativa de Entornos de Desarrollo

Características	MonoDevelop	SharpDevelop	Visual Studio Express
Autocompletar Código	Si	Si	Si
Iluminación en código	Si	Si	Si
Diseño Windows Form	No	No	Provee un diseño desarrollador
Diseño WPF	Si	Si	Si
Cobertura de Código	Si	Si	No
Unidad de Testeo	Si	No	No
Lenguajes de Programación	C#, C++, VB.NET, Boo, F#.	C#, C++, VB.NET, Boo, F#, IronPython, IronRuby	C#, C++, VB.NET, JavaScript (IronPython y IronRuby is compotible con Visual Studio 2010)

Desarrollo Móvil	No	No	Si
Desarrollo Aplicaciones Nube	No	No	Si
Expresiones de Testeo	No	Si	No
Vista de clases	Si	Si	Si
Explorador de Soluciones	Si	Si	Si
Formato de Proyectos	MSBuild	MSBuild	MSBuild
Referencias Web	Si	Si	Si
Refactorización	Renombrar, Extraer método, eliminar las importaciones no utilizadas	Renombrar, Extraer método.	Renombrar, Extraer método.
"Ir a definición"	Si	Si	Si
Encontrar Referencias	Si	Si	Si
Generación de código	Si. No tan desarrollado como visual studio.	Si. No tan desarrollado como visual studio.	Si
Buscador de Objetos	No	No	Si
Explorador de Base de Datos	Si	Si	Si

Fuente: Propia

2.11.1. Análisis y Elección de Entorno de Desarrollo

Visual Studio es el entorno de desarrollo más utilizado en mundo de la programación, es un completo IDE extensible y gratuito para crear aplicaciones modernas para Windows, Android e iOS, además de aplicaciones web y servicios en la nube. Por tal motivo, debido a que el proyecto será con tecnología .net, se optó por desarrollar en Visual Studio (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

2.12. Geolocalización Google

La API de geolocalización devuelve un radio de ubicación y precisión basado en información sobre torres de telefonía móvil y nodos WiFi que el cliente móvil puede detectar. Este documento describe el protocolo utilizado para enviar estos datos al servidor y para devolver una respuesta al cliente.

La comunicación se realiza a través del protocolo HTTP, tanto la solicitud como la respuesta están formateadas como JSON, y el tipo de contenido de ambos es application/json.

Para el presente proyecto, se utilizara el api de google maps, con el cual se pretende obtener la ubicación en tiempo real y almacenarla en la base de datos, el api se integrara a la web, para que en la web se muestre el mapa de la localización del usuario que haya accedido.

2.13. Arquitectura Orientada a Servicios (SOA)

La Arquitectura orientada a servicios, es un estilo de arquitectura, en la cual se apoya en la orientación a servicios, es decir, se desprende la lógica de negocios con sus diferentes módulos y esa porción lógica se administra desde un servicio externo a la aplicación.

Con dicha arquitectura se pretende volver más escalable y de hacer más sencillo mantenimiento a la aplicación, de igual manera se vuelve multiplaforma, al estar toda su lógica de negocios en servicios que pueden ser consumidos por cualquier aplicativo desde diferente plataforma.

CAPITULO III

ANALISIS Y DEFINICION DE REQUERIMIENTOS

El propósito fundamental de este capítulo es identificar y definir los requerimientos, lo cual permitirá guiar el desarrollo del sistema a diseñar durante el ciclo de vida, para lo cual se seguirá el enfoque en "iterativo incremental" de la Ingeniería de Software combinados con la metodología Scrum que establece los siguientes pasos; Análisis, Diseño, Desarrollo, Prueba e Implantación, realizando entregables de pequeñas funcionalidades para que puedan ser validadas por el usuario, validando y corrigiendo algunas observaciones, hasta llegar al producto final. La ingeniería de Software, como cualquier ingeniería, requiere modelar tanto el problema como sus posibles soluciones.

Los modelos ayudan a definir qué información se requiere para realizar el software y que datos son necesarios para obtener dicha información, para ello se requiere obtener conocimiento funcional de cómo se desea obtener la solución al problema, para ello se hace un levantamiento de requerimientos para determinar la funcionalidad del software.

Se debe tener un sólido conocimiento del problema al que se afronta, para así poder realizar una eficaz recolección de los requerimientos, permitiendo así diseñar un producto final que cumpla con las expectativas esperadas.

Se deben clasificar los requerimientos, para saber cuáles son los que otorguen la funcionalidad principal al proyecto o cuales son temas de rendimiento y así mejorar de la calidad del producto final, para así priorizar y realizar lo más importante primero.

3.1. Análisis de la situación actual

Actualmente la Universidad Católica Santa María, desarrollo un proyecto de plantación de árboles, con el fin de contribuir con el medio ambiente ya que es parte de sus valores, el proceso se controla por medio del área de Responsabilidad Social, el cual es ejecutado por parte del alumno.

Dicho proceso consiste en que cuando ingresen los nuevos estudiantes, se les brinda un árbol, para que ellos puedan plantar y realizar los cuidados respectivos para que este tenga un buen crecimiento, así mismo, la universidad como forma de incentivo, otorgara beneficios a aquellos alumnos que cumplan correctamente dicha labor.

El proceso antes mencionado, en la actualidad no cuenta con ninguna herramienta que apoye al seguimiento, control y evidencia de que el proceso se cumpla de manera correcta, es por ello que no saben con exactitud que alumnos puedan cumplir con la tarea, sin poder brindar los beneficios acordados de forma correcta.

El área de Responsabilidad Social de la UCSM, requiere saber la ubicación donde se plantó el árbol, así como las evidencias fotográficas del proceso realizado, también requieren alertar al alumno en caso tengan alguna observación, se requiere que el alumno pueda tener dicha alerta en su dispositivo móvil, para que este pueda estar alerta en cualquier momento y lugar, siendo más sencillo subsanar el inconveniente, con la información antes mencionada, el área de Responsabilidad Social, lograra tener un mayor control para poder brindar los beneficios a los alumnos y ver que la labor social a la cual están enfocados se cumpla de manera correcta.

3.2. Definición de requerimientos del sistema

Esta actividad incluye la determinación de los requerimientos generales para la solución del problema, mediante una serie de sesiones de trabajo con los usuarios participantes, en las cuales se obtendrá información relevante del proceso el cual se desea automatizar, con el cual se tendrá un mejor panorama del correcto funcionamiento que deberá tener el sistema. Una vez que se obtenga toda la información, se determinara la funcionalidad principal del sistema, así como los requerimientos funcionales, determinando las prioridades de los mismos, de igual manera se plantearan los requerimientos no funcionales, los cuales nos ayudaran a tener una mejor calidad de rendimiento y visualización del software a desarrollar.

3.2.1. Análisis de requerimientos

La adquisición y posterior análisis de requerimientos consiste en obtener la información correcta, la mayor posible y que sea adecuada. Es por ello que el proceso debe estar constantemente controlado, para lo cual, es necesario descomponerlas en las siguientes etapas:

- **Identificación de usuarios:** En el presente proyecto se tiene principalmente dos usuarios, los alumnos que realizaran la carga de datos al sistema y el personal del área de Responsabilidad Social, que le darán el uso principal al sistema, por tal motivo, principalmente se definieron los requerimientos con el

Área de Responsabilidad social, posteriormente con un feedback de los alumnos, se realizaron mejoras al sistema.

- **Extracción de Información:** Se realizaron entrevistas con el Área de Responsabilidad Social, para determinar los requerimientos funcionales principales que tendrá el sistema, debido a que ellos son los que realizan el proceso de dicha actividad, con ellos se determinó las funcionalidades del sistema y los requerimientos no funcionales del mismo (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

3.2.2. Identificación de actores

El sistema desarrollado fue diseñado para dos tipos de actores: ***1) Alumno***, con identificación y contraseña únicas para acceso a la aplicación web y donde podrá registrar información para que sea procesada posteriormente, ***2) Responsabilidad Social***, el cuál debe ser operado por un usuario experimentado (usuario especialista) quien utiliza todas las funciones del sistema, para visualizar los datos cargados por el usuario (alumno).

Tabla 5

Privilegios de actores.

	Alumno	*Responsabilidad Social*
Gestión Usuarios (crear, modificar, eliminar)		***X***
Subir información relevante del proceso de plantación de árboles.	***X***	***X***
Consulta de Información de Arboles	***X***	***X***
Reportes		***X***
Enviar mensajes		***X***

Fuente: Propia

Catalogación de requerimientos

Se analizó la información obtenida en las sesiones de trabajo para la Identificación de Requisitos, definiendo y catalogando los requisitos (funcionales y no funcionales) que debe satisfacer el sistema, indicando sus prioridades.

3.2.3. Requerimientos funcionales

Los requisitos funcionales se refieren a las consultas, procedimientos y/o actividades necesarios para cumplir con los objetivos del proyecto, mostrando la funcionalidad principal que tendrá el sistema, a continuación se detalla cada uno de los requerimientos funcionales del sistema:

RF1. Inicio de Sesión - Usuario

RF1.1 El sistema deberá solicitar al usuario identificarse y autenticarse para poder registrar su sesión activa, con el fin de verificar su ingreso al sistema y resguardar la seguridad del mismo.

RF1.2 Para los alumnos, el inicio de sesión, será su código de alumno y una clave que se proporcionara por defecto, con el primer ingreso al sistema, el sistema le pedirá al alumno que actualice su contraseña por una que este registre.

RF 1.3 Se tendrá diferentes interfaces de Inicio de Sesión, para los dos módulos manejados por el sistema, tanto del alumno, como de Responsabilidad Social.

RF2. Modulo Alumnos – Registro de Actividad Árbol.

RF2.1 El sistema debe permitir el registro de los datos donde realizara la plantación del árbol, llenando los datos, como de la dirección, así mismo el sistema tendrá una opción de ver el mapa, accediendo a la localización en tiempo real del alumno, con la cual una vez se realice el proceso de guardar la información, guardara esa latitud y longitud del mapa, para que pueda ser vista por el área universitaria. El sistema también deberá permitir cargar fotos, para que el alumno pueda registrar las evidencias del proceso realizado.

RF2.2 El sistema solo permitirá cargar los datos de localización una sola vez, para evitar manipulación y falsedad de la data proporcionada.

RF2.3 El sistema debe permitir que el alumno pueda subir nuevas evidencias (fotos), que serán indicadas por el área de Responsabilidad Social.

RF3. Modulo Responsabilidad Social – Seguimiento Proceso Alumnos

RF3.1 El sistema deberá permitir, que los usuarios puedan realizar la búsqueda de los alumnos por medio de su código.

RF3.2 El sistema deberá mostrar los datos del alumno previamente cargados por el mismo, mostrando también un cuadro donde se podrá ver la localización registrada mediante un mapa.

RF3.3 El sistema mostrara un mensaje de que el alumno no realizo el proceso, en caso no se obtenga datos del mismo, para que el área de Responsabilidad Social, pueda realizar las acciones que crean convenientes.

RF3.4 El sistema mostrara la última imagen cargada por el alumno, de igual manera el usuario puede acceder al historial de fotos cargadas al sistema, para visualizar el proceso realizado.

RF3.5 El sistema permitirá al Área de Responsabilidad Social, tener un mejor control del proceso que se realiza, logrando monitorear y tener debidamente almacenado los datos de los alumnos que realicen correctamente la plantación de árboles.

RF3.6 El sistema le brindara al usuario, la opción de llenar una bitácora, con la cual este tendrá registrado las observaciones realizadas, con su respectiva fecha y así ver si el alumno subsano dichas observaciones.

RF4. Modulo Responsabilidad Social – Enviar Mensajes

RF 4.1 El Sistema al momento de registrar la observación en la bitácora, enviara el mensaje al alumno, el cual le llegara a su correo personal, con ello, el alumno estará al tanto en tiempo real, debido a que dicho mensaje le llegara a su teléfono móvil y así subsanar con prontitud la observaciones realizada

RF5. Módulo Responsabilidad Social – Control Usuarios

RF5.1 El sistema podrá agregar, eliminar y cambiar contraseñas de los usuarios, controlando los usuarios que tendrán acceso al mismo.

RF6. Módulo Responsabilidad Social – Página Principal

RF6.1 El sistema mostrara un mensaje de bienvenida, donde les mostrara el menú principal para realizar las acciones anteriormente descritas.

RF7 Interfaz de Usuario

RF7.1 La interfaz de la solución web debe estar basada en ventanas y paneles, permitiendo que se muestren los menús seleccionables dentro de la misma ventana para facilitar la interacción con el usuario.

RF7.2 El sistema debe identificar en la barra de menú superior el usuario que está utilizando el sistema.

RF7.3 El sistema debe mostrar a cada usuario sólo las opciones y datos permitidos de acuerdo a su perfil

RF7.4 El acceso al sistema siempre deberá ser a través de un usuario, y una clave (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

3.2.4. Requerimientos no funcionales

Los requerimientos no funcionales son requerimientos de calidad que, representan las restricciones o las cualidades que el sistema debe tener.

RNF1. Arquitectura de Software

RNF1.1 El sitio web de la aplicación deberá poderse explotar y administrar empleando cualquier navegador web y puede ser consumido desde cualquier dispositivo con acceso a internet.

RNF1.2 El sistema, tendrá el Core funcional del mismo, en Web Services, lo que le brinda mayor escalabilidad y mejor procesamiento de datos, dándole una arquitectura SOA.

RNF2. Seguridad

RNF2.1 Los datos de la aplicación solo podrán ser modificados por aquellas personas autorizadas para ello. Los perfiles de usuario de la aplicación serán los siguientes: administrador de la aplicación web, usuario registrado.

RNF2.2 El sistema cuenta con un acceso de inicio de sesión, el cual solo permite a usuarios registrados acceder al sistema, para evitar que cualquier persona pueda acceder a los datos del sistema.

RNF2.3 La información registrada en el sistema, no podrá ser manipulada, evitando así el mal uso o falsedad de la información proporcionada por parte del alumno.

RNF3 Rendimiento y escalabilidad

RNF3.1 Las peticiones asíncronas (AJAX) que se realicen a la aplicación deberán limitarse para no correr el riesgo de sobrecargar al servidor.

RNF3.2 Las peticiones concurrentes de acceso a la base de datos deben dejar a la aplicación en un estado consistente.

RNF3.3 Al ser una aplicación web, estará disponible las 24h y pueda ser accedida desde cualquier parte, siempre y cuando se tenga un dispositivo con acceso a internet, para ver los mapas con la localización guardada.

3.3. Análisis de factibilidad para implementar de la solución web

Considerando los puntos anteriores se asume que el sistema funcionará cuando esté terminado, ya que no existen barreras que impidan su diseño, pues se dispone del apoyo por parte de los usuarios y de los usuarios finales, por lo tanto la factibilidad operacional está asegurada.

