UNIVERSIDAD CATÓLICA DE SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"SISTEMA MÓVIL DE DETECCIÓN DE CAÍDAS PARA ADULTOS MAYORES USANDO BEACONS"

Borrador de tesis presentado por el Bachiller:

Gustavo Hernando Puma Tejada

Para optar por el Título Profesional:

INGENIERO DE SISTEMAS

Arequipa – Perú

2016

Este trabajo va dedicado a mis padres,

por su apoyo incesante.

ÍNDICE

Índice de Ilustraciones

Todas las Ilustraciones referenciadas en este trabajo son de Elaboración Propia.

Índice de Tablas

Todas las Tablas referenciadas en este trabajo son de Elaboración Propia.

RESUMEN

Los adultos mayores actualmente representan el diez por ciento de la población en el Perú, cifra que está proyectada a incrementar. Según la Organización Mundial de la Salud (OMS), las caídas son la segunda causa de muerte a nivel mundial, siendo los adultos mayores los más propensos a caídas, donde la falta de atención inmediata puede resultar en la muerte, por eso es crítico poder realizar la detección oportuna de estos incidentes. Existe una vasta literatura sobre la detección de caídas y a su vez, numerosos dispositivos dedicados para detectarlas. Sin embargo, dichos dispositivos suelen ser rechazados por los usuarios por sus interfaces no amigables, su alto costo y su baja disponibilidad (e.g. solo disponibles en casas de retiro). El presente trabajo propone un sistema móvil de bajo costo, diseñado principalmente para smartphones, el cual junto con unos dispositivos inalámbricos llamados Beacons nos permite realizar la detección de caídas. Esto es posible a través de una serie de algoritmos diseñados exclusivamente para el sistema propuesto. El sistema tiene dos modos: Paciente y Cuidador. El Paciente es la persona en riesgo de sufrir una caída y El Cuidador, la persona que será notificada sobre cualquier emergencia que sea detectada por el sistema. Se desarrolló una app prototipo para Android y iOS para probar su precisión. Se encontró que el sistema es significativamente sensible al posicionamiento del smartphone; cuando el smartphone está en el bolsillo del usuario el sistema tiene una precisión casi aleatoria (50-60%), sin embargo, cuando el smartphone se encuentra a la altura del pecho de la persona alcanza una precisión de 80%. Se concluyó que la detección de caídas usando Beacons es un área de investigación prometedora y viable, que se beneficiará de las mejoras en la precisión de la información reportada por dicho dispositivos y de pruebas más extensivas, pero que también es crucial mejorar los algoritmos para que se apoyen en los otros sensores disponibles en los smartphones: e.g. acelerómetro y giroscopio.

ABSTRACT

Elderly people currently make up the 10% of the Peruvian population, number which is due to increase. According to the World Health Organization (WHO), falls are the second cause of death worldwide, elderly people being the most prone to them, where the lack of immediate medical attention can result in death, thus, it's critical to be able to detect these incidents in a timely fashion. There is a vast literature about fall detection and at the same time, numerous dedicated fall detection devices. Nonetheless, such devices are usually rejected by the users due to their lack of user-friendliness, their high cost and low availability (e.g. only available for retirement homes). The present work proposes a low-cost mobile systems, designed primarily for smartphones, which along with wireless devices known as Beacons allows as to perform fall detection. This is possible through a series of algorithms designed exclusively for the proposed system. The system contains two modes: Patient and Carer. The Patient is the person at risk of suffering a fall and The Carer, the person who will be notified about any emergency that is detected by the system. A prototype app for Android and iOS was developed to test the accuracy of the system. It was found that the system is significantly sensitive to the positioning of the smartphone; when the smartphone is in the user's pocket it has an accuracy near random (50-60%), however, when the smartphone is at chest height it reaches an accuracy of 80%. It was concluded that fall detection using Beacons is a promising and viable area of research, that will benefit from the improvements in the precision of the information reported by said devices and from more extensive tests, but that is also crucial to improve algorithms so that they benefit from other sensors available in smartphones: e.g. accelerometer and gyroscope.

INTRODUCCIÓN

En el último par de años ha resurgido el interés en el uso de las más recientes tecnologías de nivel consumidor para tratar con la salud de las personas, mayormente en países como EEUU y Australia, ya sea en la academia o en el sector privado. Dicho resurgimiento puede atribuirse al boom de los smartphones en la década pasada y al exponencial incremento en las capacidades del hardware con el que son equipados. Desafortunadamente, el resurgimiento mencionado no se ha presentado en el Perú todavía, las razones por qué podrían ser especuladas y debatidas (lo que a su vez podría ser materia de investigación para otro trabajo). Sin embargo, el hecho es de que ahora más que nunca la población peruana ha hecho evidente su exigencia de reformas en el sector salud y, aunque el objetivo de este trabajo no es plantear una solución a tal problema, lo que se propone podría aliviar de cierta manera la carga de dicho sector.

El presente trabajo es, en primer lugar, una respuesta a un problema que ha permeado a todas las sociedades modernas: las caídas de los adultos mayores (cuya significancia veremos más adelante), en segundo lugar, un enfoque innovador para mitigar el problema anterior haciendo uso de una tecnología relativamente nueva: Beacons, y, en tercer lugar, un intento por despertar el interés en la investigación en el área de la "Salud Móvil" (mHealth).

En el capítulo 1 se establece el problema de detección de caídas como la motivación que guía al resto del trabajo y se plantea la hipótesis o solución que será desarrollada a lo largo de la investigación, junto con sus alcances y limitaciones. Adicionalmente, se detalla la metodología y procedimientos de investigación a ser usados.

En el capítulo 2 se sientan las bases teóricas: se examina la literatura existente sobre la detección de caídas, los enfoques usados alrededor del área de Salud Móvil (mHealth), junto con los vacíos de conocimiento actuales en dicha área. A su vez, se presenta la terminología y conceptos predominantes a lo largo del documento.

En el capítulo 3 se realiza el diseño del sistema como solución propuesta. Se utilizan métodos estándares de la industria como diagramas UML, mockups y diagramas de Entidad-Relación. Así mismo, se especifica el diseño de los algoritmos que conforman

el núcleo del sistema y se presentan las consideraciones filosóficas que entraron en consideración en esta etapa, en especial aquellas pertinentes al diseño de la interfaz.

En el capítulo 4 se presentan los detalles pertinentes a la implementación del sistema como un prototipo de software, más específicamente, una aplicación móvil, con la finalidad de poder reproducir los resultados posteriormente si se requiriera.

En el capítulo 5 y 6 se especifica el procedimiento de pruebas usado y se discuten los resultados encontrados, respectivamente. En el capítulo 7 se documentan las conclusiones del trabajo de investigación, principalmente de los resultados de las pruebas. En el capítulo 8 se realizan las recomendaciones y direcciones futuras a tomarse para futuros emprendimientos en el área.

CAPÍTULO 1 - PLANTEAMIENTO TEÓRICO

1.1. Planteamiento del Problema

Según el último censo poblacional, los adultos mayores (de 65 años de edad a más) representan el 10% de la población en el Perú, y esta cifra está proyectada a incrementarse en un 3% adicional para el año 2025 (Inei, 2015). Las caídas son la segunda causa mundial de muerte accidental y son precisamente los adultos mayores los que sufren más caídas mortales (OMS, 2012). Estas caídas, aunque no son inicialmente mortales, requieren atención médica inmediata para poder prevenir una fatalidad (OMS, 2012). Los sistemas de salud en múltiples países desarrollados están experimentando una crisis de costos actualmente, siendo este problema el más importante que han encontrado hasta ahora (Korhonen & Bardram, 2004). Esta crisis de costos es mucho más evidente en un país en desarrollo como el nuestro, donde los hospitales no se dan abasto ni con los profesionales médicos (Perú 21, 2014), ni con el equipo requerido (Bardales, 2015).

Dado que el 32% de los adultos mayores (AM) peruanos viven solos (Olivera & Clausen, 2014) están en riesgo constante de sufrir un accidente y no disponer de nadie que recurra a su auxilio. Aún en el caso de aquellos AM que viven con otras personas, estos se encuentran solos por la mayor parte del día debido a que, usualmente, los familiares se encuentran trabajando y/o estudiando (Olivera & Clausen, 2014). Tener una persona dedicada al cuidado del AM, así como invertir en una casa de retiro (ya sea financiado por el mismo AM o por la familia) es algo factible sólo para aquellos en los sectores socioeconómicos más altos, lo cual deja en el aire a la mayoría de la población en nuestro país.

Existe la necesidad de poder responder a este tipo de emergencias de una manera rápida, que no requiera la constante supervisión del AM y/o el desembolso de considerables sumas de dinero para el cuidado del AM, haciendo uso de las tecnologías existentes.

1.2. Objetivos de la Investigación

1.2.1. General

Diseñar e implementar un sistema ubicuo de detección de caídas para adultos mayores que permita la detección y respuesta rápida, sin requerir la constante supervisión del AM por un individuo y/o el desembolso de considerables sumas de dinero para el cuidado del AM, haciendo uso de las tecnologías disponibles a nivel consumidor.