Se considera también que el proyecto es factible técnicamente, pues la tecnología que se empleará existe actualmente en el mercado, básicamente consistirá en la implantación de una arquitectura SOA, los servicios a utilizar serán Servicios Rest, los cuales se comunican mediante el protocolo HTTP y json, haciendo más fluida la comunicación con la aplicación informática, desarrollada en el lenguaje de programación C# .NET mediante tecnología Web, la comunicación entre servicios y el cliente, es independiente del lenguaje en el cual se desarrolle cada uno, ya que los servicios sirven para conectar sistemas externos que pueden estar o no desarrollados en el mismo lenguaje del cliente que los consuma (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

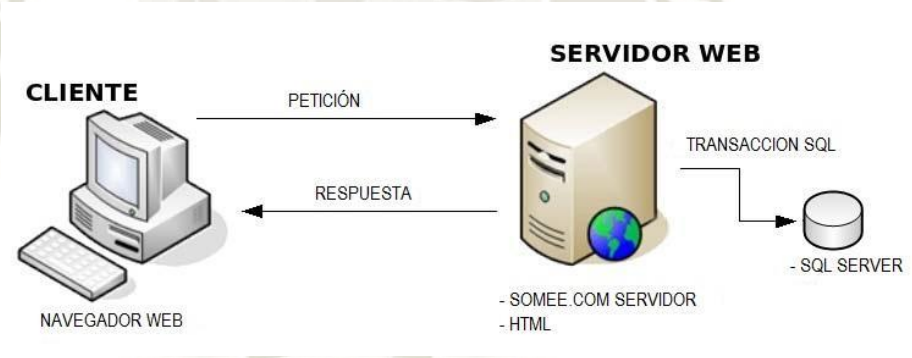

Figura 3.1: Representación de la tecnología usada en el sistema

Fuente: Propia

3.4. Análisis de Casos de Usos

3.4.1. Elaboración del Caso de Uso Iniciar Sesión - Usuario Alumno

Descripción: Permite al alumno acceder al sistema con permisos para el modulo del alumno de cargar datos de plantación de árbol. (Ver Tabla 5)

Actores: Usuario/Alumno.

Precondiciones: El usuario tiene que estar registrado en el sistema.

Flujo de eventos: Diagrama del caso de uso figura 3.2 (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

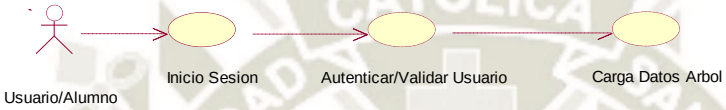

Figura 3.2: Caso de uso inicio sesión usuario/Alumno

Fuente: Propia

3.4.2. Elaboración del Caso de Uso Iniciar Sesión – Responsabilidad Social

Descripción: Permite al usuario de Responsabilidad Social acceder al sistema con permisos para los módulos de seguimiento alumnos, así como el modulo principal. (Ver Tabla 5)

Actores: Responsabilidad Social.

Precondiciones: El administrador tiene que estar registrado en el sistema y contar con privilegios de administrador.

Flujo de eventos: Diagrama del caso de uso figura 3.3 (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

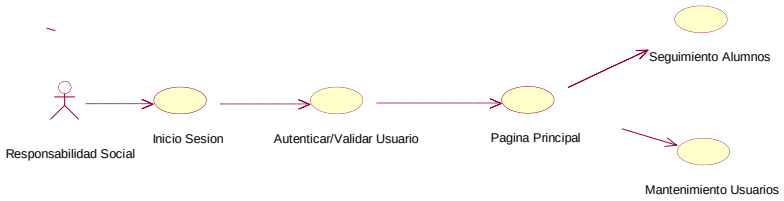

Figura 3.3: Caso de uso inicio sesión Responsabilidad Social

Fuente: Propia

3.4.3. Elaboración del Caso de Uso – Registro Datos Plantación Árbol

Descripción: Permite al alumno, poder cargar su proceso de plantación de árboles, como la dirección donde lo hizo, la localización mediante un mapa y la evidencia mediante una fotografía.

Actores: Alumno.

Precondiciones: El alumno debe estar previamente registrado en la Base de Datos, si el alumno ya cargo anteriormente los datos, el sistema mandara un mensaje de que ya realizo el proceso y solo permitirá subir una nueva foto, debe tener una conexión con acceso a internet, para visualizar el mapa de localización.

Flujo de eventos: Diagrama del caso de uso figura 3.4 (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

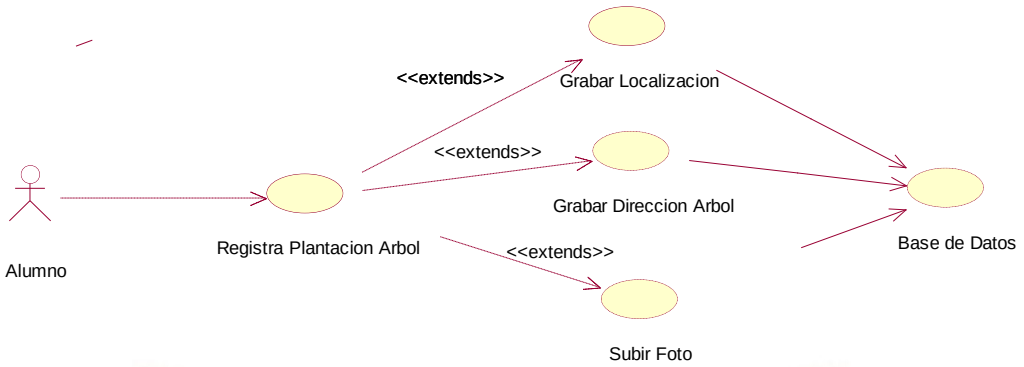

Figura 3.4: Caso de uso Registra Plantación Árbol

Fuente: Propia

3.4.4. Elaboración del Caso de Uso – Seguimiento Alumnos

Descripción: Permite al usuario de Responsabilidad Social, poder realizar el seguimiento del proceso de plantación de árboles de cada alumno, viendo las fotos que estos cargan y la localización donde lo hicieron, en caso deseen corroborar con una visita.

Actores: Responsabilidad Social.

Precondiciones: El usuario debe estar registrado en el sistema y debe acceder al sistema, que tenga conexión a internet para visualizar el mapa de localización.

Flujo de eventos: Diagrama del caso de uso figura 3.5.

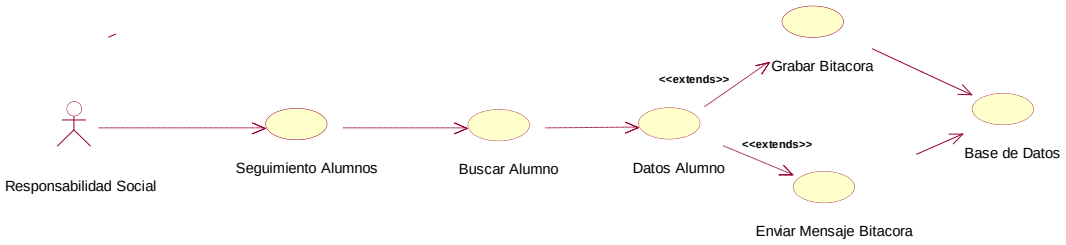

Figura 3.5. : Caso de uso Seguimiento Alumnos.

Fuente: Propia

3.4.5. Elaboración del Caso de Uso – Mantenimiento Usuarios

Descripción: Permite al usuario Responsabilidad Social, crear o eliminar usuarios, así como también cambiar de contraseña a usuarios existentes.

Actores: Responsabilidad Social.

Precondiciones: El usuario de Responsabilidad Social tiene que estar registrado en el sistema.

Flujo de eventos: Diagrama del caso de uso figura 3.6.

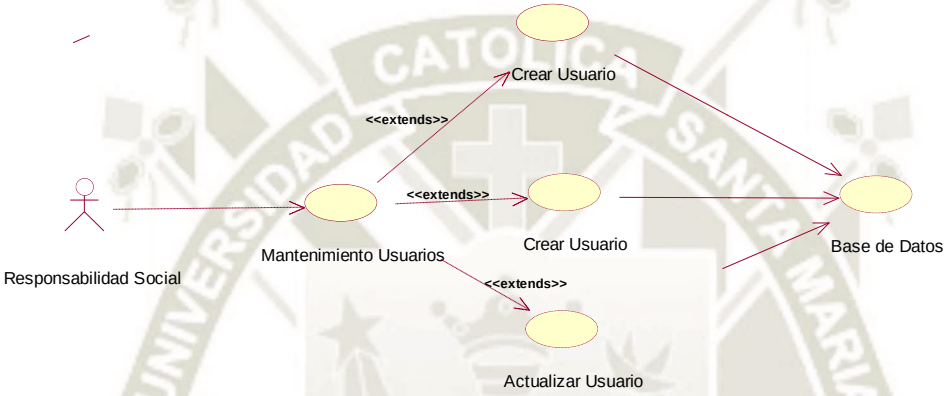

Figura 3.6: Caso de uso Mantenimiento Usuarios.

Fuente: Propia

CAPITULO IV

DISEÑO, DESARROLLO E IMPLEMETACION DE LA SOLUCIÓN WEB

4.1. Arquitectura de Sistema

4.1.1. Arquitectura Lógica del Sistema

Uno de los patrones arquitectónicos más usados en la actualidad para el desarrollo de aplicaciones web es el denominado “MVC”, el cual consiste en tener independizado el Modelo, la Vista y el Controlador.

El Modelo, son las entidades que maneja la lógica de negocio, la Vista, son las páginas que interactúan con el usuario y el Controlador, se encarga de enlazar los datos del Modelo y mostrarlos en la Vista.

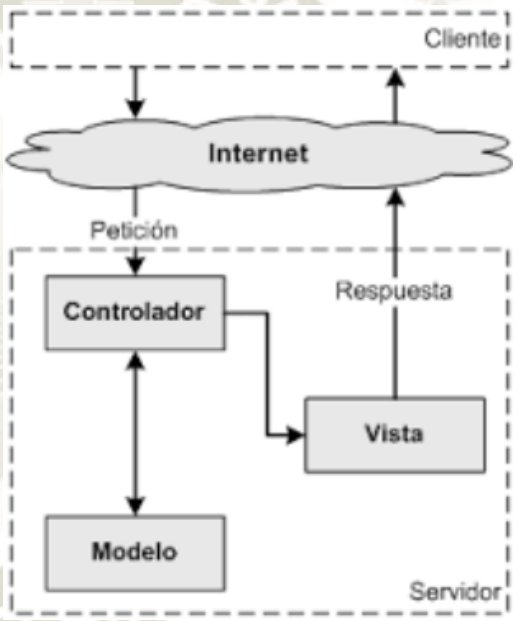

Figura 4.1: Patrón de Diseño MVC

Fuente: Propia

El Modelo, es el que tiene las entidades del negocio, es decir, las cuales son una réplica a las de la Base de Datos, con las cuales manejaremos los datos obtenidos desde el servidor, para poder mostrarlos en la vista y así lograr la persistencia y obtención de datos.

La Vista, se encarga es la parte visual del software, donde se encuentran las interfaces e interactúa con el usuario, es desde aquí que el usuario realiza peticiones al servidor, para que posteriormente se muestren al usuario.

El controlador, se encarga de enlazar la Vista con el Modelo procesando las peticiones que realicen el usuario, realizar la lógica de negocio necesaria para el procesamiento de los datos, enlazarlos a las entidades del modelo y pasarlos a la Vista, para que pueda mostrarle al usuario mediante su interfaz gráfica.

La aplicación web a desarrollar, tendrá el patrón arquitectónico MVC, para así mantener un mayor desacoplamiento de las partes y poder desarrollar independientemente cada una, así tener una mejor calidad y respuesta más fluida con el usuario, logrando una mayor calidad del software, para mejorar la interacción con el usuario, se utilizara JavaScript, HTML, CSS3, AJAX, con el fin de tener una usabilidad sencilla y una mejor experiencia para el usuario.

Para el presente desarrollo se optó por realizar una Arquitectura basada en Servicios (SOA), para así lograr una mayor escalabilidad con el sistema, logrando tener la lógica de negocios en los Servicios, con lo cual se podrían integrar dichos servicios con otros desarrollos en diferentes plataformas, esto también permite desarrollar los servicios o funcionalidades independientemente, logrando un bajo acoplamiento entre sus componentes, con ello el mantenimiento del software, será más sencillo y se podrán agregar más funcionalidades sin cambiar mucho del software.

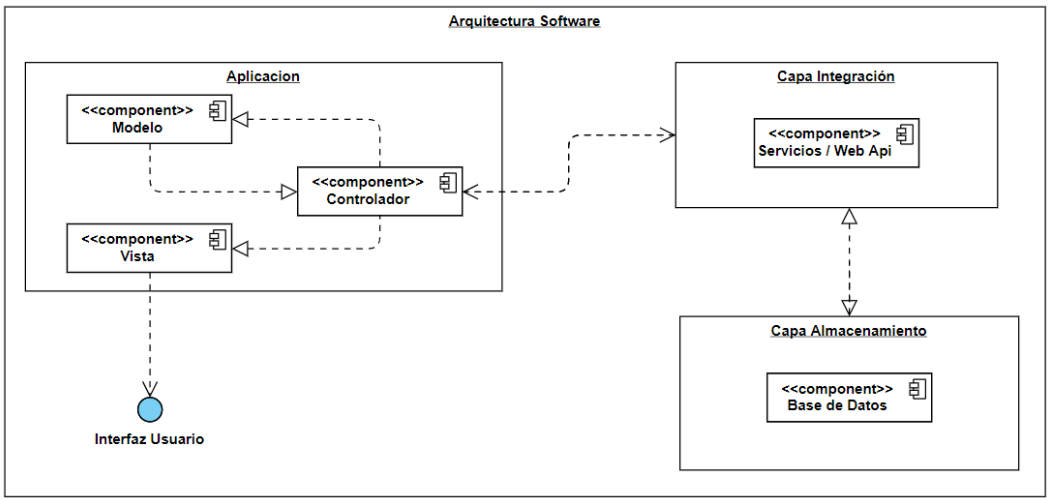

Figura 4.2: Arquitectura de la Aplicación Web

Fuente: Propia

La capa Aplicación, tiene como objetivo la interacción con el Usuario, es la aplicación con sus funcionalidades, los requerimientos funcionales definidos anteriormente, dichas funcionalidades están separadas en los diferentes módulos que presenta la aplicación, estos módulos serán presentados en páginas, las cuales se le muestra al usuario para su mejor interacción y procesamiento de datos, los módulos identificados de la aplicación y sus interdependencias, son los siguientes:

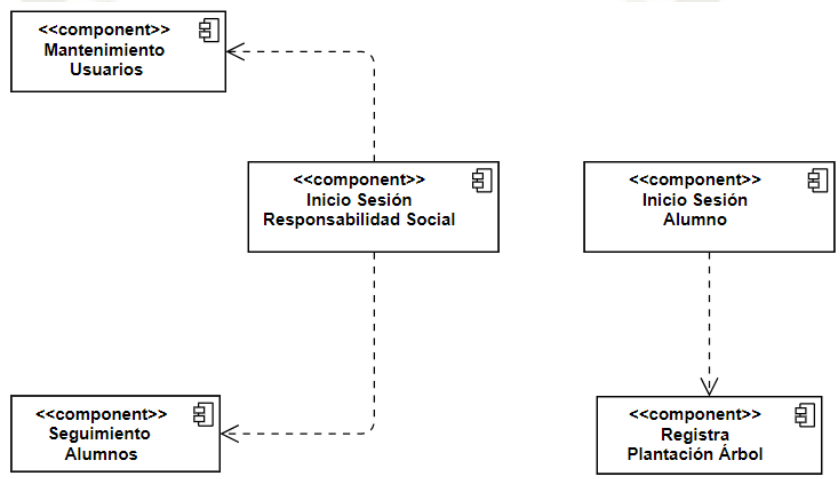

Figura 4.3: Módulos Aplicación

La capa de Integración, es donde se encuentran los servicios, en dichos servicios esta la lógica de negocios que utiliza el sistema, estos servicios se usan desde los módulos antes descritos, mediante el controlador hace la llamada a los servicios web, enviando los datos de la solicitud y este responde para su respectivo procesamiento, a continuación se detallan los servicios utilizados en la aplicación:

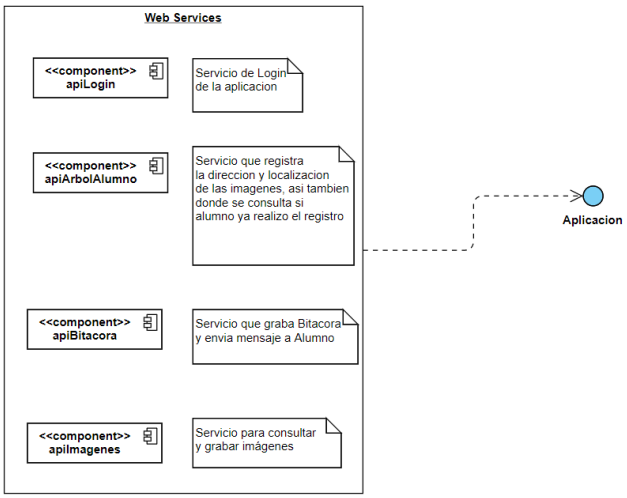

Figura 4.4: Diagrama de Servicios

Fuente: Propia

La capa acceso a datos, es la encargada de la persistencia de los datos, almacena en la Base de Datos, toda la información ingresada por el alumno, dirección del árbol plantado, geo localización (latitud y longitud) representada por un mapa y las fotos, para que posteriormente el Área de Responsabilidad Social, pueda validar y verificar los datos y con ello tener mejor controlado, dando un mejor seguimiento de los alumnos que realicen el correcto proceso de la actividad plantación de árboles.

4.1.2. Arquitectura Física del Sistema

Se describirá la topología del sistema, mostrando como será asignado en forma física la aplicación web (software) a los diferentes equipos de computación (hardware). La arquitectura física del sistema es representada con la siguiente figura.

Figura 4.5: Arquitectura Física del Sistema

Fuente: Propia

Esta arquitectura física considera la distribución de la aplicación web en dos tipos de nodos: Cliente y Servidor. El primer nodo representa a las estaciones de trabajo de los usuarios finales. El nodo Servidor representa al equipo en donde correrá la aplicación web y el alojamiento de la base de datos (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

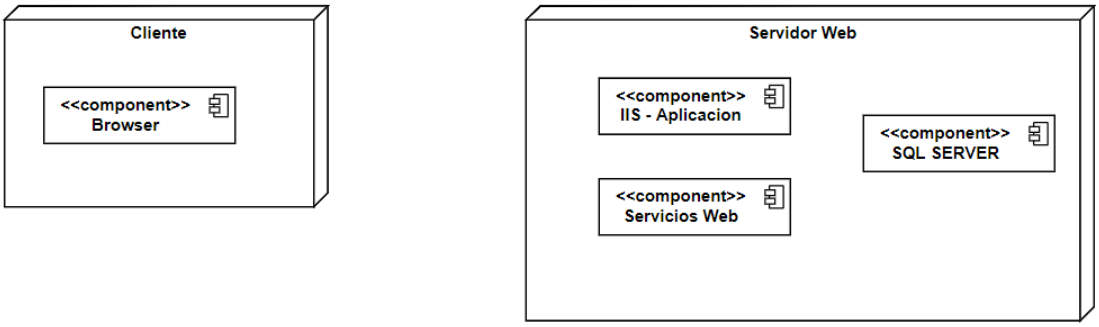

Figura 4.6: Diagrama de Despliegue de la Arquitectura Física del Sistema

Fuente: Propia

4.2. Diseño de la Base de Datos

Esta sección describe los diseños de la base de datos que determinan como los datos que van a ser incluidos en la base de datos, están lógica y físicamente organizados. Para ello, el diseño de la base de datos se establece en dos niveles de detalle.

El Primer nivel, muestra cómo están representados los datos en la aplicación, esto se logra mediante las entidades y gráficamente será expuesto mediante el modelo Entidad Relación. El segundo nivel, contempla el modelo de la Base de Datos, y el mapeo Objeto Relacional que se tendrá para acceder y registrar los datos (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

4.2.1. Esquema conceptual de la base datos

Para poder realizar una buena persistencia y almacenamiento de datos, se deben tener las entidades adecuadas, las cuales representaran las tablas que se tendrá en base de datos y así tener un mejor control de los datos en la aplicación, a continuación se representara las entidades mediante un diagrama entidad relación, como se muestra en la Figura 4.8.

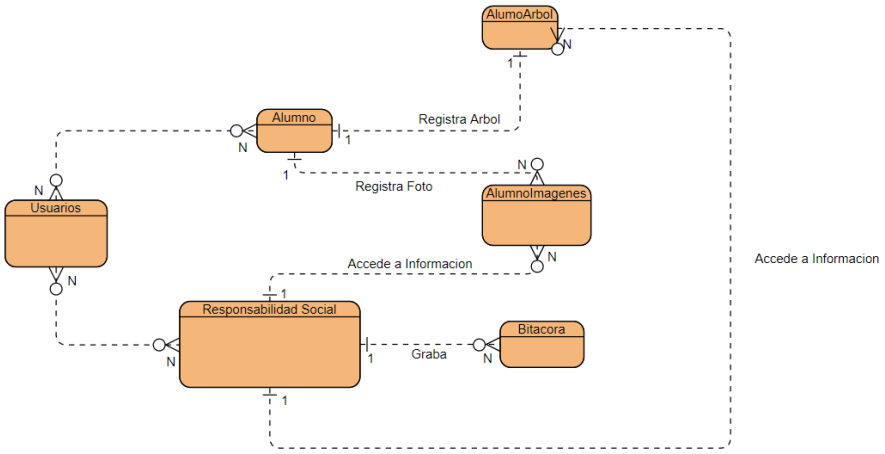

Figura 4.7: Modelo Entidad – Relación del Sistema Plantación Arboles.

Fuente: Propia

4.2.2. Esquema Implementación de la Base de Datos

Como se mencionó anteriormente, las entidades generadas en la aplicación, son las que representan a las tablas de la Base de Datos, se muestra las tablas generadas para el Sistema Plantación de Árboles, el cual estará ejecutándose en el gestor de Base de Datos SQL Server, como se muestra en la Figura 4.9.

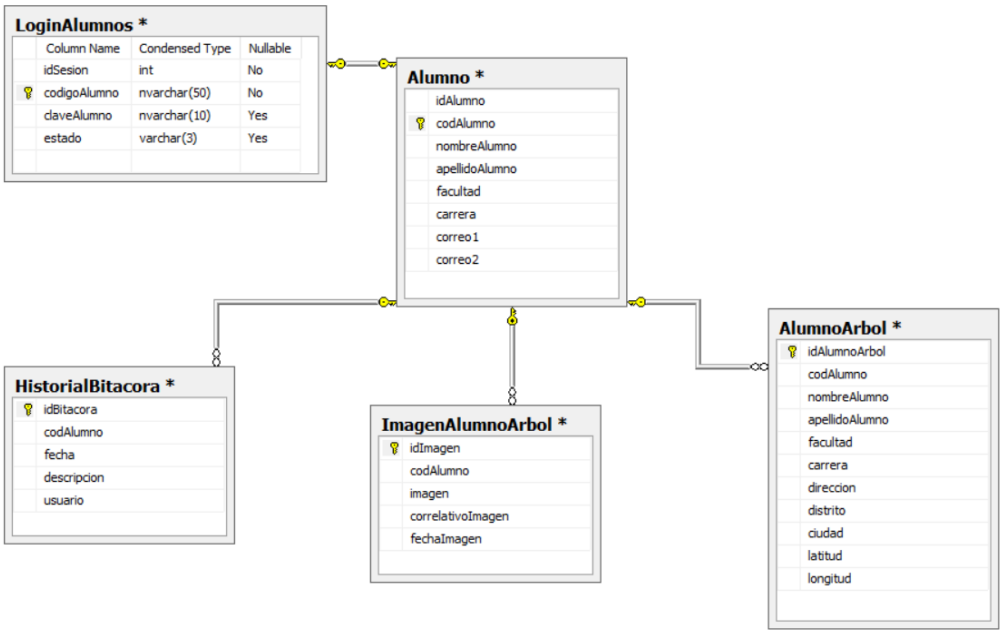

Figura 4.8: Modelo Físico BD del Sistema Plantación de Árboles.

Fuente: Propia

4.3. Mapeo Objeto Relacional

Se crearon clases entidades, las cuales tenían las mismas características de datos que las tablas de la Base de Datos y así poder manejar los datos por medio de los objetos de las entidades creadas, para tener un mejor control del proceso y mejorar la optimización de la aplicación, se optó por los Stored Procedures, mediante los cuales, se realizaran la interacción con la Base de Datos, ello nos permite manejar mejor los errores y tener una mejor performance, debido a que se tendrá en el servidor los procesos que se requiera, a continuación se muestra algunos los Stored Procedures que se crearon.

Se observa una de las entidades creadas en la web Api, para tener el control de los datos que se obtengan desde la BD la Figura 4.10

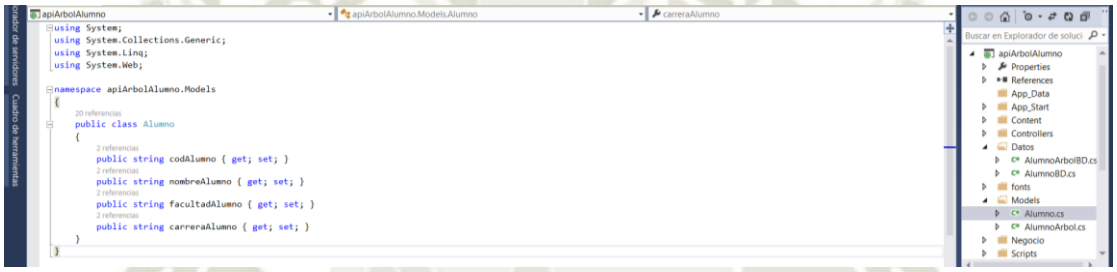

Figura 4.9: Entidad Alumno.

Fuente: Propia

Para poder enlazar los datos obtenidos con los modelos creados en el sistema, se crea el objeto del modelo y se almacenan los campos obtenidos en la entidad, como se muestra en la Figura 4.11

Figura 4.10: Entidad enlazada a BD.

Fuente: Propia

4.4. Diseño de las Interfaces

La interacción con el sistema es a través de un navegador Web (Browser), las interfaces generadas por el Sistema de Plantación de Árboles, son representadas por el Browser, como paginas HTML, el cual se interpreta de distintas formas en los diferentes browsers, con el fin de mejorar la interacción con el usuario y tener una página más atractiva, se utilizaron Hojas de estilo (CSS), con ello mejoramos la visibilidad de las páginas y también se agregan algunas animaciones para mejorar la interacción con el Usuario.

4.4.1. Diseño de Interfaz de Inicio de Sesión

En la figura 4.12, se muestra la imagen de Inicio de Sesión, para el módulo de los alumnos, en donde se podrá ingresar un Usuario y Contraseña, para así poder ingresar al sistema.

Figura 4.11: Interfaz Inicio Sesión Alumno.

Fuente: Propia

En la figura 4.13, se muestra la imagen de Inicio de Sesión, para el módulo de Responsabilidad Social de la UCSM, en donde se podrá ingresar un Usuario y Contraseña, para así poder ingresar al sistema.

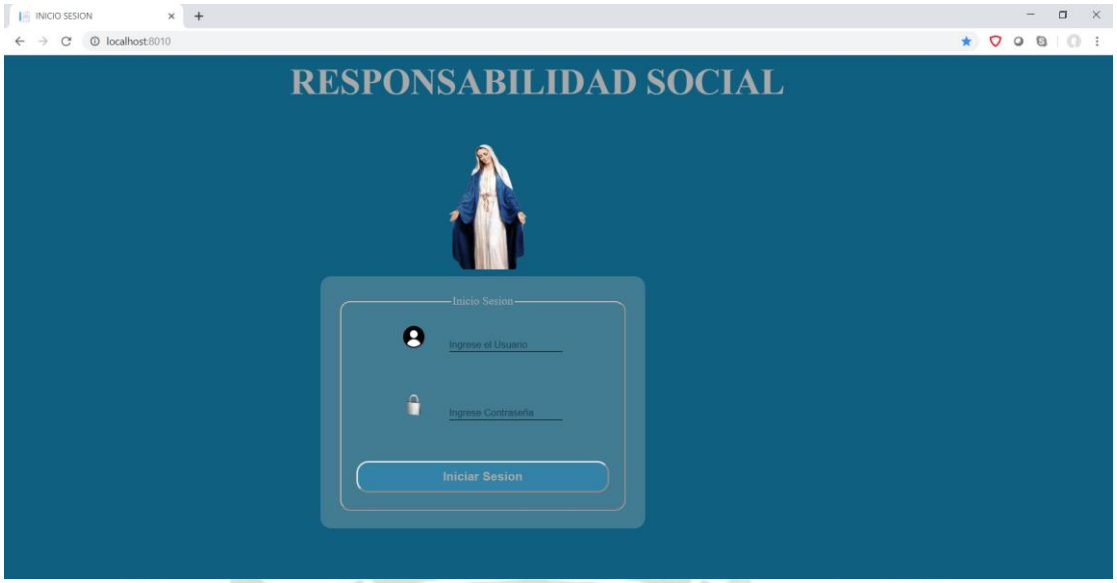

Figura 4.12: Interfaz Inicio Sesión Responsabilidad Social.