1.2.2. Específicos

- Definir la arquitectura del sistema y de la información, casos de uso, mockups, wireframes y otras secciones relevantes.
- Diseñar los algoritmos que empoderan las funciones críticas del sistema.
- Implementar un prototipo del sistema empleando buenas prácticas de desarrollo de software.
- Probar la precisión de detección de emergencias del sistema diseñado, distinguiendo las diferentes situaciones contempladas por cada algoritmo y considerando la posición del dispositivo final relativa al cuerpo.

1.3. Preguntas de Investigación

- ¿De qué manera debería ser diseñado un sistema ubicuo de detección de caídas cuyo usuario final es el adulto mayor? ¿Qué consideraciones de diseño de interfaces humano-computador son relevantes cuando tratamos de este grupo demográfico?
- ¿Cuáles son las mejores prácticas qué deben emplearse al desarrollar software para dispositivos móviles? ¿Cómo podemos hacer un uso eficiente de los recursos de los dispositivos finales?
- ¿Cómo podemos medir la efectividad del sistema diseñado?

1.4. Área y línea de investigación

Área: Computación Móvil y Ubicua

Línea de investigación: mHealth (Mobile Health: Salud Móvil)

1.5. Tipo de la Investigación

- **Tecnológica**

1.6. Nivel de Investigación

- **Experimental o Evaluatoria**

1.7. Técnica e instrumentos

- Para la obtención de la información se realizará una revisión literaria.
- Para implementar el diseño especificado se recurrirá al desarrollo de un prototipo.
- Para probar la funcionalidad del prototipo se realizarán pruebas controladas sobre cada caso de uso presentado.

1.8. Procedimientos de recolección

El procedimiento de recolección de datos se basó en una revisión literaria, extrapolando conocimiento de las fuentes bibliográficas especificadas en la última sección. Se utilizaron las bases de datos académicas más reconocidas mundialmente para buscar trabajos académicos relevantes, los cuales sirvieron a su vez como punto de enlace con otras investigaciones. En el caso de los papers, solo se tomó en cuenta a aquellos que hayan pasado por el proceso de revisión por pares.

1.9. Plan de análisis de datos

Sobre los materiales discutidos se aplicará el método de análisis, que consiste en la descomposición funcional del tema en partes separadas para su estudio individual, y el método de síntesis, que consistirá en una reunión racional de los elementos dispersos para estudiarlos en su totalidad. Al final de la investigación se llevarán a cabo dos fases experimentales: la primera consistirá en desarrollar el prototipo según el diseño especificado. La segunda consiste en la prueba del prototipo, los resultados de esta prueba serán utilizados para derivar las conclusiones pertinentes.

1.10. Plan de Trabajo

Investigación Documental	**1 semana**
Diseño Sistema	**2 semanas**
Implementación Prototipo	**3 semanas**
Prueba Prototipo	**1 semanas**

1.11. Solución Propuesta

1.11.1. Descripción y justificación de la solución

La solución propuesta es un sistema móvil de detección de caídas diseñado para smartphones. Dicho sistema hará uso de dispositivos llamados Beacons, los cuales nos permiten obtener información contextual sobre el ambiente en el que se encuentran, y en conjunto con un Smartphone, nos permiten realizar una serie de cálculos que servirán de base para nuestros algoritmos de detección. El enfoque de esta investigación se encuentra en los datos de proximidad y de ubicación que nos proveen estos beacons para poder monitorear las actividades cotidianas del AM. Adicionalmente, para incrementar la precisión de detección de nuestros algoritmos, estos trabajarán en conjunto con los datos del acelerómetro del Smartphone.

El sistema está dividido en dos módulos: el módulo Paciente y el módulo Cuidador. El módulo del paciente es el sistema centralizado (que corre en el smartphone del paciente) que recibe y procesa los datos transmitidos por los beacons. Este módulo está automatizado en casi su totalidad, lo que quiere decir que todos los cálculos se realizan en segundo plano sin requerir la intervención del paciente, resultando en un sistema plenamente ubicuo, siendo la única excepción cuando una emergencia es detectada. Los algoritmos propuestos, a pesar de hacer uso extensivo de las funciones sensores y de red del dispositivo en mención, en esencia realizarán cálculos básicos. El sistema está diseñado para aprovechar las características actuales de los smartphones, sean de gama alta o baja.

El sistema está basado en la premisa de que el Paciente esté a cargo de una persona (normalmente un familiar) que será designada como Cuidador. El módulo cuidador es el que hace las funciones de administrador y de punto de contacto cuando existe una emergencia. Cualquier cambio en la configuración del módulo del paciente es manejado por el Cuidador a través de una contraseña.

El cuidador puede supervisar remotamente al paciente o, si lo desea, solo ser notificado cuando se detecta una posible emergencia. Una posible emergencia es un evento generado por los algoritmos que corren en segundo plano en el dispositivo del paciente cuando se ha detectado una anomalía, como una posible caída, pérdida de conexión con el paciente, batería baja, entre otros. Estos eventos son presentados como notificaciones en el dispositivo del cuidador, solicitando acción por su parte, e.g. llamar al paciente, llamar a una ambulancia, etc. Adicionalmente, el cuidador puede ver el estado del paciente a cualquier hora. Por el lado del paciente, éste puede confirmar si es que la emergencia detectada es legítima o si simplemente fue un falso positivo.

Este sistema está enfocado en proveer autonomía para los AM, utilizando tecnologías ya establecidas, a diferencia de la mayor parte de los productos disponibles en la industria, los cuales están basados en dispositivos especializados con costos inasequibles para el peruano promedio. También busca brindar la detección de diferentes tipos de emergencias, todo consolidado en un solo sistema. La visión final es crear un ambiente que sirva como base para que cualquier persona que necesite encargarse de alguien pueda hacerlo usando tecnologías a nivel del consumidor a través de este sistema, dejando la mayoría de sus preocupaciones de lado e interviniendo solo cuando es necesario, garantizando la autonomía de ambas partes.

Aunque las estadísticas disponibles sobre el uso de smartphones en el Perú son escasas, fue determinado que el 23% de la población cuenta con un smartphone (Ipsos, 2015), (Futuro Labs, 2015). Cifra que está proyectada a crecer en un 22% adicional dentro de cuatro años (Ericsson, 2016). Es por esta razón que el autor cree que la plataforma móvil es la más apta para un sistema de este tipo, considerando además los precios cada vez más bajos de estos dispositivos y el hecho de que el sistema propuesto será diseñado para funcionar sin problemas en los smartphones modernos de gama baja.

1.11.2. Alcances y limitaciones

Alcances

- El sistema será desarrollado utilizando el framework Xamarin, el cual permite utilizar una sola base de código compartida entre las plataformas móviles estándar: Android y iOS.
- El enfoque del sistema será la detección de caídas en adultos mayores utilizando beacons, con la posibilidad de utilizar la detección por acelerómetro si resulta ser lo suficientemente precisa.

Limitaciones

- El sistema será probado en dispositivos Android solamente, debido al requerimiento de poseer una Mac para poder desplegar proyectos de iOS.
- El sistema requiere dispositivos móviles con soporte de Bluetooth Low Energy (BLE 4.0) para la comunicación con los beacons.
- El sistema diseñado, al estar basado en el uso de BLE, también está limitado por, valga la redundancia, las mismas limitaciones físicas y tecnológicas que le son inherentes a dicho protocolo, estas son, principalmente, rango de comunicación e intervalo de transmisión de datos, entre otras.
- Los dispositivos móviles requieren una conexión a internet constante.

CAPÍTULO 2 - FUNDAMENTOS TEÓRICOS

2.1. Estado del arte

Desde su incepción, los dispositivos móviles han sido tomados en cuenta en la comunidad académica por su potencial tremendo para transformar la industria de la salud. Varios estudios se han llevado a cabo sobre el uso de celulares como soporte para el sistema médico, sin embargo, la mayoría se ha concentrado en su uso en la recolección de información médica para su futura investigación (Blaya, Fraser, & Holt, 2010), y como apoyo también al sector de educación en medicina (Lindquist, Johansson, Petersson, Saveman, & Nilsson, 2008). Otras áreas que han sido exploradas es el uso de celulares en el diagnóstico remoto de enfermedades (Kaplan, 2006) y, uno de los usos más comunes hoy en día es en forma de aplicaciones para smartphones que presentan información médica al usuario (Klasnja & Pratt, 2012). Al mismo tiempo, otras aplicaciones móviles están actuando como redes sociales de soporte para gente que padece de alguna condición médica (Klasnja & Pratt, 2012).