Fuente: Propia

4.4.2. Diseño de Interfaz Nivel Usuario

4.4.2.1. Diseño de Interfaz Actualizar Clave - Alumno

En la Figura 4.14, una vez que el alumno ingrese su usuario y clave secreta por defecto que se le brindo (ingreso primera vez), el sistema re direccionara al alumno, a la página para que pueda actualizar su clave y ponga una nueva de su elección.

Figura 4.13: Interfaz Actualizar Clave Alumno.

Fuente: Propia

4.4.2.2. Diseño de Interfaz de Ingreso Datos Árbol – Alumno

En la Figura 4.15, una vez que el alumno ingrese su usuario y clave secreta, el sistema re direccionara al alumno, a la página para que registre los datos del árbol plantado, teniendo la sección donde podrá registrar la dirección y la otra sección donde se mostrara el mapa de la ubicación, la cual el sistema geo localizara la ubicación desde donde se haga el registro del árbol.

Figura 4.14: Interfaz Registrar Datos Árbol Alumno.

Fuente: Propia

En la Figura 4.16, que es parte de la vista anterior, pero, esta sección es la parte donde se puede realizar la carga de la foto, que servirá como evidencia para registrar la plantación del árbol o posteriormente ingresar las nuevas evidencias que se registre del cuidado y crecimiento del árbol

Figura 4.15: Interfaz Registrar Datos Árbol Alumno.

Fuente: Propia

4.4.2.3. Diseño de Interfaz Inicio – Responsabilidad Social

En la Figura 4.17, se muestra la página principal del módulo de Responsabilidad Social, después de inicio de sesión, se llega a esta página,

donde se tiene un menú en la parte superior, con el cual se podrá acceder al seguimiento de alumnos y configuraciones (claves y usuarios).

Figura 4.16: Interfaz Inicio – Responsabilidad Social.

Fuente: Propia

4.4.2.4. Diseño de Interfaz Seguimiento Alumnos – Responsabilidad Social

Una vez que se ingrese a la opción "SEGUIMIENTO ALUMNOS", se muestra la opción para realizar la búsqueda del alumno a consultar, dicha consulta será por medio del código del alumno, una vez realizada la búsqueda, el sistema muestra el resultado obtenido, mostrando los datos del alumno (Nombres/Facultad/Carrera), para así poder acceder a su información como se muestra en la Figura 4,18.

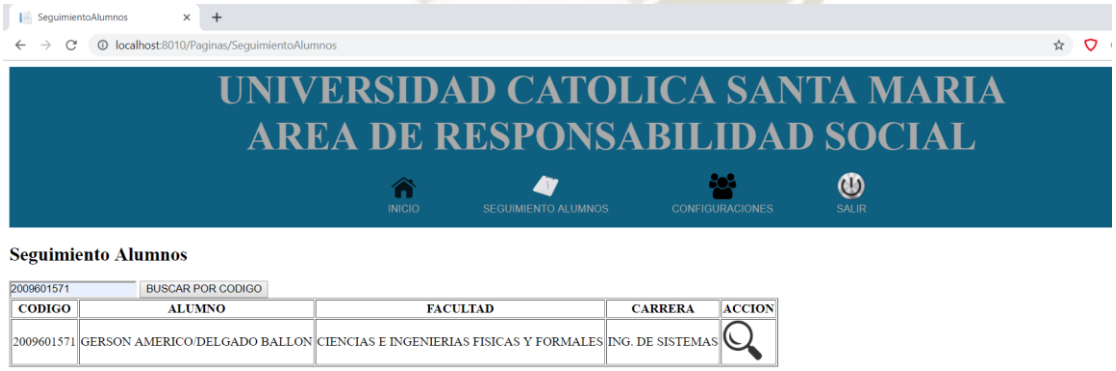

Figura 4.17: Interfaz Seguimiento Alumnos – Responsabilidad Social.

Fuente: Propia

Una vez que se encuentre el alumno, se accede a ver la información, donde se muestra lo que el alumno registro al plantar su árbol, se muestra en una

primera sección la información del alumno, nombres, facultad y carrera del alumno; en la segunda sección, se muestra la dirección referencial donde se plantó el árbol; en la tercera sección, se visualiza mediante un mapa, el lugar donde fue registrada dicha información, para así tener una mejor referencia donde se realizó la plantación del árbol, como se observa en la Figura 4.19.

Figura 4.18: Interfaz Datos Alumnos – Responsabilidad Social.

Fuente: Propia

En la cuarta sección, inicialmente, se muestra la última imagen que fue registrada, pero, donde existe un botón para ver el historial de imágenes subidas al sistema, donde se mostrara la cantidad de imágenes que se tienen grabadas y de igual manera ir navegando por cada una de ellas, para poder visualizar las imágenes, mostrando la fecha en que fue registrada para tener mayor evidencia de la misma y en la quinta sección, se observa una bitácora, donde se podrá ver el historial de la misma, para que se tenga información de las observaciones que se habían realizado y validar si fueron subsanadas, de igual manera se puede registrar un mensaje en bitácora, el cual llegara al alumno a sus correo personal, el cual podrá revisar desde su teléfono móvil, para que este esté alertado de alguna observación que se realice y pueda subsanar el mismo, como se observa en la Figura 4.20.

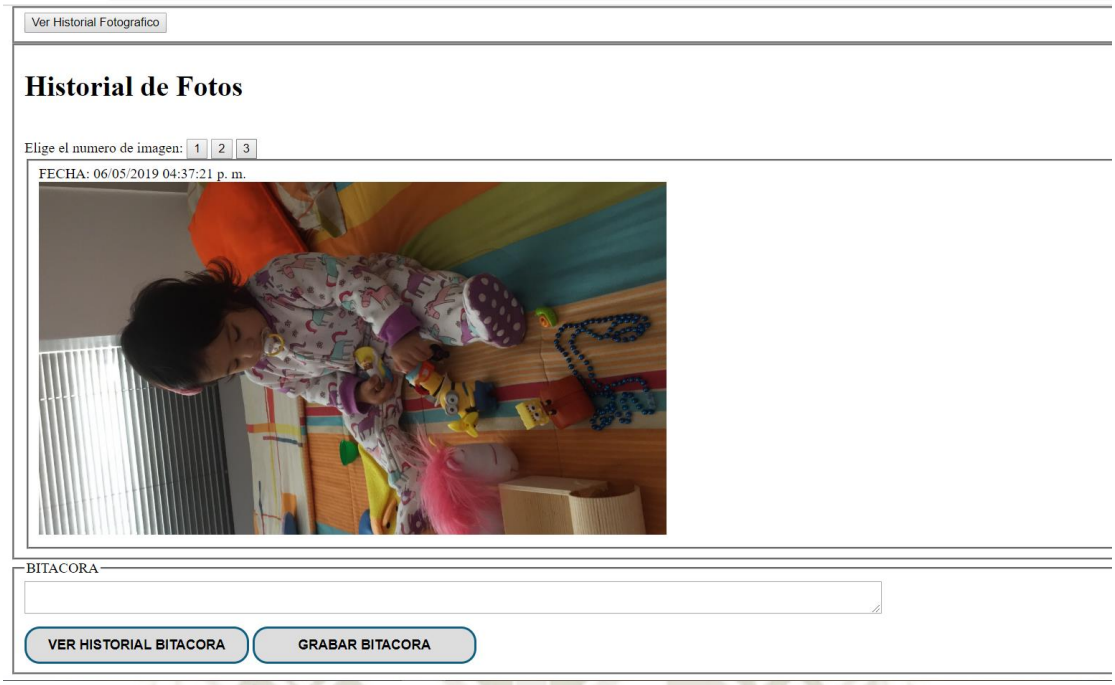

Figura 4.19: Interfaz Seguimiento Fotos Alumnos – Responsabilidad Social.

Fuente: Propia

4.4.2.5. Diseño de Interfaz Configuraciones – Responsabilidad Social

En la Figura 4.20, se observa las opciones que se tiene para la configuración de Usuarios, teniendo opción a ingresar nuevos usuarios o modificar/eliminar usuarios.

Figura 4.20: Interfaz Configuraciones– Responsabilidad Social.

Fuente: Propia

En la Figura 4.21, se observa el modulo para ingresar usuarios, siendo un formulario básico, para poder ingresar un usuario, password y el rol que tendrá para así poder tener acceso a la opción que requiera su rol.

Figura 4.21: Interfaz Mantenimiento Usuarios– Responsabilidad Social.

Fuente: Propia

En la Figura 4.22, se observa el modulo para actualizar o eliminar usuarios, donde inicialmente nos muestra los usuarios activos que tiene el sistema, para así poder acceder al usuario especifico que se elija y dar mantenimiento al mismo, como cambiar clave o el rol que se tiene.

Figura 4.22: Interfaz Modificación Usuario – Responsabilidad Social.

Fuente: Propia

En la Figura 4.23, se observa el formulario que se muestra para actualizar o eliminar el usuario que se ha elegido en la vista anterior, en caso de actualizar, solo se podrá modificar el Password o el Rol del usuario.

Figura 4.23: Interfaz Actualiza/Elimina Usuario– Responsabilidad Social.

Fuente: Propia

4.4.2.6. Diseño de Estadísticas – Responsabilidad Social

El software tiene el módulo de Estadísticas, mediante la cual se podrá observar la cantidad de alumnos que culminaron su proceso, aquellos que no completaron el proceso y aquellos que no realizaron el proceso, el filtro que se usa para mostrar las estadísticas es en base al año de ingreso del alumno, que de acuerdo al código del mismo, se puede obtener el año de ingreso, de acuerdo a ello se realiza la diferencia con el año presente y todos aquellos que al menos tengan mayor o igual a 5 años serán incluidos en las estadísticas, para validar que el proceso está completo, se obtiene la fecha de su última foto, la cual deberá ser como mínimo el presente año, como se muestra en la Figura 4.24.

Figura 4.24: Estadísticas Proceso – Responsabilidad Social.

Fuente: Propia

4.5. Desarrollo del Sistema

Se expondrá la manera en la cual ha sido desarrollada la aplicación web, basando el trabajo en el contexto tecnológico previamente definido, explicando cada uno de los puntos de desarrollo fundamentales para el funcionamiento de la aplicación (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

4.5.1. Desarrollo de la Base de Datos

Para, un mejor procesamiento y manejo de datos, se realizaron las interacciones con BD a través de Stored Procedures, para así tener lógica de procesamiento en el mismo servidor y optimizar consultar aligerar los procesos para una mejor respuesta y performance del sistema.

Para realizar la inserción de los datos que el alumno proporciono de su árbol, se creó el siguiente Stored Procedure: sp_InsertaAlumnoArbol, como se muestra en la Figura 4.25.

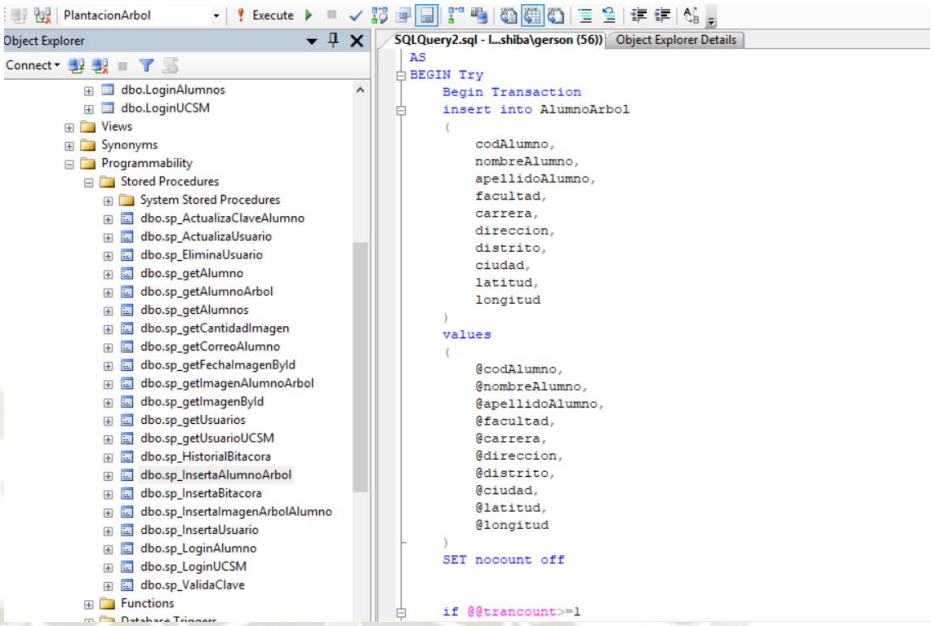

Figura 4.25: Stored Procedure Registra Alumno Árbol.

Fuente: Propia

Para Insertar las imágenes de los alumnos, se creó el Stored Procedure: sp_InsertarImagenArbolAlumno, como se muestra en la Figura 4.26.

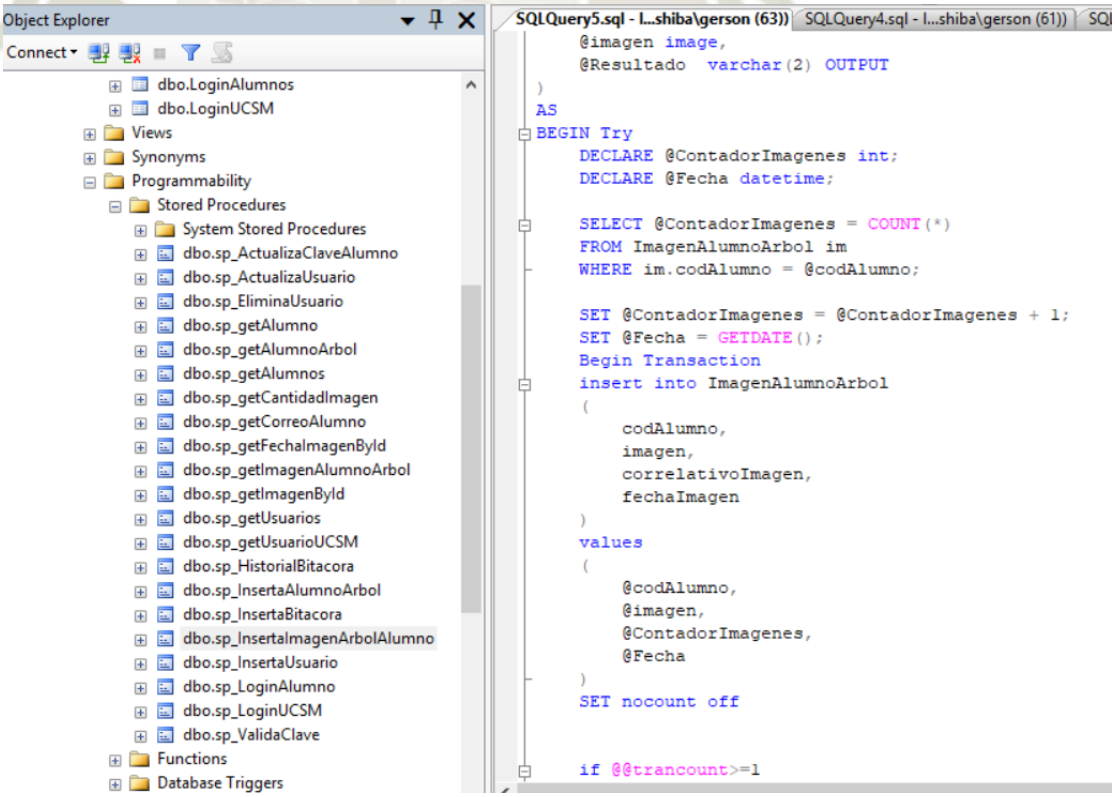

Figura 4.26: Stored Procedure Registra Imagen Alumno Árbol.

Fuente: Propia

4.5.2. Desarrollo de la Aplicación

La siguiente tecnología utilizada es Visual Studio, como se indicó en capítulos anteriores. Inicialmente, se creó un nuevo proyecto de aplicación web vacía de ASP.NET MVC basado en lenguaje C#. Como pasos básicos se definió el nombre "Plantación Arboles" para el proyecto y se definió un fichero local como almacenamiento temporal del proyecto (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

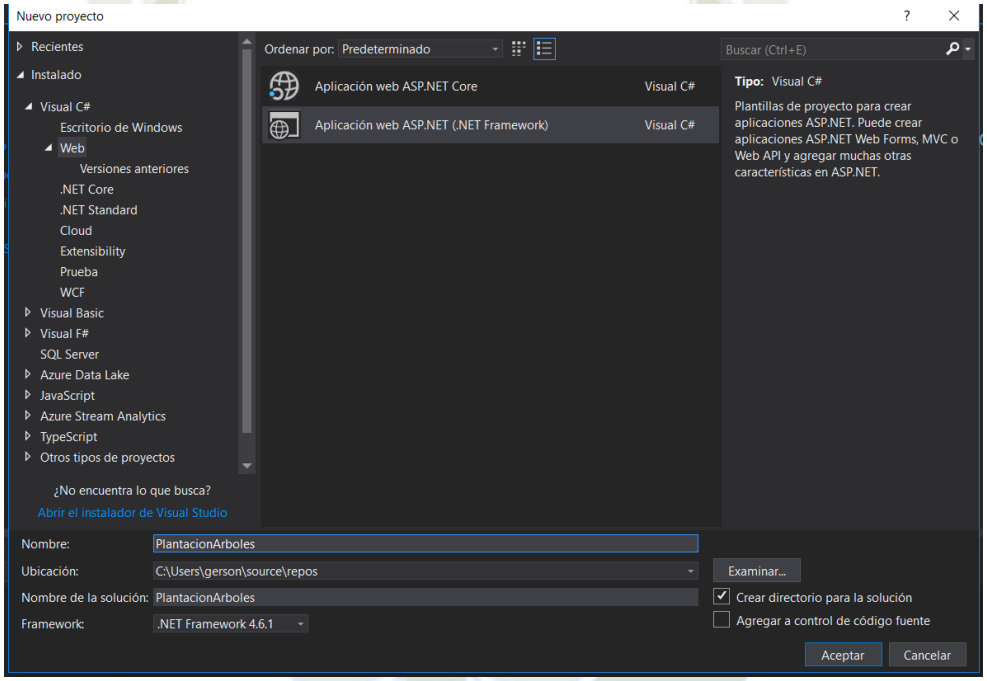

Figura 4.27: Creación de proyecto Web.

Fuente: Propia

Se elige el proyecto MVC, el cual nos creara el sistema de carpetas para poder trabajar con el patrón MVC.

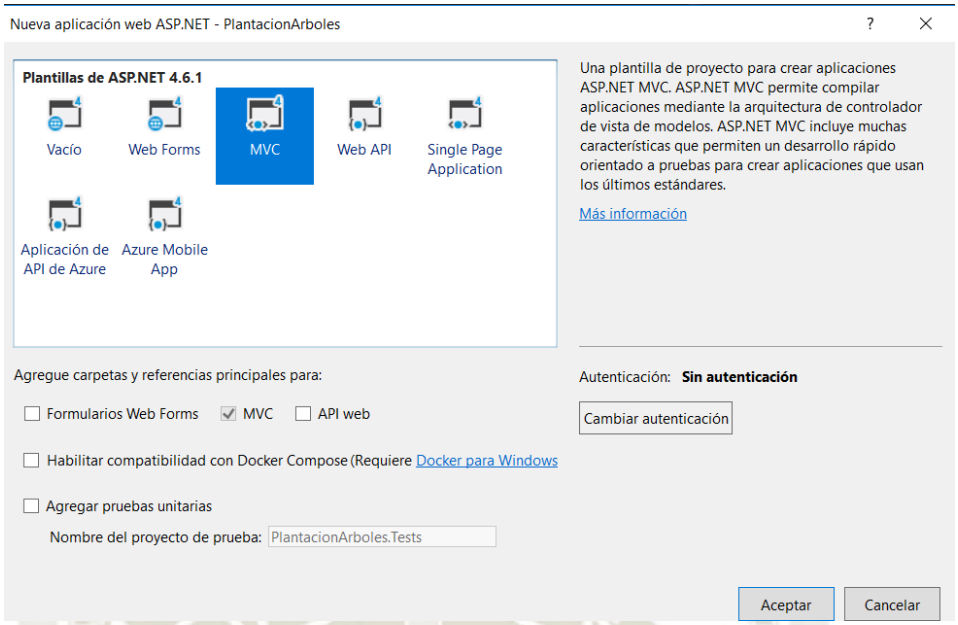

Figura 4.28: Creación de proyecto "Plantación Arboles" Web MVC.

Fuente: Propia

Las interfaces web de proyecto están basadas en una página principal, que al ser un proyecto web MVC, se encuentra en la carpeta "Shared", donde se almacenan las páginas principales y poder uniformizar la vista que tendrán las demás páginas, la página principal tiene por nombre "_Pincipal.cshtml", la cual tiene el formato HTML, como se muestra en la Figura 4,28.

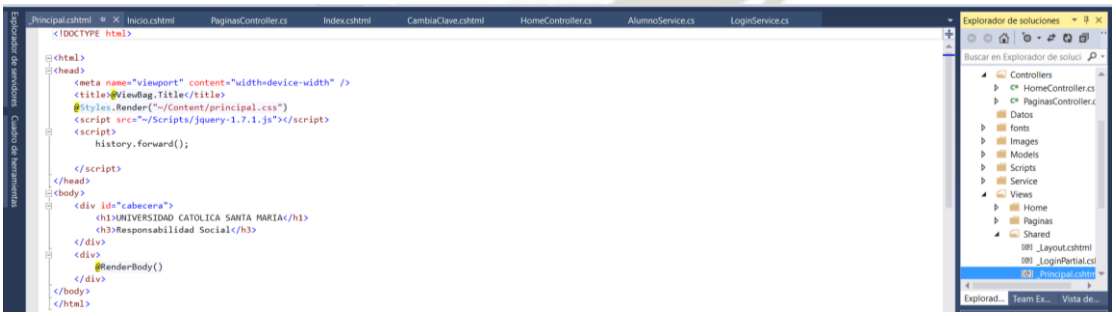

Figura 4.29: Página Principal sistema WEB

Fuente: Propia

Las hojas de estilos CSS en el proyecto web nos permite cambiar en cualquier momento alguna sección o la totalidad del diseño de nuestras páginas (interfaces), con solo modificar el código de nuestra hoja de estilo sin que ello

requiera modificar el contenido como tal. Enlazando la hoja de estilos a todas las páginas maestras se agiliza el proceso de diseño y minimizamos el trabajo (Henríquez Fierro, E., & Zepeda González, M. I, 2003).

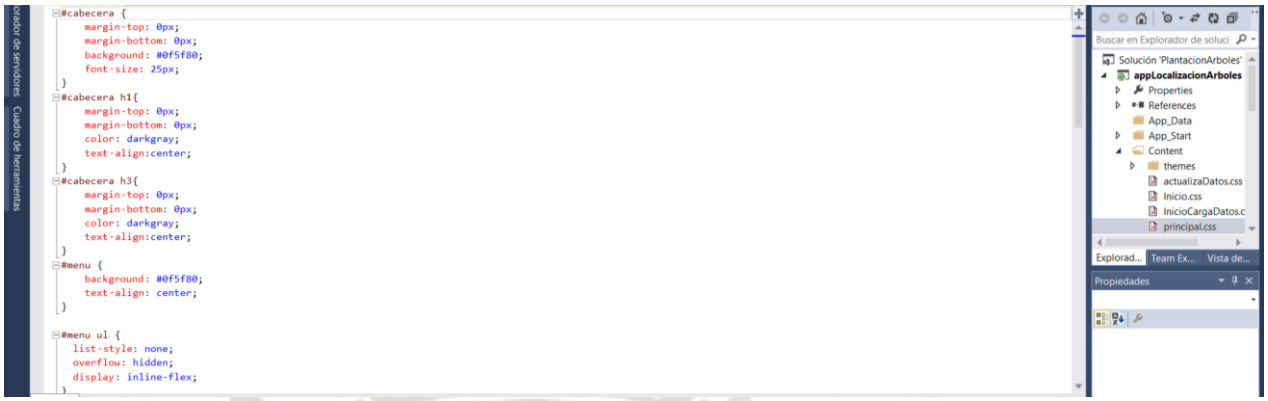

Figura 4.30: CSS de Página Principal

Fuente: Propia

La estructura que maneja el proyecto MVC, se establece por medio de carpetas, donde se tiene la carpeta **Controllers** (donde se encuentran los controladores que realizan peticiones con el servidor enviando los datos para que muestre la vista), la carpeta **Models** (donde se encuentran las entidades donde se almacenaran los datos obtenidos desde la BD) y la carpeta **Views** (donde se tienen las vistas que se mostraran en el browser y es la que interactúa con el usuario), como se fe en la Figura 4.31.

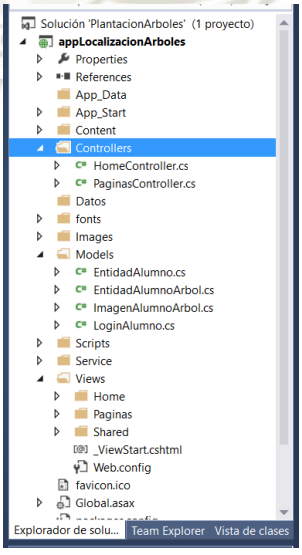

Figura 4.31: Estructura de Carpetas proyecto MVC

Fuente: Propia

Debido a que la presente Web, tiene la Arquitectura Orientada a Servicios, se tiene la Carpeta “Service”, donde se encuentran la conexión a los servicios enviando los datos de entrada al servicio y recibiendo la respuesta por medio de Json, serializando el resultado para que se almacene en las entidades creadas, como se observa en la Figura 4.32.

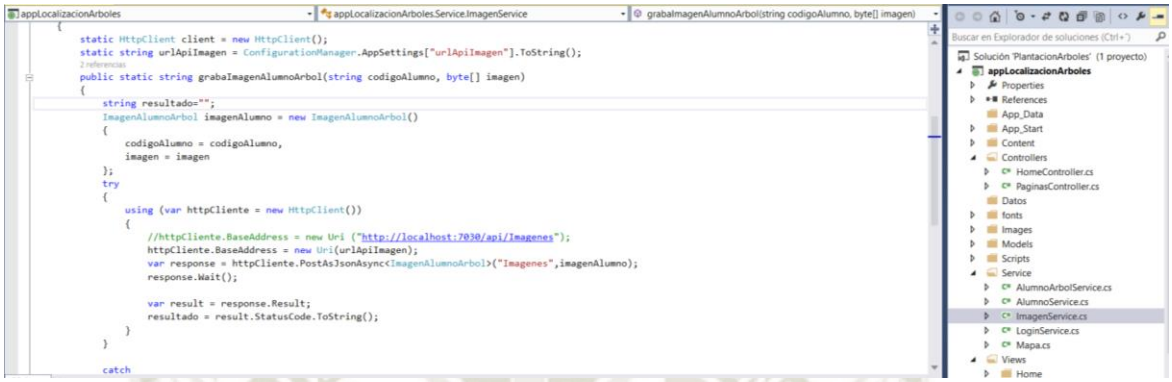

Figura 4.32: Conexión de Web Service

Fuente: Propia

Para la creación de lo Web Services, se optó por crear Web Api, que son Servicios Rest, basados en el protocolo HTTP y así poder realizar conexión con las diferentes plataformas mediante el protocolo HTTP, el proyecto se crea como una aplicación ASP.NET, pero con WEB API, el cual tiene la estructura de carpetas MVC y se desarrolla bajo el mismo patrón.

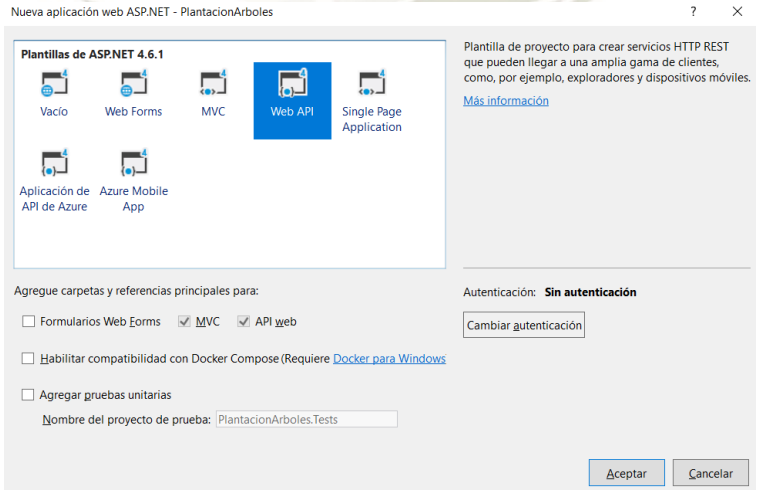

Figura 4.33: Creación Proyecto WEB API

Fuente: Propia

En la Web Api, la lógica la recibe el Controlador, donde se realiza los procesos e interacción con la BD y dicho controlador es el que responde a la aplicación que lo consulta, como se observa en la Figura 4,34.