En lo que respecta a detección de caídas, varios métodos y dispositivos han sido desarrollados, ya sea en la academia o con fines comerciales: existen dispositivos portátiles que permiten al usuario presionar un botón en el caso de que sufra una caída (Mubashir, Shao, & Seed, 2013), mientras hay otros que realizan la detección automática, usualmente por medio de acelerómetros (Kangas, Konttila, Lindgren, Winblad, & Jämsä, 2008). Fuera del método específico para la detección, el común denominador es que han sido diseñados para dispositivos dedicados, que no están disponibles al consumidor. Otro problema que suele ser reportado es lo incómodo de estos dispositivos (Bourke, Van de Ven, Chaya, OLaighin, & Nelson, 2008), factor que suele ser el determinante en la aceptación de los usuarios (Demiris et al., 2004).

A fin de superar los retos existentes de entregar un sistema de detección de caídas ubicuo, con un nivel de aceptación alto, es que el autor propone el uso de beacons BLE como una solución innovadora y de bajo costo. Estos dispositivos cuentan con una vida útil de dos años sin ser recargados (Estimote Inc., 2016a), son pequeños, ligeros, inalámbricos y altamente precisos en combinación con el uso de un smartphone. Sobra decir que esta tecnología permea todos los ámbitos de nuestra sociedad hoy en día (además que ya es estándar en varias casas de reposo alrededor del mundo (Korhonen & Bardram, 2004)).

2.2. Conceptos claves

- **Computación ubicua**

  Es la integración de la computación en el entorno cotidiano, a un nivel en que las computadoras ya no se pueden diferenciar con otros objetos. Es la computación realizada de una manera omnipresente (Weiser, 1991).

- **Computación corporal**

  Es el estudio o la práctica de inventar, diseñar, construir o usar dispositivos sensoriales y computacionales diseñados para ser puestos por las personas como una prenda o accesorio más (Mann, 1996).

- **Acelerómetro**

  Es un dispositivo que mide la vibración o aceleración de movimiento de una estructura (OMEGA, s/f). Actualmente, es estándar que los dispositivos móviles contengan un acelerómetro, cuyo uso más común es el de posicionamiento satelital (GPS).

- **Beacon**

  Son sensores que transmiten información constantemente a los dispositivos electrónicos cercanos, lo que permite que dispositivos inteligentes (como smartphones o tablets) realicen acciones específicas cuando se encuentran en proximidad de un beacon (Pointr, s/f).

- **Bluetooth Low Energy (BLE)**

  También conocido como Bluetooth Smart, es una tecnología de telecomunicaciones para redes personales diseñada para aplicaciones innovadoras en salud, beacons, seguridad y entretenimiento. A comparación de Bluetooth clásico, BLE está diseñado para consumir mucho menos energía y mantener un rango de comunicación similar (Bluetooth, s/f).

- **Cloud Computing**

  Es un tipo de computación basada en el Internet, la cual provee recursos de procesamiento y almacenamiento a otros dispositivos que lo demanden. Es un modelo que permite usar recursos a demanda de una manera ubicua, con un esfuerzo mínimo de administración (Below, Hassan, & Stark, 2011).

- **mHealth**

  Es la práctica de la medicina y de la salud pública con el apoyo de dispositivos móviles como smartphones, tablets y muchos más (Adibi, 2015).

- **eHealth**

  Es la práctica del cuidado de la salud soportada por procesos electrónicos y telecomunicaciones (Della Mea, 2001). Aunque a veces se usa de manera intercambiable con mHealth, eHealth es un concepto mucho más antiguo que precede la venida de los smartphones.

- **Mockup**
- Es una representación realista de como el producto final lucirá, sin tener que construir el producto final primero. También llamado "modelo a escala" (Kieser, 2014, p.).

- **Wireframe**

  Es una representación simple de la funcionalidad del producto final, en contraste con el mockup, que está enfocado en la apariencia. Demuestra lo que se puede hacer con el diseño ("terminology - What is the difference between wireframes and mockups?", s/f).

- **Interacción humano-computador**

  Es la disciplina que trata del diseño, evaluación e implementación de sistemas de computación interactivos para uso humano y del estudio de las idiosincrasias que los rodean (ACM, 2009).

- **Context awareness**

  Es la propiedad de ciertos dispositivos móviles, y por ende el software que utilizan, de ser conscientes de su contexto con el fin de contribuir al funcionamiento del mismo (Schmidt, s/f).

- **Sensor**

  Es un dispositivo cuyo propósito es detectar eventos o cambios en su entorno, y proveer una salida correspondiente (WhatIs.com, s/f).

CAPÍTULO 3 - DISEÑO DEL SISTEMA

3.1. Arquitectura del sistema

El sistema emplea una arquitectura cliente-servidor donde existen dos clientes: Paciente y Cuidador, los cuales se comunican con el servidor en la nube que contiene la base de datos. La comunicación entre clientes y servidor se da a través de Servicios Web REST (Representational State Transfer). Veremos más sobre REST en la sección de **Implementación**, pero básicamente nos permiten interactuar con entidades a través de métodos HTTP (GET, POST, PUT).

Adicionalmente, el Paciente actúa como una especie de servidor para los Beacons, ya que estos últimos están constantemente emitiendo señales que el Paciente se encarga de recibir y procesar en su teléfono. Los detalles del flujo de información se verán en la siguiente sección.

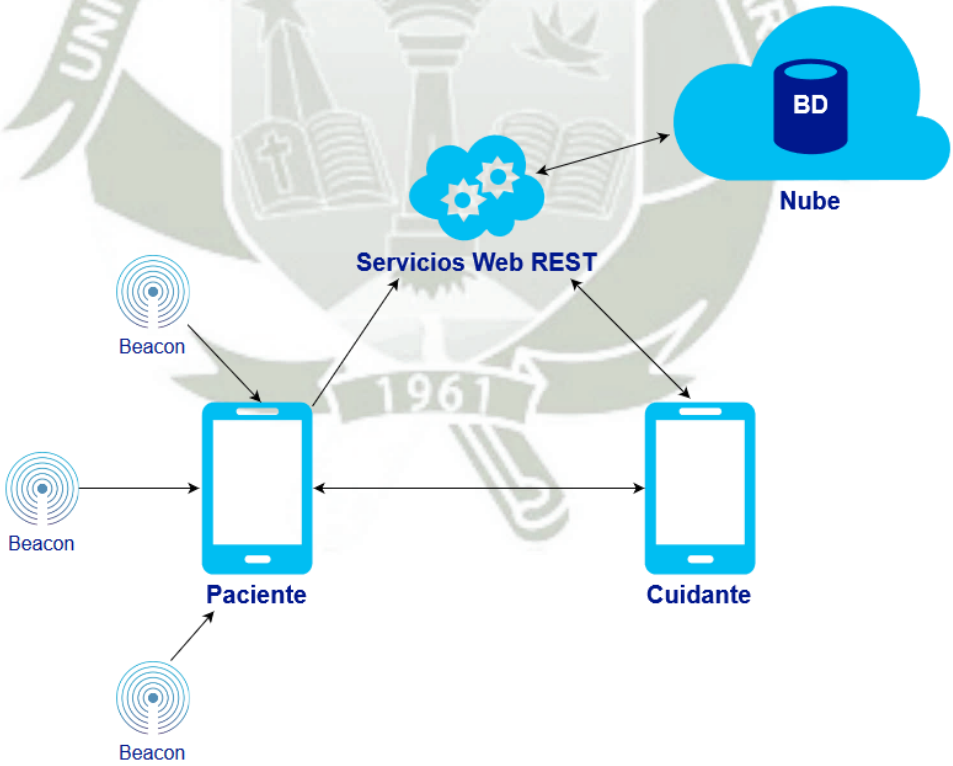

Ilustración 1: Arquitectura del sistema.

3.2. Flujo de la información

El flujo principal de la información consiste en detectar una emergencia en el paciente y todo el proceso necesario para lograr que éste obtenga asistencia. Una vez que el Cuidador recibe notificación de la emergencia, se considera como tratada, a menos que el Cuidador no haya respuesta de parte del Cuidador. Dicho flujo se puede apreciar en la **Ilustración 2.**

Además del flujo mencionado existen otros que no fueron ilustrados, como el de crear un usuario, obtener los pacientes a cargo de un Cuidador, crear y leer configuraciones. Todos estos flujos son relativamente simples: el dispositivo se conecta con el servidor en la nube e inserta o lee los datos correspondientes. Se trata de operaciones CRUD, que no ameritan una explicación detallada.