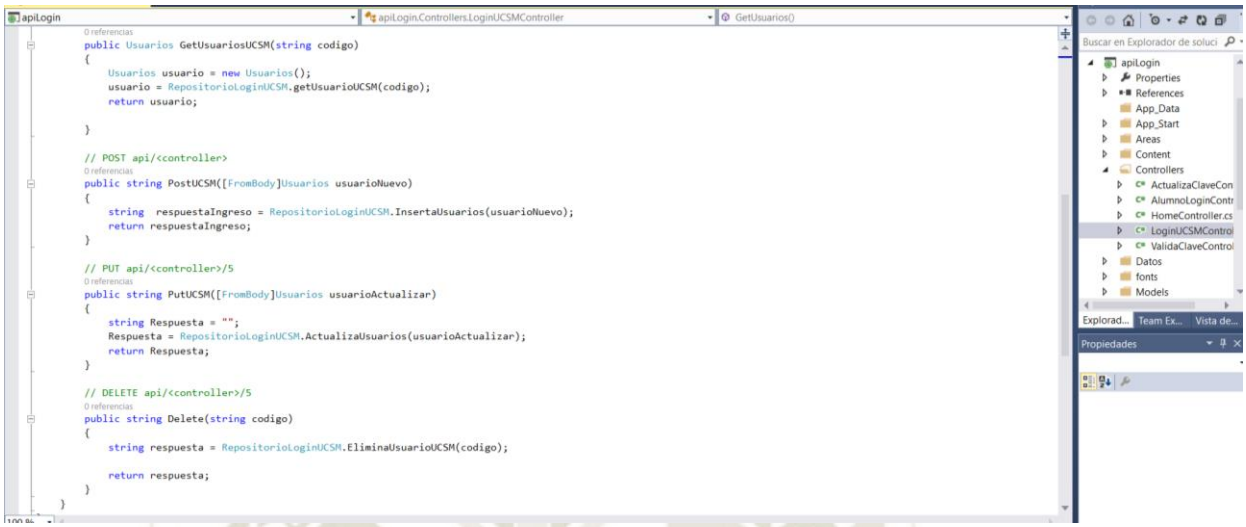

Figura 4.34: Estructura apiLogin

Fuente: Propia

A continuación se muestra como se encuentra desplegada la aplicación web, en el servidor de aplicaciones Azure, donde se podrá observar las veces que se accede a la página web, así como también las veces que la misma pueda tener un error de conectividad, en dicha plataforma también se encuentra desplegada la Base de Datos, esto debido a que azure nos proporciona de forma segura y con alta latencia de respuesta el alojamiento de aplicaciones y base de datos.

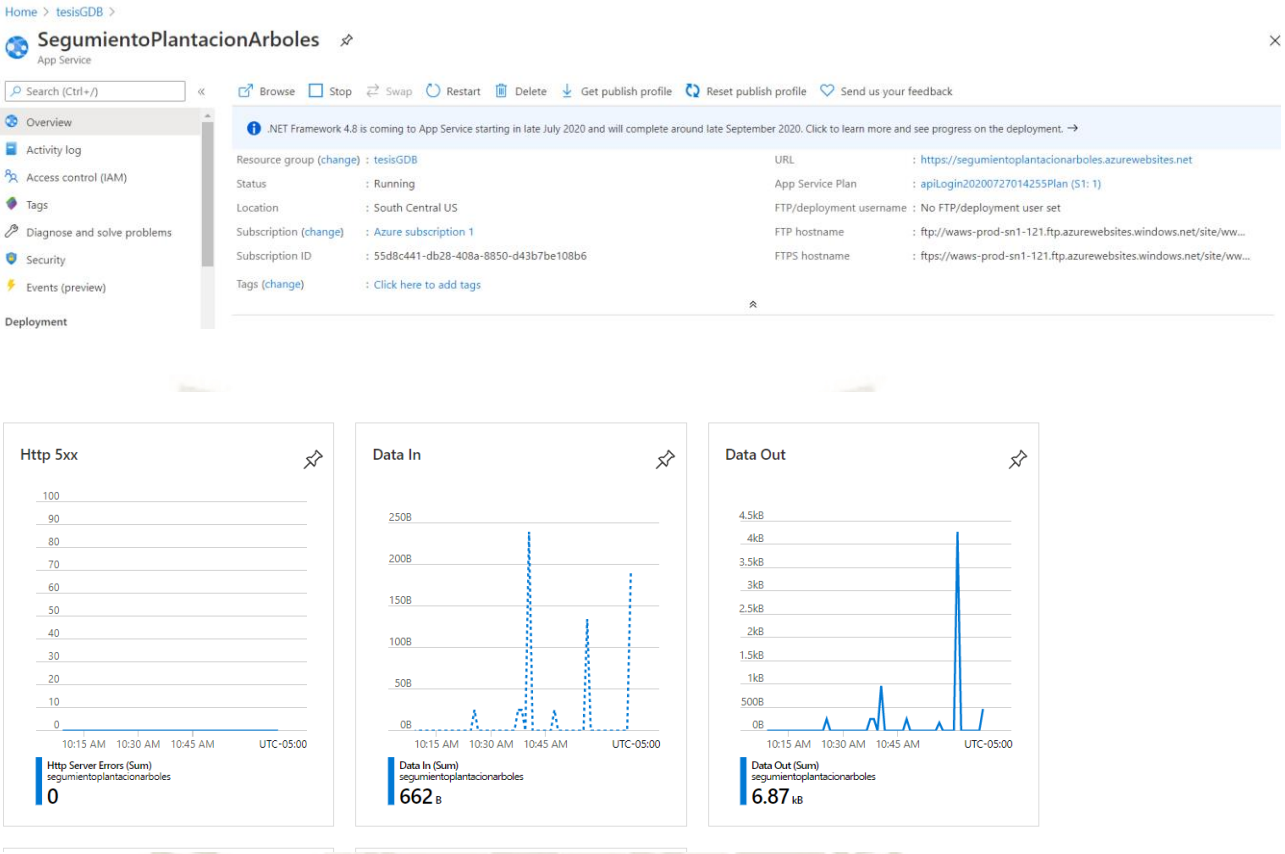

Figura 4.35: Despliegue Web en Azure

Fuente: Propia

Así como se muestra la información de la web desplegada, también podemos observar como se encuentra nuestra base de datos, brindándonos información sobre la capacidad del servidor y la cantidad usada, para así tener un mejor control y optimización de base de datos.

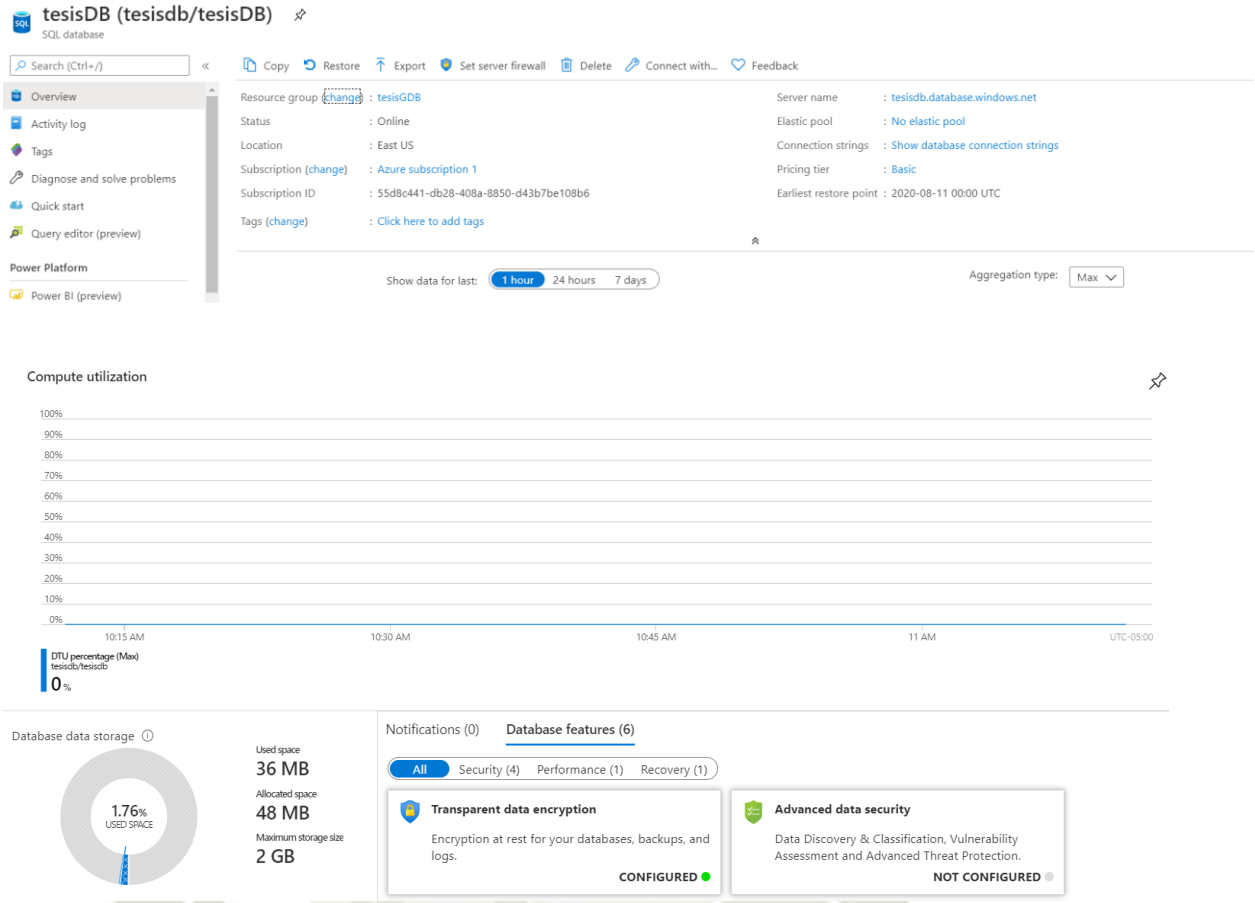

Figura 4.36: Despliegue Base de Datos en Azure

Fuente: Propia

4.5.3. Testeo de la Aplicación

A continuación se muestran los test unitarios, con lo cual se asegura que el código implementado están pasando los escenarios propuestos y con ello aseguramos la calidad sobre el desarrollo del presente software.

Los test unitarios, se realizan a nivel de código, los test para el desarrollo del proyecto, al estar bajo el framework MVC.NET, se realizara a nivel de los controladores, los cuales son el enlace entre la petición de la vista (browser) y el back end del software, es decir, desde la vista se generan las peticiones y el controlador según la petición realizada re direccionara al método que ejecuta la lógica del software.

Se realizara el test Unitario al método que devuelve el detalle del alumno, el mismo que nos muestra los datos grabados de la localización, en el presente

test se realizara que la vista que devuelve el controlador sea la adecuada, así como el modelo de datos de la misma, para asegurar la integración de data que se mostrara en la vista.

```
[TestMethod]
public void TestDetalleAlumno()
{
    var controller = new PaginasController();
    var result = controller._DetalleAlumno("2009601571") ;
    var detalleAlumno = (PartialViewResult)result;
    var model = detalleAlumno.Model;
    Assert.AreEqual("_DetalleAlumno", detalleAlumno.ViewName);
    Assert.IsNotNull(model);
}
```

Figura 4.37: Test Unitario Detalle Alumno

Fuente: Propia

También se realizara el test unitario al método de Historial Bitácora, con la cual se obtiene el historial de observaciones que se tuvo desde el área de responsabilidad social, en el test unitario se logra validar que el controlador devuelva la vista adecuada para la petición y el modelo de datos devuelto sea el que requiere la vista.

```
[TestMethod]
public void TestHistorialBitacora()
{
    var controller = new PaginasController();
    var result = controller._HistorialBitacora("2009601571");
    var bitacora = (PartialViewResult)result;
    var model = bitacora.Model;
    Assert.AreEqual("_HistorialBitacora", bitacora.ViewName);
    Assert.IsNotNull(model);
}
```

Figura 4.38: Test Unitario Historial Bitácora

Fuente: Propia

El presente test unitario, es para validar el método Búsqueda del alumno, el cual nos valida la existencia del alumno y sus datos básicos, como el código, nombres, carrera y facultad, se asegura que el controlador devuelva la vista

adecuada para el método, que el modelo de datos sea el requerido y que el dato devuelto sea el que se indicó en la búsqueda.

```
[TestMethod]
public void TestBusquedaAlumno()
{
    var controller = new PaginasController();
    var result = controller._BusquedaAlumno("2009601572");
    var busquedaAlumno = (PartialViewResult)result;
    var model = busquedaAlumno.Model;
    var alumnoEncontrado = (Alumnos)model;
    Assert.AreEqual("_BusquedaAlumno", busquedaAlumno.ViewName);
    Assert.IsNotNull(model);
    Assert.AreEqual("2009601572", alumnoEncontrado.codAlumno);
}
```

Figura 4.39: Test Unitario Búsqueda Alumno

Fuente: Propia

El siguiente test unitario, se encarga de garantizar el correcto funcionamiento del método Grabar Bitácora, con el cual asegura la integridad de datos y que estos se están guardando de manera adecuada.

```
[TestMethod]
public void TestGrabarBitacora()
{
    var controller = new PaginasController();
    var result = controller.SeguimientoAlumnos("GRABAR", "2009601572");
    var grabaBitacora = (ViewResult)result;
    var model = grabaBitacora.Model;
    var resultGrabaBitacora = (EntidadBitacora)model;
    Assert.AreEqual("SeguimientoAlumnos", grabaBitacora.ViewName);
    Assert.AreEqual("OK", resultGrabaBitacora.resultadoBitacora);
    Assert.IsNotNull(model);
}
```

Figura 4.40: Test Unitario Grabar Bitácora

Fuente: Propia

Con el siguiente test unitario, se validara el correcto funcionamiento del método de historial de fotos, con el cual se asegura el correcto funcionamiento del almacenamiento y visualización del repositorio fotográfico que se tenga.

```
[TestMethod]
public void TestHistorialFotos()
{
    var controller = new ImagenController();
    var result = controller.HistorialImagen("2009601572");
    var historialFotos = (ViewResult)result;
    var model = historialFotos.Model;
    var resultHistorialFotos = (EntidadImagenAlumno)model;
    Assert.AreEqual("HistorialImagen", historialFotos.ViewName);
    Assert.AreEqual("2009601572", resultHistorialFotos.codigoAlumno);
    Assert.IsNotNull(model);
}
```

Figura 4.41: Test Unitario Historial Fotos

Fuente: Propia

De acuerdo a los test antes mencionados, se presentan los resultados de los mismos, logrando así asegurar el correcto funcionamiento unitario de cada método y asegurando que el desarrollo se realizó de una manera adecuada.

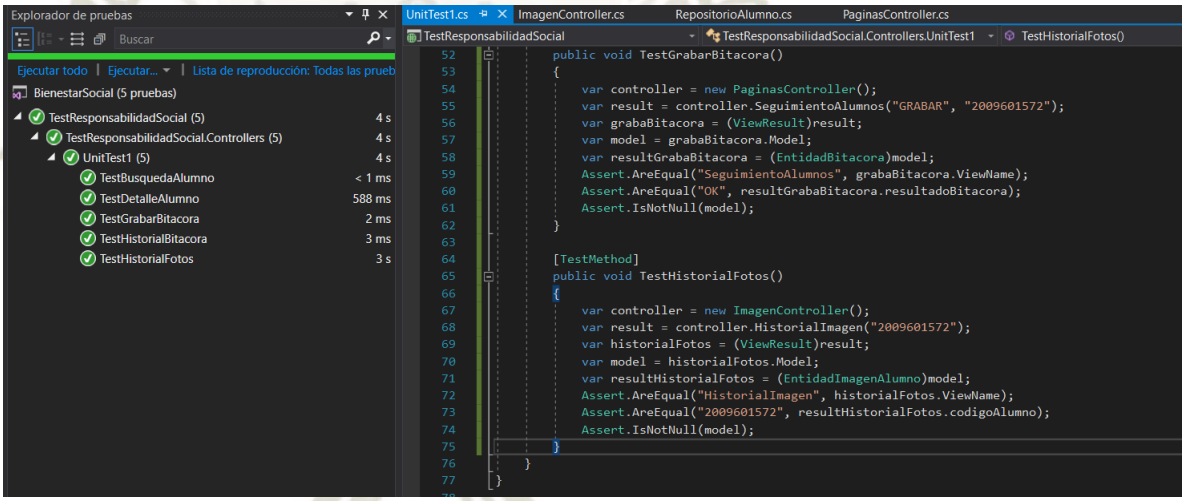

Figura 4.42: Resultados de Test Unitarios

Fuente: Propia

CAPITULO V

EJECUCION DE LA APLICACIÓN WEB

Este capítulo tiene como finalidad, presentar la ejecución de la aplicación web desarrollada frente a nuestros usuarios finales, con el objetivo de obtener y verificar los resultados provenientes de la aplicación.

5.1. Escenario Previo a la Puesta en Producción

La aplicación PlantacionArboles, tiene como objetivo principal, poder realizar la automatización del proceso de plantación de árboles, llevando un mejor control y tener almacenado todo en una aplicación web, para poder realizar el seguimiento adecuado a la actividad mencionada, con ello se obtendrán mayores beneficios, tanto para el alumno, como para el área de Responsabilidad social que es la encargada de validar el desarrollo adecuado de la actividad.

Actualmente no se tiene registro de los procesos, por lo que el proceso será aprobado por los alumnos de la carrera profesional de Ingeniería de Sistemas, validando así el correcto funcionamiento del sistema y mostrando los beneficios que se tendrá al utilizar el presente sistema.

Actualmente, el proceso es tedioso, debido a que se debe tener elementos físicos para las pruebas de la realización de la plantación, como fotos y estas deben ser archivadas físicamente, estando libres de cualquier error humano ocasionando perdida de información y poco control de la misma, siendo también que toma mucho tiempo en el almacenamiento y búsqueda de datos físicos cuando se requiera alguna validación.

5.2. Presentación de la Aplicación Web

Con fecha 01 de Agosto, se puso en producción el Sistema de Plantación de árboles, siendo publicada en la plataforma Azure, la cual será probada y validada por los alumnos de la facultad de ingeniería de sistemas, con el objetivo de conocer la forma de utilizar la aplicación y evidenciar las mejoras sobre el proceso que actualmente se viene brindando en el área, de igual manera el modulo para el área de responsabilidad social, se encuentra en la web publica, con lo cual podrá realizar el seguimiento y validación de los procesos que los alumnos registren en el sistema

5.3. Escenario Posterior a la Puesta en Producción

Una vez que sea usada la plataforma web, se podrá ver la utilidad de tener una web que sea accedida desde cualquier dispositivo que tenga acceso a internet, con ello se lograra tener un mejor registro de las evidencias que registren los alumnos, así como también la geolocalización de donde se realizó el plantado, debido a que la web accede a la localización del dispositivo en tiempo real y cuando completen el proceso de registro, se registrara la ubicación donde se realizara el proceso, haciendo el proceso más sencillo y fácil para el alumno; Por el lado del área de responsabilidad social, de igual manera podrán acceder a validar los datos ingresados por los alumnos, desde cualquier dispositivo móvil y con ello poder tener un registro más ordenado y más fácil de utilizar al momento de llevar un mejor control y validación del mismo.

CONCLUSIONES

PRIMERA:

Se desarrolló una solución web, de acuerdo a la necesidad que requería el área de Responsabilidad Social de la Universidad Católica Santa María, para así tener automatizado el proceso de plantación de árboles, así como modernizar dicho proceso llevándolo a un entorno web y tener un mejor control del mismo, la aplicación está disponible 27/7, siendo alojada en el servidor web Azure, lo cual asegura la estabilidad de la aplicación y el rendimiento de la misma.

SEGUNDA:

Debido a que la aplicación se encuentra en un entorno web y se encuentra alojada en Azure, lo que brinda la fluidez y disponibilidad de la empresa, debido a que cuenta con el soporte de Microsoft, esto con el fin de estar disponible a toda hora y pueda ser accedida desde cualquier dispositivo móvil, lo que ayudara en mejorar el proceso que realizara el alumno, ya que puede realizar el ingreso de los datos, desde el lugar donde realiza la plantación del árbol con su dispositivo móvil y subir las fotos que tome con el dispositivo móvil y ya que la aplicación accede a la geolocalización del dispositivo en tiempo real, registrada la localización correcta donde se realizó la plantación.

TERCERA:

Con la solución desarrollada, se logra mejorar y automatizar el proceso de plantación de árboles que actualmente se lleva a cabo de forma manual, ya que el proceso se realiza en un entorno digital, evita errores humanos, como perdida de información, lo cual beneficia al alumno para tener un correcto registro de la actividad y tener los beneficios que este proceso brinda.

CUARTA:

Se logró realizar un mejor control del proceso para el área de responsabilidad social, ya que actualmente el proceso, al ser datos físicos, suele ser tedioso en la búsqueda de la información y validación de la misma, con la presente aplicación, ese proceso se automatiza y se tiene más rápido la información que se necesite, logrando evaluar que el proceso sea correcto.

QUINTA:

El presente software, está desarrollado con una arquitectura orientada a servicios, lo que indica que gran parte del Core se encuentra en los servicios, lo que permite realizar una migración más sencilla a cualquier otra plataforma que se requiera, así como también integrar y acceder a sistemas externos de la universidad, para tener un cruce de información y evaluar más criterios que a futuro se tengan.

RECOMENDACIONES

PRIMERA

El sistema actual, funciona correctamente en una plataforma web, pero, al tener relación directa con la geolocalización, puede ser migrado a un entorno móvil, con lo cual se pueda aprovechar características propias de dichos entornos, como cámara, GPS, etc., lo que también ayudaría en el acceso para el alumno al tener una aplicación instalada en su propio dispositivo móvil y poder registrar datos sin necesidad de acceso a internet, para no perder la información y cuando el dispositivo se logre conectar a una red con internet, dichos datos sean enviados al servidor central y como el software desarrollado tiene gran parte de su Core en los servicios, dicha migración es más sencilla.

SEGUNDA

El sistema de Localización de Arboles (web y base de datos) de la presente tesis, se encuentra alojado actualmente en la plataforma Azure, que actualmente para el fin académico se tiene una licencia gratuita, la cual brinda estabilidad y fluidez para el soporte de la aplicación, sin embargo, para una mejor performance y mayores beneficios, se puede optar por obtener una licencia pagada, debido a que azure ofrece servicios los cuales solo cobra por lo que se usa, es recomendable validar dicha propuesta.

TERCERA

Para la geolocalización y presentación de mapas, se realizó integración con google maps, dicha integración para el fin académico, se tiene la versión de desarrollo, la cual limita algunas funcionalidades, se sugiere obtener la licencia para las APIS de google y con ello se lograra visualizar el mapa de forma correcta y podrá tener mayores herramientas con el tema de geolocalización.

REFERENCIAS BIBLIOGRÁFICAS

Alpañez Soler . (2012). *Escuela técnica superior de ingeniería de telecomunicación universidad politécnica de Cartagena*. Obtenido de http://docplayer.es/2825532-Escuela-tecnica-superior-de-ingenieria-de-telecomunicacion-universidad-politecnica-de-cartagena-proyecto-fin-de-carrera.html

Bonells. (2020). *Sistema de informacion geografica localización e identificacion de espacios en parques y jardines*. Obtenido de https://jardinessinfronteras.com/2020/05/19/sistemas-de-gestion-localizacion-e-identificacion-de-espacios-y-plantas-en-parques-y-jardines/

Carballo Yusneyi. (2007). *Programación Orientada a Objetos.* Venezuela: Universidad Central de Venezuela Escuela de Computación - Algoritmos y Programación.

Henríquez Fierro, E., & Zepeda González, M. I. (2003). Preparación de un proyecto de investigación. *Ciencia y enfermería, 9*(2), 23-28.

Hernández Orallo. (2002). *El Lenguaje Unificado de Modelado (UML).* España: Universidad Politécnica de Valencia.

IEDGE Business School. (2020). *IEDGE – Ciclo de vida en el Desarrollo de Software, primera parte*. Obtenido de https://www.iedge.eu/pablo-almunia-ciclo-de-vida-en-el-desarrollo-de-software-primera-parte

IPCC, Climate Change. (2013). *The Physical Science Basis - Summary for Policymakers, Observed Changes in the Climate System.* IPCC AR5 WG1, p. 15.

Jackson, R. y A. Jenkins. (2012). *Vital signs of the planet: global climate change and global warming: uncertainties.* California Institute of Technology: Earth Science Communications Team at NASA's Jet Propulsion Laboratory.

Jacobson, I., Booch, G. y Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.Addison-Wesley.*

James R, Ivar J, Grody B. (2002). *El Lenguaje Unificado de Modelado. Manual de Referencia.* México: 1era edición, Editorial Prentice Hall.

Otaya Burbano, L. A., Sánhez Zapata, R. D. J., Morales Soto, L., & Botero Fernández, V. (2006). Geographical information systems (gis), a great tool for urban sylviculture. *Revista Facultad Nacional de Agronomía Medellín*, *59*(1), 3201-3216.

Pressman, R. S. (2006). *ngeniería del Software: Un enfoque Práctico.* España: Mc-Graw-Hill.

Rivas Torres. (2000). *Arbolsig: sistema de informaci”n geográfica para árboles urbanos.* Santa Fé De Bogotá: [Tesis para optar al título de Especialista en Sistemas de Información Geográfica]. Obtenido de http://www.rivasdaniel.com/Articulos/proyecto_1027.pdf

Rivera Figueroa, D. (2019). *Tratamiento de aguas residuales y su influencia en el impacto ambiental del distrito de huaura año 2018.*

ANEXOS

ANEXO I

CODIGO FUENTE

INTRODUCCIÓN

A continuación, se presenta el código fuente del software realizado, mediante el cual se podrá revisar de forma rápida, como está estructurado técnicamente el software y así tener mayor conocimiento técnico de lo realizado.

En el módulo de registro de localización del alumno, primero se tiene el código donde se accede a la localización del dispositivo:

```
<script
src='https://maps.googleapis.com/maps/api/js?v=3.exp&key=AIzaSyCg-9bU-1n03SkxX-e1ML
XHZdDV8nVyIvU'></script>
<script type='text/javascript'>
    function init_map() {
        var lat;
        var lng;
        var alumno = $('#txtAlumno').val();
        var direccion = $('#txtDireccion').val();
        var distrito = $('#txtDistrito').val();
        var ciudad = $('#txtCiudad').val();
        //OBTENER LOCALIZACION ACTUAL DE SISTEMA Y MOSTRARLA
        if (navigator.geolocation) { //check geolocation available
            navigator.geolocation.getCurrentPosition(function (position) {
                lat =  position.coords.latitude;
                lng = position.coords.longitude;
                $("#latitudID").val(lat);
                $("#longitudID").val(lng);

                //MUESTRA MAPA
                var myOptions = { zoom: 17, center: new google.maps.LatLng(lat, lng),
mapTypeId: google.maps.MapTypeId.ROADMAP };
                map = new google.maps.Map(document.getElementById('gmap_canvas'),
myOptions);
                marker = new google.maps.Marker({ map: map, position: new
google.maps.LatLng(lat, lng) });
                infowindow = new google.maps.InfoWindow({ content: '<strong>' + alumno
+ '</strong><br>' + direccion + '<br>' + distrito + '<br>' + ciudad+'<br>' });
                google.maps.event.addListener(marker, 'click', function () {
infowindow.open(map, marker); });
                infowindow.open(map, marker);
                //FIN
            });
        } else {
            console.log("¡El navegador no admite la geolocalización!");
        }
        //FIN

    }
    google.maps.event.addDomListener(window, 'load', init_map);
</script>
```

Posteriormente, visualizamos la forma en que se están grabando los datos del alumno, tanto la localización como las evidencias (fotos) cargadas, el proceso donde se realiza el guardado de datos, se hace en el controlador, el cual realiza las conexiones con los servicios.

```
[HttpPost]
        public ActionResult Inicio(HttpPostedFileBase file, FormCollection form)
        {
            string mensajeRespuesta = "";
            string codigoAlumno = Session["codigoAlumno"].ToString();
            string direccion = form["direccion"];
            string distrito = form["distrito"];
            string ciudad = form["ciudad"];
            string latitud = form["latitud"];
            string longitud = form["longitud"];

            /* SECCION DONDE GRABA DATOS DE ALUMNO Y ARBOL*/
            EntidadAlumno alumnoEncontrado = new EntidadAlumno();
            alumnoEncontrado = (EntidadAlumno)Session["AlumnoEncontrado"];
            EntidadAlumnoArbol alumnoGrabar = new EntidadAlumnoArbol()
            {
                codAlumno = alumnoEncontrado.codAlumno,
                nombreAlumno = alumnoEncontrado.nombreAlumno,
                facultadAlumno = alumnoEncontrado.facultadAlumno,
                carreraAlumno = alumnoEncontrado.carreraAlumno,
                direccionAlumno = direccion,
                ciudadAlumno = ciudad,
                distritoAlumno = distrito,
                latitud = latitud,
                longitud = longitud
            };
            string resultado = AlumnoArbolService.grabaAlumnoArbol(alumnoGrabar);
            /* FIN */

            /* GRABA IMAGEN DEL ARBOL */
            if (resultado == "OK")
            {
                if (file != null)
                {
                    using (MemoryStream ms = new MemoryStream())
                    {
                        file.InputStream.CopyTo(ms);
                        byte[] array = ms.GetBuffer();
                        string resultImagen =
ImagenService.grabaImagenAlumnoArbol(codigoAlumno, array);
                        if (resultImagen == "OK")
                        {
                            mensajeRespuesta = "SE INGRESO REGISTRO CORRECTAMENTE";
                        }
                        else
                        {
                            mensajeRespuesta = "SE INGRESO REGISTRO CORRECTAMENTE. \n
OCURRIO UN ERROR AL INGRESAR LA IMAGEN \n SE SUGIERE ELEGIR UNA IMAGEN DE MENOR TAMAÑO";
                        }
                    }

                }
            }
            else
```

```
            {
                mensajeRespuesta = "OCURRIO UN ERROR AL INGRESAR LA DIRECCION DEL ARBOL
PLANTADO";
            }
            /* FIN */
            Session["MensajeRespuestaAlumnoArbol"] = mensajeRespuesta;
            return RedirectToAction("Respuesta");
        }
```