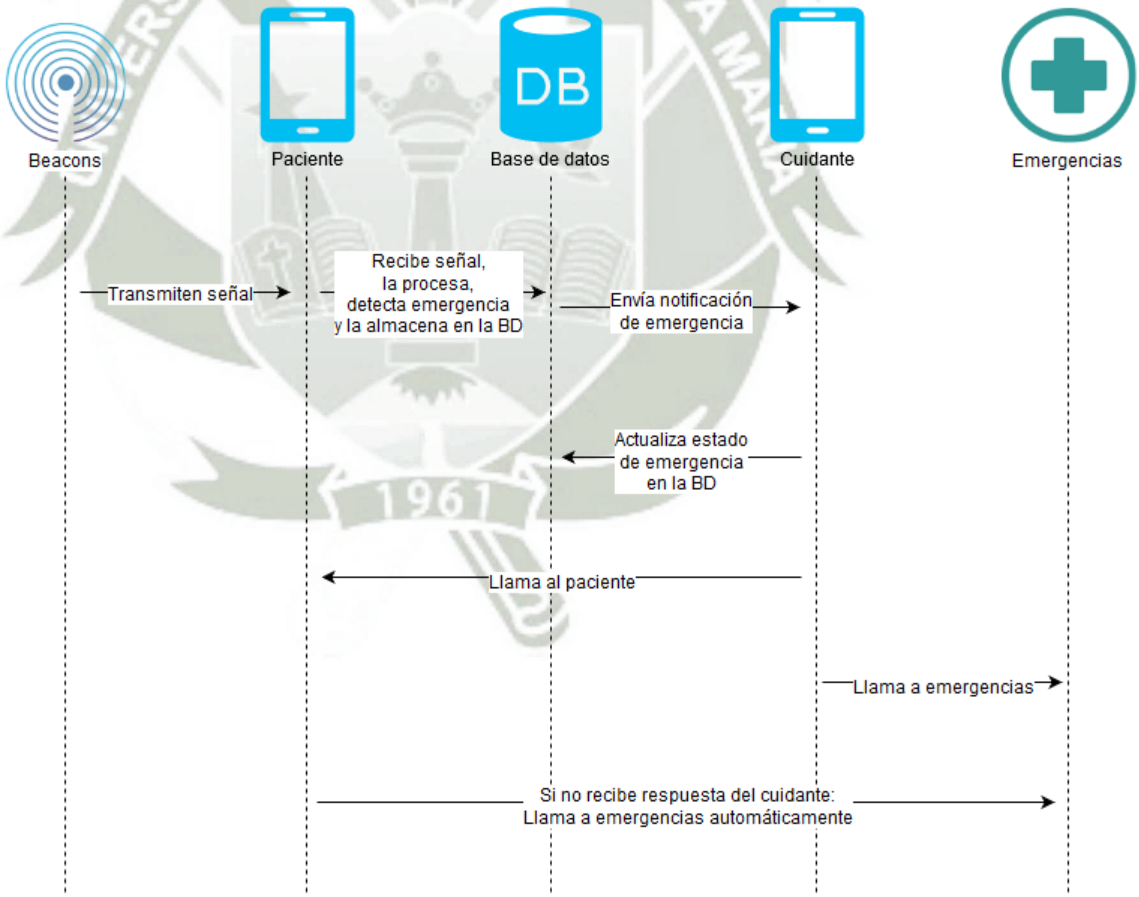

Ilustración 2: Flujo de la información.

3.3. Diseño de la Base de Datos

3.3.1. Base de datos Relacional vs NoSQL

En el diseño de la base de datos se consideró utilizar un modelo no relacional (NoSQL). Las bases de datos relacionales son aquellas que almacenan la información de una manera bastante estructurada (tablas, columnas, filas), usualmente con un conjunto de restricciones que rigen las relaciones entre estas entidades. En contraste, una **base de datos no relacional** (NoSQL) está basada en una serie de **documentos** que carecen de una estructura fija: cada documento puede tener los campos que requiera, y, aun así pertenecer a un conjunto de entidades del mismo tipo, normalmente carecen de restricciones (e.g. claves foráneas) (Leavitt, 2010). El uso de BD NoSQL ha incrementado de manera considerable, dado a que se presta a las características de la Web (específicamente Web 2.0+), en contraste con el modelo relacional cuyo origen se remonta al nivel de estructuración y rigidez requerido por los grandes sistemas transaccionales que han dominado el mundo de la computación hasta hace unos quince años (Leavitt, 2010). Los exponentes más grandes de NoSQL son Facebook, Amazon y Google (Mohan, 2013).

Se realizó un diseño rápido de un esquema NoSQL para la BD del sistema pero se encontró que había considerable repetición de datos y nunca se llegó a encontrar un esquema que realmente se acomode a las necesidades del negocio. En conclusión, la naturaleza rígida de las entidades que vamos a representar y el alto nivel de interdependencia que existe entre ellas se presta mucho mejor a un esquema relacional que a un esquema NoSQL.

3.3.2. Diagrama Entidad-Relación (DER)

Un DER es un diagrama en bases de datos relacionales que nos muestra gráficamente el esquema de la base de datos, como su nombre lo dice nos muestra las entidades (las tablas) y sus relaciones (las claves foráneas). A continuación podemos observarlo. Las flechas indican las claves foráneas. *1…** indica una relación de cardinalidad de *Uno a Muchos*, donde *Uno* es el origen de la flecha y *Muchos*, el destino, respectivamente. Esto significa que la entidad originaria el *Uno* representa un valor único, mientras *Muchos* indica que la entidad destino puede tener muchas referencias al valor original. La otra relación que existe es *…*, indicando una relación de *Muchos a Muchos.*

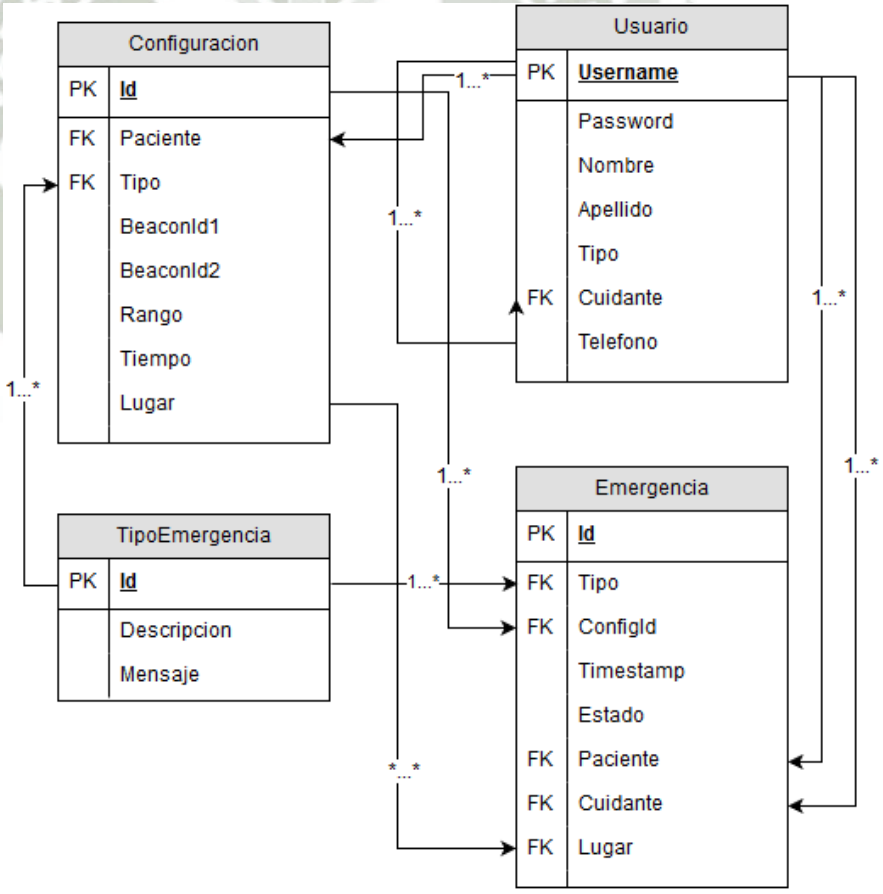

Ilustración 3: Diagrama Entidad-Relación

La tabla **Usuario** es autoexplicativa, contiene información del usuario como nombres y apellidos, usuario y contraseña, etc. Si el usuario es un Paciente necesita especificar el nombre de usuario de su Cuidador.

La tabla **TipoEmergencia** contiene básicamente los tipos de algoritmo que soporta nuestra aplicación (actualmente cuatro, que los veremos en la sección **Algoritmos**). Está diseñada como tabla para la opción de añadir más algoritmos en el futuro.

La tabla **Configuración** es una abreviación de Configuración de Alerta. Esta entidad representa a una configuración específica creada por el usuario la cual se utilizará para la detección de emergencias. Ejemplo: una configuración diseñada para el cruce de las gradas de la cocina, la cual utiliza los Beacons con Id 47 e Id 56. Las configuraciones están asociadas a pacientes.

La tabla **Emergencia** es la que representa las emergencias generadas por un paciente, por razones de conveniencia y rendimiento contiene campos calculables, como los nombres del paciente y Cuidador y el lugar de la emergencia. Como esta tabla será consultada a intervalos constantes e indefinidamente para generar las notificaciones, el tener todos estos campos opcionales listos nos evita tener que estar haciendo inner joins (una de las operaciones más costosas) cada vez que es consultada.