A continuación, se muestra la conexión a los servicios y la obtención de la respuesta serializando el json que envían los servicios.

```
public static string grabaAlumnoArbol(EntidadAlumnoArbol alumnoArbol)
        {
            string resultado = "";
            try
            {
                using (var httpCliente = new HttpClient())
                {
                    httpCliente.BaseAddress = new Uri(urlApiAlumnoArbol);
                    var response =
httpCliente.PostAsJsonAsync<EntidadAlumnoArbol>("AlumnoArbol",alumnoArbol);
                    response.Wait();

                    var result = response.Result;
                    resultado = result.StatusCode.ToString();
                }
            }

            catch
            {

            }

            return resultado;
        }
```

En el módulo de Responsabilidad Social, se muestra el código de como obtiene los datos del alumno, obteniendo mediante los servicios el historial de fotos y sus datos de localización.

```
$(document).ready(function () {
        $(".verAlumno").click(function () {
            var codigoAlumno = $(this).parents("tr").find("td").eq(0).html();
            var alumno = $(this).parents("tr").find("td").eq(1).html();
            var facultad = $(this).parents("tr").find("td").eq(2).html();
            var carrera = $(this).parents("tr").find("td").eq(3).html();
            $.ajax({
                url: '@Url.Action("_DetalleAlumno", "Paginas")',
                data: { codAlumno: codigoAlumno },
                type: "post",
                cache: false,
                success: function (retorno) {
                    $("#secSeguimientoAlumno").html(retorno);
                    $('#secVerAlumnos').hide("1s");
                    $('#secSeguimientoAlumno').show("1s");
                },
                error: function () {
                    alert("Se ha producido un error");
                }
            });
        });
    });



[HttpPost]
        public ActionResult _DetalleAlumno(string codAlumno)
        {
            EntidadAlumnoArbol alumnoArbol = new EntidadAlumnoArbol();
            Alumnos alumnoSinPlantar = new Alumnos();
            alumnoArbol = AlumnoArbolService.buscarAlumno(codAlumno);
            if (alumnoArbol.nombreAlumno == null)
            {
                alumnoSinPlantar = AlumnosService.buscarAlumno(codAlumno);
                alumnoArbol.codAlumno = alumnoSinPlantar.codAlumno;
                alumnoArbol.nombreAlumno = alumnoSinPlantar.nombreAlumno;
                alumnoArbol.facultadAlumno = alumnoSinPlantar.facultadAlumno;
                alumnoArbol.carreraAlumno = alumnoSinPlantar.carreraAlumno;
                ViewBag.ErrAlumnoArbol = "EL ALUMNO NO REGISTRO DATOS DE SU ARBOL";
                ViewBag.FlagErr = "TRUE";
                return PartialView(alumnoArbol);

            }
            else
            {
                return PartialView(alumnoArbol);
            }
            return PartialView(alumnoArbol);
        }



[HttpPost]
        public ActionResult _BusquedaAlumno(string codAlumno)
        {
            Alumnos objAlumno = new Alumnos();
```

```
            objAlumno = AlumnosService.buscarAlumno(codAlumno);
            return PartialView(objAlumno);
        }

[HttpPost]
        public ActionResult _HistorialFotos(string codAlumno)
        {
            List<CantidadImagenes> idImagenes = new List<CantidadImagenes>();
            Session["codAlumnoHistorialFotos"] = codAlumno;
            idImagenes = ImagenService.getCantidadImagenes(codAlumno);

            return PartialView(idImagenes);
        }

[HttpPost]
        public ActionResult _VistaHistorialFotos(int idImagen)
        {
            CantidadImagenes Imagen = new CantidadImagenes();
            string codAlumno = Session["codAlumnoHistorialFotos"] .ToString();
            Imagen = ImagenService.getFechaImagenById(codAlumno, idImagen);
            Imagen.idImagen = idImagen;

            return PartialView(Imagen);
        }
```

La conexión a los servicios es la siguiente, donde se observa la socialización de la respuesta de los servicios en formato Json al modelo respectivo.

```
static HttpClient client = new HttpClient();
        static string urlApiAlumnoArbol =
ConfigurationManager.AppSettings["urlApiAlumnoArbol"].ToString();
        public static EntidadAlumnoArbol buscarAlumno(string codigoAlumno)
        {
            EntidadAlumnoArbol alumnoEncontrado = new EntidadAlumnoArbol();

            try
            {

                string sUrlRequest = urlApiAlumnoArbol + codigoAlumno;
                var json = new WebClient().DownloadString(sUrlRequest);
                JavaScriptSerializer ser = new JavaScriptSerializer();
                alumnoEncontrado = ser.Deserialize<EntidadAlumnoArbol>(json);

            }
            catch
            {
            }

            return alumnoEncontrado;
        }
```

A continuación se muestra el desarrollo y codificación de los servicios creados, los cuales tienen la interacción directa con BD.

El servicio de Búsqueda Alumnos, es el siguiente:

```
public Alumno GetAlumno(string codigoAlumno)
        {
            Alumno alumnoEncontrado = new Alumno();

            alumnoEncontrado = RepositorioAlumno.getAlumno(codigoAlumno);
            return alumnoEncontrado;
        }

public static Alumno getAlumno(string codigoAlumno)
        {
            Alumno alumno = new Alumno();
            alumno = AlumnoBD.getAlumno(codigoAlumno);

            return alumno;
        }

public static Alumno getAlumno(string codigoAlumno)
        {
            Alumno alumno = new Alumno();
            string cadenaConexion = CadenaConexion.getCadenaConexion();
            try
            {
                SqlConnection con = new SqlConnection(cadenaConexion);
                con.Open();
                SqlCommand comando = new SqlCommand();
                comando.Connection = con;
                comando.CommandType = CommandType.StoredProcedure;
                comando.CommandText = "sp_getAlumno";
                comando.Parameters.Add("@codigoAlumno", SqlDbType.VarChar).Value =
codigoAlumno;
                SqlDataReader reader = comando.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        alumno.codAlumno = reader[1].ToString();
                        alumno.nombreAlumno = reader[2].ToString() + "/" +
reader[3].ToString();
                        alumno.facultadAlumno = reader[4].ToString();
                        alumno.carreraAlumno = reader[5].ToString();

                    }
                }
                reader.Close();

                con.Close();

            }
            catch
            {

            }
            return alumno;
        }
```

El servicio que graba los datos de localización del alumno, es el siguiente:

```
public string PostAlumnoArbol([FromBody]AlumnoArbol alumno)
        {
            string respuesta = RepositorioAlumnoArbol.grabaAlumnoArbol(alumno);
            return respuesta;
        }

public static string grabaAlumnoArbol(AlumnoArbol alumno)
        {
            string resultado = "";
            resultado = AlumnoArbolBD.grabaAlumnoArbol(alumno);
            return resultado;
        }

public static string grabaAlumnoArbol(AlumnoArbol alumnoArbol)
        {
            string cadenaConexion = CadenaConexion.getCadenaConexion();
            string nombreAlumno;
            string apellidoAlumno;
            string[] datosAlumno;
            datosAlumno = alumnoArbol.nombreAlumno.Split('/');
            nombreAlumno = datosAlumno[0];
            apellidoAlumno = datosAlumno[1];
            string resultado = "";
            try
            {
                SqlConnection con = new SqlConnection(cadenaConexion);
                con.Open();
                SqlCommand comando = new SqlCommand();
                comando.Connection = con;
                comando.CommandType = CommandType.StoredProcedure;
                comando.CommandText = "sp_InsertaAlumnoArbol";
                comando.Parameters.Add("@codAlumno", SqlDbType.VarChar).Value =
alumnoArbol.codAlumno;
                comando.Parameters.Add("@nombreAlumno", SqlDbType.VarChar).Value =
nombreAlumno;
                comando.Parameters.Add("@apellidoAlumno", SqlDbType.VarChar).Value =
apellidoAlumno;
                comando.Parameters.Add("@facultad", SqlDbType.VarChar).Value =
alumnoArbol.facultadAlumno;
                comando.Parameters.Add("@carrera", SqlDbType.VarChar).Value =
alumnoArbol.carreraAlumno;
                comando.Parameters.Add("@direccion", SqlDbType.VarChar).Value =
alumnoArbol.direccionAlumno;
                comando.Parameters.Add("@distrito", SqlDbType.VarChar).Value =
alumnoArbol.distritoAlumno;
                comando.Parameters.Add("@ciudad", SqlDbType.VarChar).Value =
alumnoArbol.ciudadAlumno;
                comando.Parameters.Add("@latitud", SqlDbType.VarChar).Value =
alumnoArbol.latitud;
                comando.Parameters.Add("@longitud", SqlDbType.VarChar).Value =
alumnoArbol.longitud;
                /* PARAMETRO DE SALIDA/OUTPUT */
                SqlParameter result = new SqlParameter("@Resultado", SqlDbType.VarChar,
2);
                result.Direction = ParameterDirection.Output;
                comando.Parameters.Add(result);

                int rowsAffected = comando.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
```

```
                    resultado =
Convert.ToString(comando.Parameters["@Resultado"].Value);
                }
                else
                    resultado = "NO";

                con.Close();

            }
            catch
            {

            }
            return resultado;
        }
```

El Servicio de Historial de fotos, se muestra a continuación:

```
public EntidadImagenAlumno GetImagen(string codigoAlumno)
        {
            EntidadImagenAlumno imagen = new EntidadImagenAlumno();
            imagen = RepositorioImagenes.getImagenAlumno(codigoAlumno);
            return imagen;
        }

public static EntidadImagenAlumno getImagenAlumno(string codigoAlumno)
        {
            EntidadImagenAlumno imagenAlumno = new EntidadImagenAlumno();
            imagenAlumno = ImagenesBD.getImagenAlumno(codigoAlumno);
            return imagenAlumno;
        }

public static EntidadImagenAlumno getImagenAlumno(string codigoAlumno)
        {
            EntidadImagenAlumno imagenAlumno = new EntidadImagenAlumno();
            string cadenaConexion = CadenaConexion.getCadenaConexion();
            try
            {
                SqlConnection con = new SqlConnection(cadenaConexion);
                con.Open();
                SqlCommand comando = new SqlCommand();
                comando.Connection = con;
                comando.CommandType = CommandType.StoredProcedure;
                comando.CommandText = "sp_getImagenAlumnoArbol";
                comando.Parameters.Add("@codAlumno", SqlDbType.VarChar).Value =
codigoAlumno;
                SqlDataReader reader = comando.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        imagenAlumno.codigoAlumno = reader[1].ToString();
                        imagenAlumno.imagen = (Byte[])reader[2];
                    }
                }
                reader.Close();

                con.Close();
            }
            catch
            {

            }
            return imagenAlumno;
        }
```

El Servicio para grabar bitácora, es el siguiente:

```
public string PostBitacora([FromBody]EntidadBitacora bitacora)
        {
            string respuesta = RepositorioBitacora.grabaBitacora(bitacora);

            return respuesta;
        }

public static string grabaBitacora(EntidadBitacora bitacora)
        {
            string resultado = "";
            resultado = BitacoraBD.grabaBitacora(bitacora);
            return resultado;
        }

public static string grabaBitacora(EntidadBitacora bitacora)
        {
            string cadenaConexion = CadenaConexion.getCadenaConexion();
            string resultado = "";
            try
            {
                SqlConnection con = new SqlConnection(cadenaConexion);
                con.Open();
                SqlCommand comando = new SqlCommand();
                comando.Connection = con;
                comando.CommandType = CommandType.StoredProcedure;
                comando.CommandText = "sp_InsertaBitacora";
                comando.Parameters.Add("@codAlumno", SqlDbType.VarChar).Value =
bitacora.codigoAlumno;
                comando.Parameters.Add("@fecha", SqlDbType.DateTime).Value =
Convert.ToDateTime(bitacora.fecha);
                comando.Parameters.Add("@descripcion", SqlDbType.VarChar).Value =
bitacora.descripcion;
                comando.Parameters.Add("@usuario", SqlDbType.VarChar).Value =
bitacora.usuario;
                /* PARAMETRO DE SALIDA/OUTPUT */
                SqlParameter result = new SqlParameter("@Resultado", SqlDbType.VarChar,
2);
                result.Direction = ParameterDirection.Output;
                comando.Parameters.Add(result);

                int rowsAffected = comando.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    resultado =
Convert.ToString(comando.Parameters["@Resultado"].Value);
                }
                else
                    resultado = "NO";

                con.Close();

            }
            catch
            {

            }
            return resultado;
        }
```

Finalmente el código, para envío de mensajería, se detalla a continuación:

```
public string GetSendEmailBitacora(string mensaje, string codigo_alumno)
        {
            List<string> lstCorreos = new List<string>();
            string mensajeError = string.Empty;
            lstCorreos = RepositorioCorreoAlumno.getCorreoAlumno(codigo_alumno);

            string correo1 = lstCorreos[0];
            string correo2 = lstCorreos[1];
            if (correo2 == null || correo2 == "")
            {
                correo2 = correo1;
            }
            string EnviaMail =
ConfigurationManager.AppSettings["MailEnvia"].ToString();
            string passwordEnviaMail =
ConfigurationManager.AppSettings["ClaveMailEnvia"].ToString();
            string mensajePara = correo1;
            string mensajeCopia = correo2;
            string asunto = ConfigurationManager.AppSettings["AsuntoMail"].ToString();
            string hostMail = ConfigurationManager.AppSettings["HostMail"].ToString();

            /* CONTENIDO DEL MAIL */
            System.Net.Mail.MailMessage email = new System.Net.Mail.MailMessage();
            email.To.Add(mensajePara); //Destino del mensaje, correo de quien recibe
            email.Subject = asunto; // Asunto del mensaje
            email.SubjectEncoding = System.Text.Encoding.UTF8; //Codificacion UTF para
servidor
            email.Bcc.Add(mensajeCopia); //Correo que se encuentra en copia
            email.Body = mensaje; //Contenido del mensaje
            email.BodyEncoding = System.Text.Encoding.UTF8; //Codificacion UTF del
mensaje para el servidor
            email.From = new System.Net.Mail.MailAddress(EnviaMail); //Correo que envia
el mail

            /* CLIENTE PARA ENVIAR MAIL */
            System.Net.Mail.SmtpClient cliente = new System.Net.Mail.SmtpClient();
            cliente.Credentials = new System.Net.NetworkCredential(EnviaMail,
passwordEnviaMail); //Credenciales del correo que envia mensaje
            cliente.Port = 587; //puerto para GMAIl
            cliente.EnableSsl = true; //SEGURIDAD
            cliente.Host = hostMail;//"smtp.gmail.com";

            /* ENVIAR MENSAJE */
            try
            {
                cliente.Send(email);
                mensajeError = "Mensaje Enviado con Exito";
            }
            catch (Exception ex)
            {
                mensajeError = "Error al enviar mensaje";
            }

            return mensajeError;
        }
```