3.3.3. Diccionario de datos

El diccionario de datos es una herramienta de documentación en la que se explica a detalle cada campo (columna) de cada tabla. A continuación se explica cada columna de la tabla:

- **Tabla**: nombre de la tabla a la que pertenece el campo
- **PK/FK**: si el campo en mención es clave primaria (PK) or clave foránea (FK)
- **Inc**: Si el campo es auto**inc**rementable (Identity), quiere decir que el campo es autogenerado con cada registro nuevo y es un número
- **Nombre**: nombre del campo
- **Tipo**: tipo de dato que almacena el campo
- **n:** Indica el tamaño del campo, según el tipo de datos, usualmente solo aplica a caracteres
- **Permite nulos?:** 1 si el campo permite valores nulos, 0 si no
- **Campo que referencia:** Aplica para claves foráneas. Indica el campo original al que se hace referencia
- **Descripción:** Una descripción del contenido o propósito del campo

Tabla	PK/ FK	Inc.	Nombre	Tipo	n	Nulos?	Referencia	Descripción
Usuario	PK	0	Username	nvarchar	50	0	--	Nombre de usuario único. Usado al iniciar sesión.
Usuario	--	0	Password	nvarchar	128	0	--	La contraseña del usuario
Usuario	--	0	Nombre	nvarchar	50	0	--	Su nombre real. Usado en las interfaces.
Usuario	--	0	Apellido	nvarchar	50	0	--	Apellido del usuario
Usuario	--	0	Tipo	bit	--	0	--	Tipo de usuario: 0->paciente, 1->Cuidador
Usuario	FK	0	Cuidador	nvarchar	50	1	Usuario.Username	El nombre de usuario del Cuidador. Obligatorio si el tipo de usuario es paciente.
Usuario	--	0	Telefono	nvarchar	20	0	--	El teléfono de contacto del usuario
TipoEmergencia	PK	1	Id	int	--	0	--	Entero identificador único. Campo autoincrementable.
TipoEmergencia	--	0	Descripción	nvarchar	50	0	--	El nombre identificador del tipo de emergencia.
TipoEmergencia	--	0	Mensaje	nvarchar	100	0	--	Una descripción más verbosa si se requiere.
Configuracion	PK	1	Id	int		0	--	Entero identificador único. Campo

								autoincrementable.
Configuracion	FK	0	Lugar	nvarchar	50	0	--	Nombre para la identificación. Suele usarse el sitio donde se encuentran los beacons (e.g. baño, escaleras)
Configuracion	FK	0	Paciente	nvarchar	50	0	Usuario.Username	Nombre de usuario del paciente asociado a la configuración
Configuracion	--	0	Tipo	int	--	0	TipoEmergencia.Id	Tipo de emergencia procesado por la configuración
Configuracion	--	0	BeaconId1	int	--	0	--	El número que identifica al primer beacon configurado
Configuracion	--	0	BeaconId2	int	--	1	--	El número que identifica al segundo beacon configurado. Requerido para aquellas configuraciones que usan dos beacons.
Configuracion	--	0	Rango	int	--	0	--	Número que indica la distancia relativa del celular a los beacons usada para la detección: 1->Inmediata, 2->Cerca, 3->Lejos
Configuracion	--	0	Tiempo	int	--	0	--	Tiempo en milisegundos utilizado para los algoritmos de detección.

Configuracion	--	0	Hora	time	--	0	--	Hora del día. Usada para emergencias del tipo ProximidadPorPeriodoAHora
Emergencia	PK	1	Id	int	--	0	--	Entero identificador único. Campo autoincrementable.
Emergencia	FK	0	Tipo	int	--	0	TipoEmergencia.Id	Tipo de emergencia procesado por la configuración asociada a la emergencia
Emergencia	FK	0	ConfigId	int	--	0	Configuracion.Id	Id de la configuración asociada a la emergencia
Emergencia	--	0	Timestamp	datetime	--	0	--	Hora y fecha cuando se detectó la emergencia
Emergencia	--	0	Estado	bit	--	0	--	Estado de la emergencia: 0->Creada, 1->Vista (por el Cuidador)
Emergencia	FK	0	Paciente	nvarchar	50	0	--	Paciente que generó la emergencia.
Emergencia	FK	0	Cuidador	nvarchar	50	0	--	Cuidador del paciente que generó la emergencia.
Emergencia	FK	0	Lugar	nvarchar	50	0	Configuracion.Lugar	Nombre identificador de la configuración. Usualmente el lugar.

Tabla 1: Diccionario de datos

3.4. Casos de uso

Los casos de uso son un tipo de diagramas, definidos en el lenguaje UML (OMG, 2016), que nos permite ilustrar la interacción que ocurre entre los actores (usuarios) y el sistema en distintos contextos. Las burbujas también se llaman casos de uso y representan una acción concreta, mientras las flechas punteadas, el tipo de asocación. Una asociación <<*include*>> indica que el caso de uso precedente (el origen de la flecha) consiste necesariamente del caso de uso consecuente (el destino de la flecha). La asociación <<*extends*>> indica que el consecuente es opcional.

3.4.1. Inicio de sesión

Esta pantalla le da la bienvenida al usuario y le permite iniciar sesión o crear una cuenta si es que todavía no cuenta con una.

Ilustración 4: UC - Inicio de sesión

3.4.2. Creación de usuario

Permite poder registrar un nuevo Paciente o Cuidador en el sistema. Como cada Paciente está vinculado a un Cuidador, cuando se selecciona este tipo de usuario debe indicar el nombre de usuario de su Cuidador. Debido a la edad avanzada de los Pacientes se espera que el Cuidador se encargue de crearles la cuenta e iniciar sesión por primera vez.

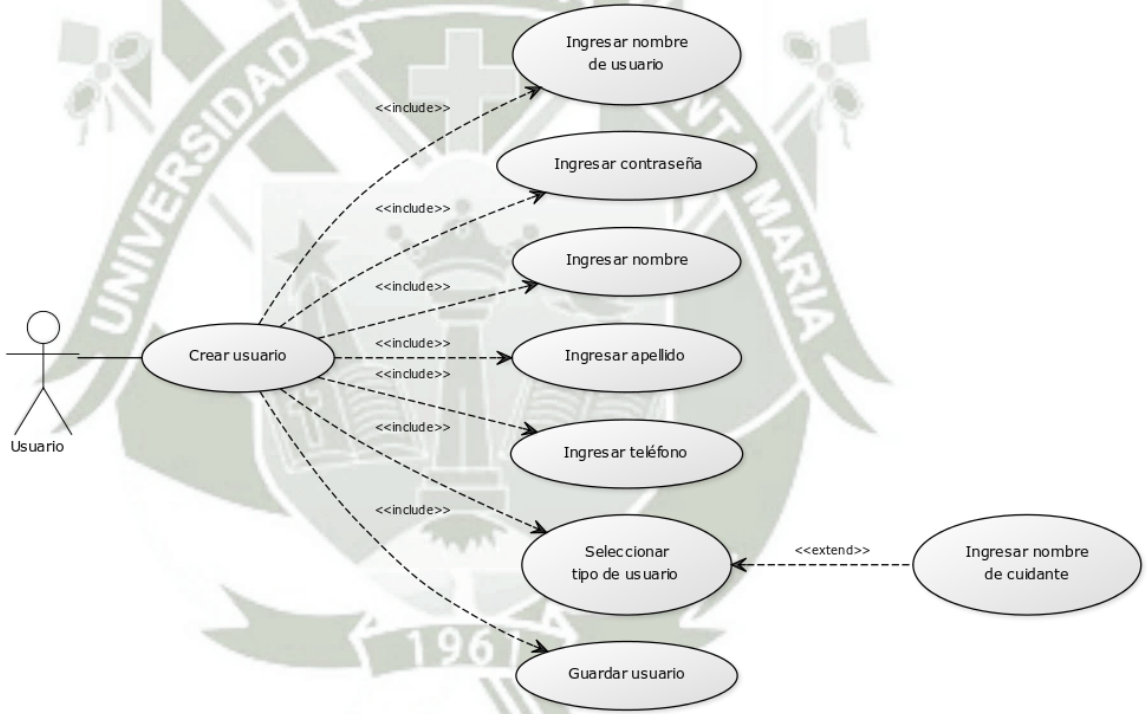

Ilustración 5: UC - Creación de Usuario

3.4.3. Pantalla principal del Paciente y creación de Configuración

La pantalla principal del Paciente es la que se le muestra después del inicio de sesión. Se le presenta la lista de configuraciones existentes y permite crear nuevas. Más allá de la configuración inicial del sistema no se requiere interacción alguna por parte del Paciente y simplemente podría guardar su teléfono en el bolsillo (o donde prefiera). En la **Ilustración 7** podemos observar un asterisco al lado del Paciente, esto indica que aunque técnicamente la creación de configuraciones debe ser llevada a cabo con el usuario del Paciente, esto debería ser hecho por el Cuidador.

Cabe señalar también que el caso de uso *Ingresar Hora* es opcional porque solo es requerido por el tipo de emergencia de ProximidadXHora.

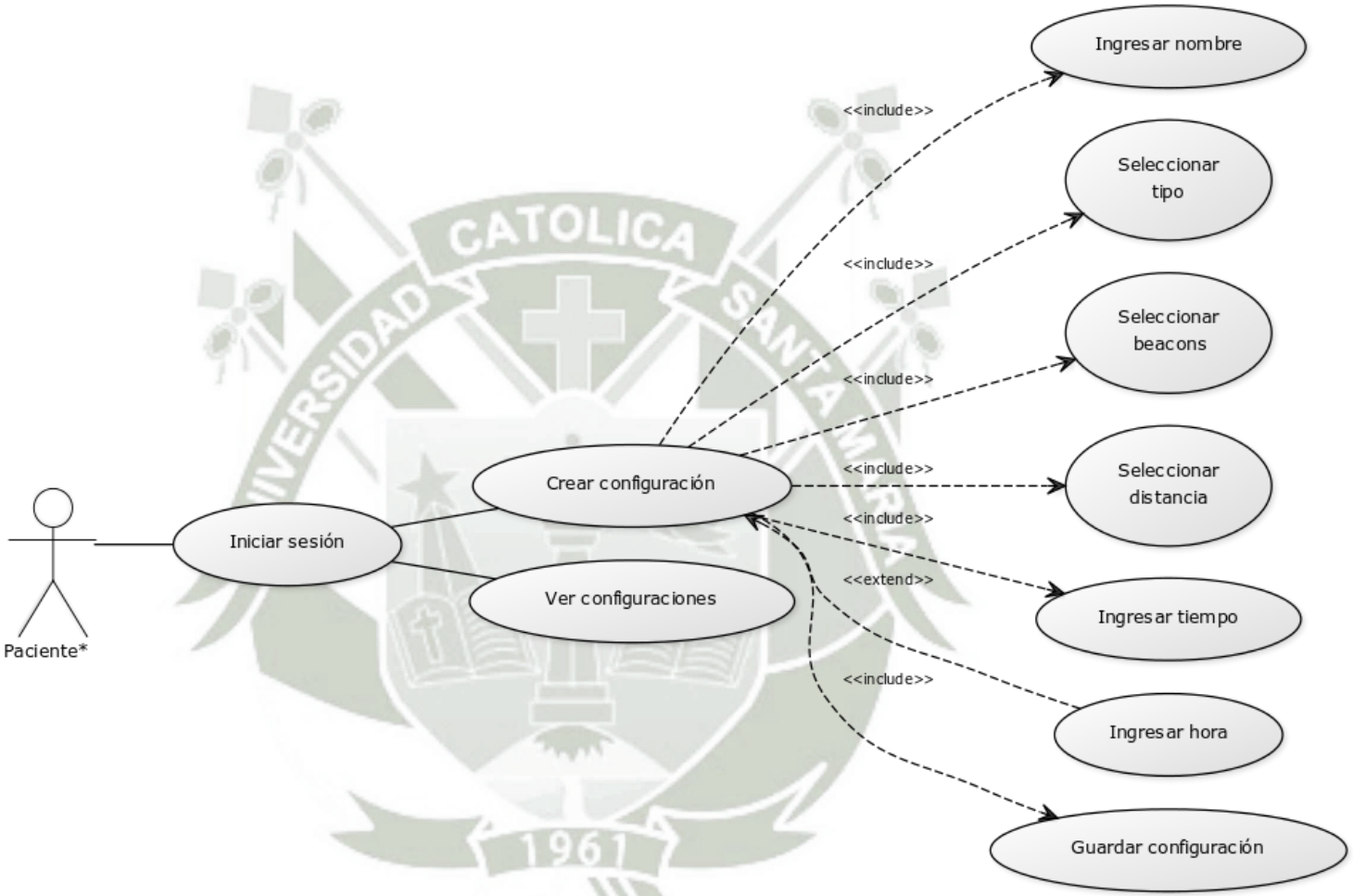

Ilustración 6: UC - Creación de Configuración

3.4.4. Pantalla principal del Cuidador

La interfaz principal del Cuidador permite ver a todos los pacientes a su cargo o crear nuevos pacientes. Adicionalmente, puede revisar el historial de emergencias de cada paciente y si lo deseara, llamarlo.

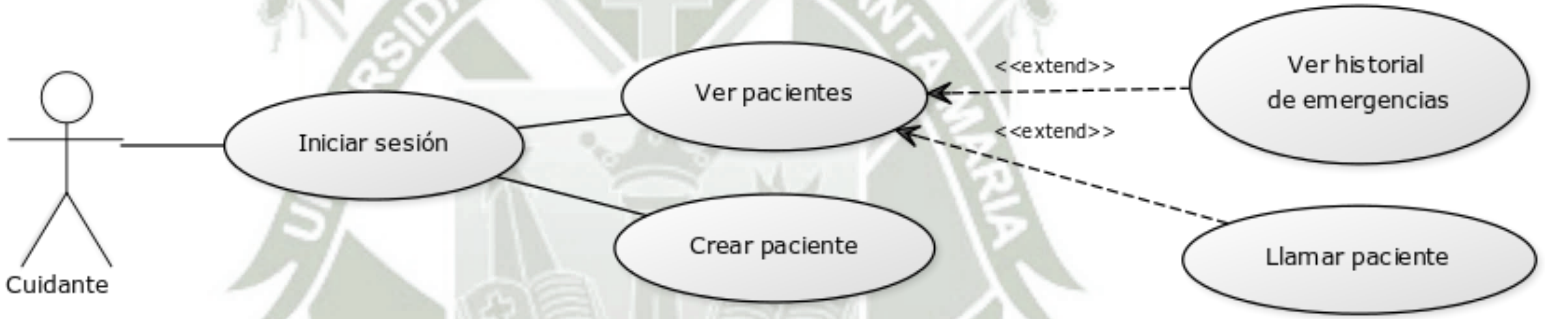

Ilustración 7: UC - Pantalla principal del Cuidador

3.4.5. Pantalla de emergencia del Paciente

La pantalla de emergencia es aquella que se le muestra al paciente cuando el sistema ha detectado una posible emergencia. Si es que resulta siendo un falso positivo el paciente tiene un periodo de tiempo en el que puede indicar que se encuentra bien. Si no, la emergencia es comunicada al Cuidador de manera automática, como esta acción no requiere intervención por parte del usuario no es ilustrada como un caso de uso.

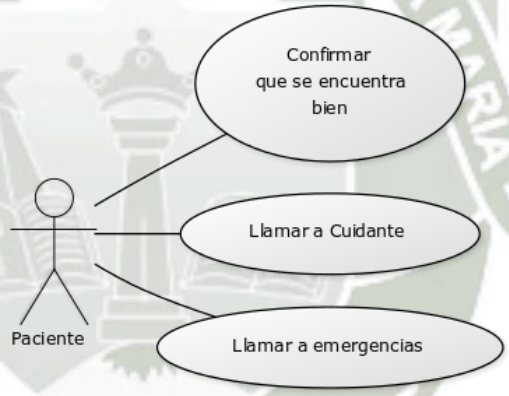

Ilustración 8: UC - Pantalla Emergencia del Paciente

3.4.6. Pantalla de emergencia del Cuidador

Una vez que el Cuidador recibe notificación de la emergencia tiene dos opciones: contactar al Paciente o contactar a Emergencias.

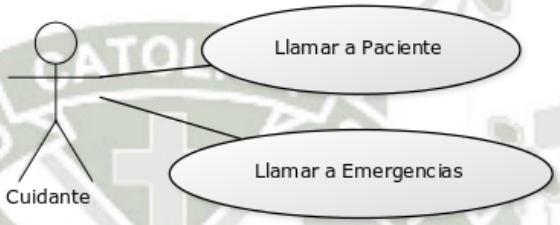

Ilustración 9: UC - Pantalla Emergencia del Cuidador

3.5. Algoritmos de detección

Se diseñaron cuatro algoritmos para detección de caídas o emergencias en base a la información transmitida por los Beacons y a la información que podemos calcular en base a ellos. Se consideró que estos cuatro algoritmos cubren la mayoría de situaciones y hacen un uso eficiente de los recursos debido a su simplicidad. Los algoritmos están descritos en seudocódigo; en una sección posterior se ilustrará exactamente cómo fueron implementados. Adicionalmente, se ilustró cada algoritmo con un diagrama de actividad en el que se muestra el flujo del algoritmo a raíz de las acciones del usuario.

3.5.1. Proximidad Por Periodo

La proximidad por periodo ocurre cuando el Paciente se encuentra en un área por más tiempo del que se planteó: e.g. el paciente se encuentra en el baño por más de una hora. Esto se logra posicionando un beacon en el baño, el sistema se encarga del resto.

```
FUNC ProximidadXPeriodo(BeaconId, Distancia, TiempoLimite)
  WHILE(TRUE)
    TiempoTranscurrido = 0
    IF (EstaEnRango(BeaconID, Distancia))
      TiempoTranscurrido++
    ELSE
      RETURN
    END IF
    IF (TiempoTranscurrido >= TiempoLimite)
      CrearEmergencia()
      RETURN
    END IF
  END WHILE
END FUNC
```

Ilustración 10: Algoritmo de Proximidad Por Periodo

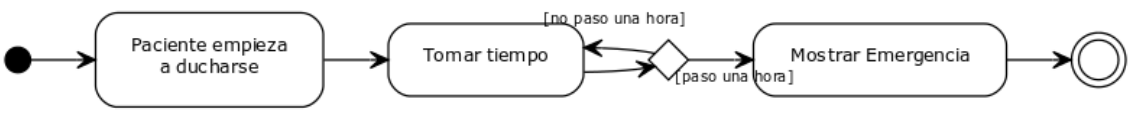

Ilustración 11: Diagrama de actividad del algoritmo de Proximidad Por Periodo

3.5.2. Proximidad Por Periodo A Hora

Este algoritmo es igual al anterior con la diferencia de que solo está activo a una hora específica del día: e.g. se quiere monitorear si es que el Paciente no se llega a levantar de su cama a su hora normal.

```
FUNC ProximidadXPeriodoYHora(BeaconId, Distancia,
TiempoLimite, Hora)
   WHILE(TRUE)
      TiempoTranscurrido = 0
      IF (EstaEnRango(BeaconID, Distancia) AND
          HoraSistema == (Hora + TiempoTranscurrido))
         TiempoTranscurrido++
      ELSE
         RETURN
      END IF
      IF (TiempoTranscurrido >= TiempoLimite)
         CrearEmergencia()
         RETURN
      END IF
   END WHILE
END FUNC
```

Ilustración 12: Algoritmo de Proximidad Por Periodo

Ilustración 13: Diagrama de actividad del algoritmo de Proximidad Por Periodo Y Hora

3.5.3. Cruce Rápido

El cruce rápido está diseñado para cuando el Paciente se mueve de una zona a otra y, por ende, requiere de dos beacons. La situación principal que lo ilustra es cuando el Paciente sube o baja las gradas de su casa; una persona de edad avanzada suele hacerlo a un ritmo determinado y con poca variación: e.g. dos minutos. Si se demorara 30 segundos probablemente algo inusual ocurrió.

```
FUNC CruceRapido(BeaconId1, BeaconId2, Distancia,
TiempoCruce)
   WHILE(TRUE)
      IF (EstaEnRango(BeaconId1, Distancia) AND Cruce=="NO")
         ComenzarCronometro()
         Cruce = "INICIADO"
      ELSE IF (EstaEnRango(BeaconId2, Distancia) AND
Cruce=="INICIADO")
         PararCronometro()
         Cruce = "TERMINADO"
      END IF
      IF (CRUCE == "TERMINADO" AND TiempoTranscurrido <
TiempoCruce)
         CrearEmergencia()
         CRUCE = “NO”
         RETURN
      END IF
   END WHILE
END FUNC
```

Ilustración 14: Algoritmo de Cruce Rápido

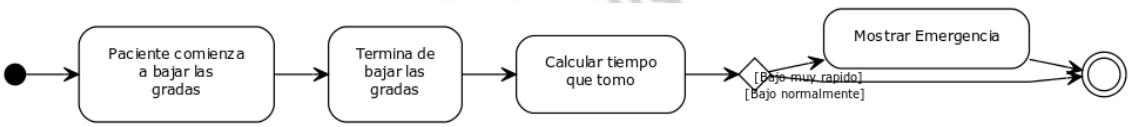

Ilustración 15: Diagrama de Actividad del algoritmo de Cruce Rápido

3.5.4. Cruce Incompleto

Este algoritmo es la contraparte del cruce rápido: aquí lo que buscamos es detectar si el paciente se está demorando más de lo usual para pasar de un lado a otro e.g.: el paciente empieza a bajar las gradas pero nunca llega al final.

```
FUNC CruceIncompleto(BeaconId1, BeaconId2, Distancia,
TiempoCruce)
   WHILE(TRUE)
      IF (EstaEnRango(BeaconId1, Distancia) AND Cruce=="NO")
         ComenzarCronometro()
         Cruce = "INICIADO"
      ELSE IF (EstaEnRango(BeaconId2, Distancia) AND
Cruce=="INICIADO")
         PararCronometro()
         Cruce = "TERMINADO"
      END IF
      IF (CRUCE == "TERMINADO" AND TiempoTranscurrido >
TiempoCruce)
         CrearEmergencia()
         CRUCE = “NO”
         RETURN
      END IF
   END WHILE
END FUNC
```

Ilustración 16: Algoritmo de Cruce Incompleto

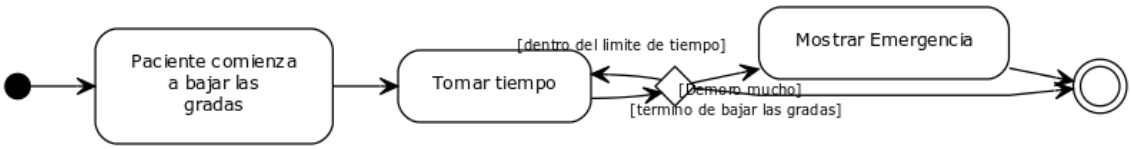

Ilustración 17: Diagrama de Actividad del algoritmo de Cruce Incompleto

3.6. Diseño de la Interfaz

En esta sección se ilustra cada pantalla que tendrá la aplicación. A continuación se muestra un diagrama con estructura árbol para explicar el flujo de cada pantalla.

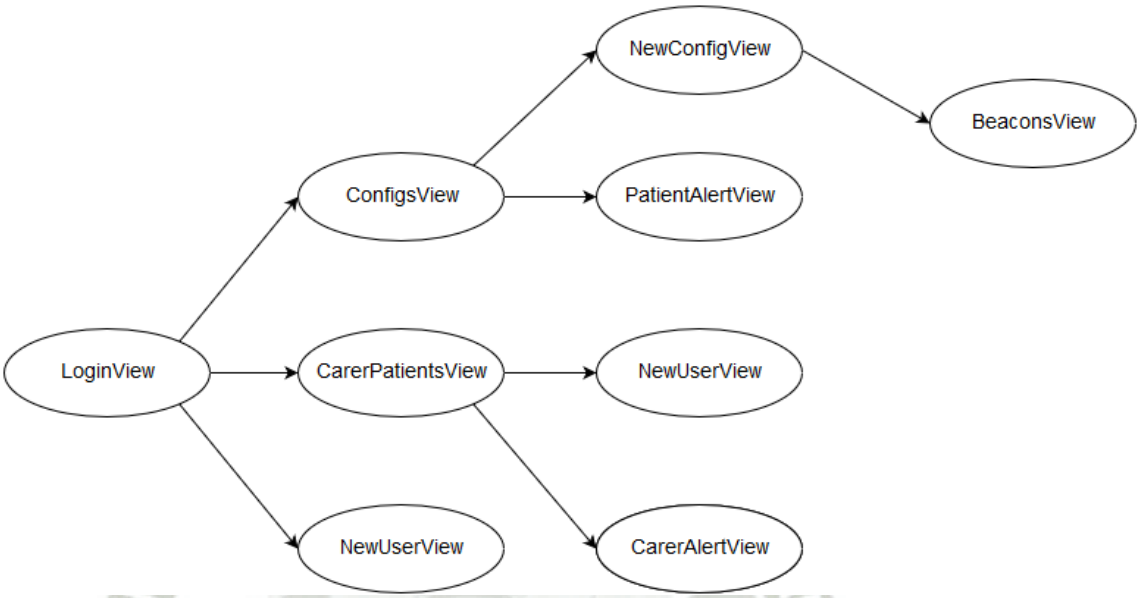

Ilustración 18: Estructura de las interfaces del sistema.

3.6.1. LoginView

Es la interfaz inicial que se le presenta al usuario. Permite iniciar sesión o sino crear un nuevo usuario.

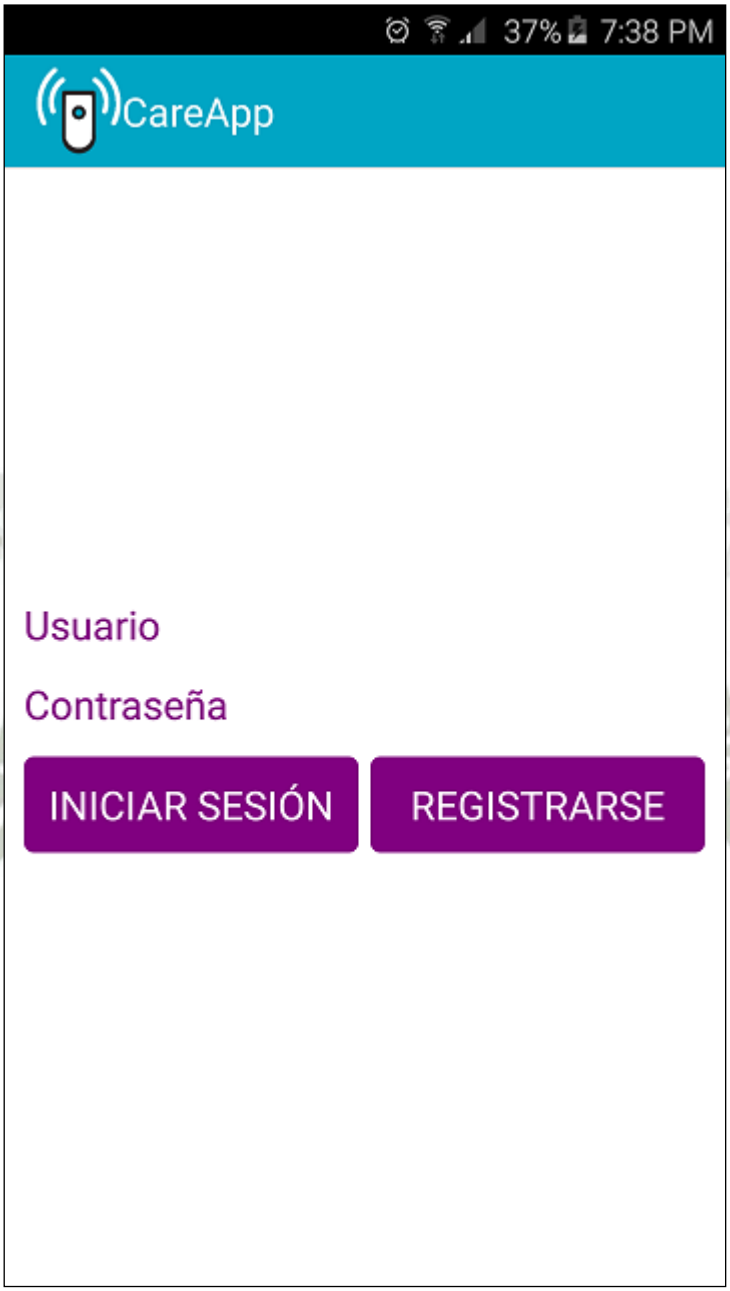

Ilustración 19: Interfaz LoginView

3.6.2. NewUserView

Es la interfaz que maneja la creación de nuevos usuarios. Es reutilizada también para la modificación de los usuarios.

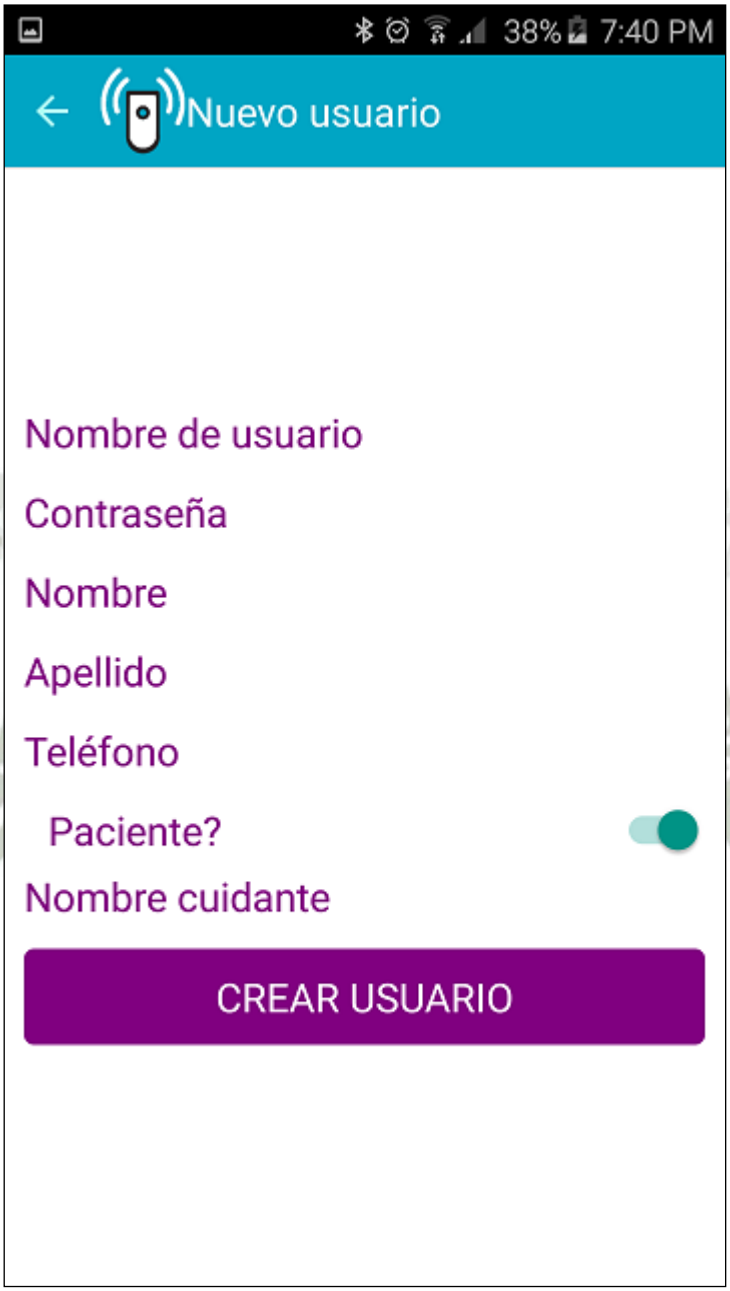

Ilustración 20: Interfaz NewUserView

3.6.3. ConfigsView

Es la pantalla principal de un usuario del tipo Paciente, se le muestran todas las configuraciones asociadas a su cuenta y la posibilidad de crear una nueva configuración de emergencia.

Ilustración 21: Interfaz ConfigsView

3.6.4. NewConfigView

Es la interfaz que permite crear nuevas configuraciones. También es reutilizada para la modificación de configuraciones existentes.

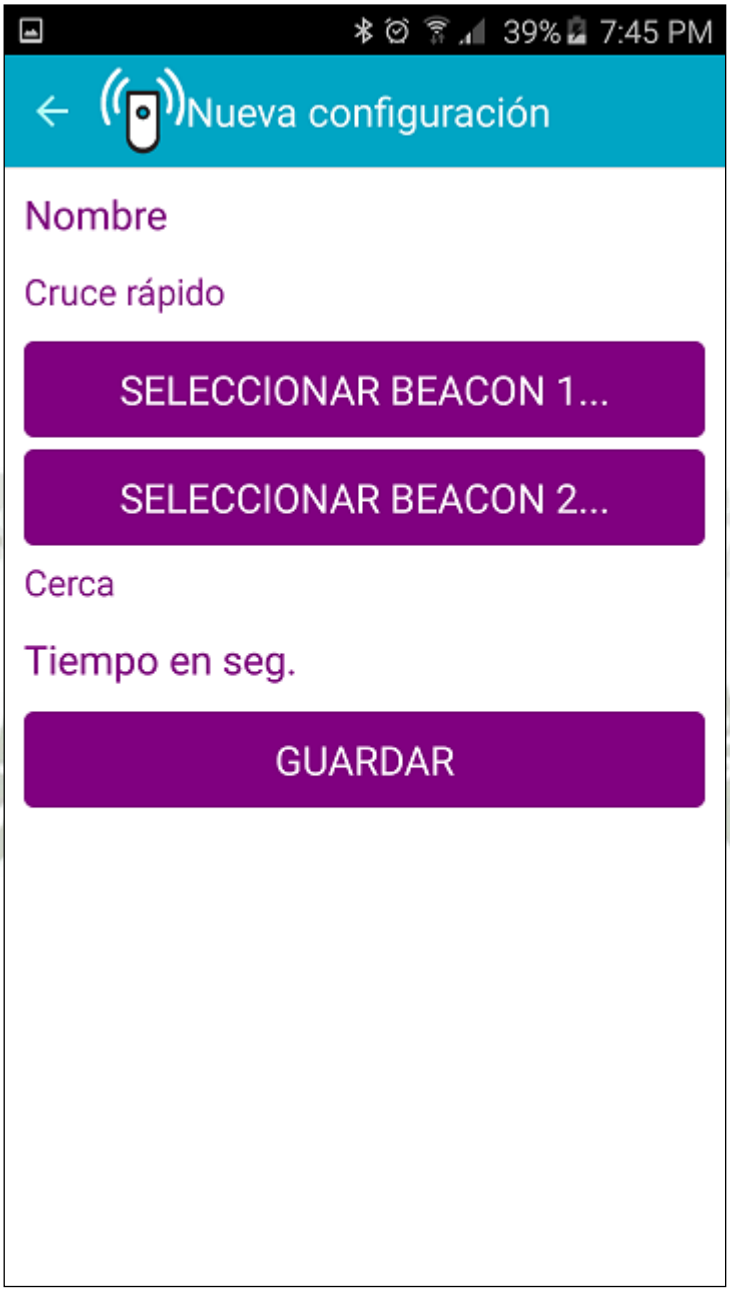

Ilustración 22: Interfaz NewConfigView

3.6.5. BeaconsView

Esta es la interfaz que se encarga de mostrar al usuario los beacons que se detectan a su alrededor. Es invocada cuando se crea o modifica una configuración.

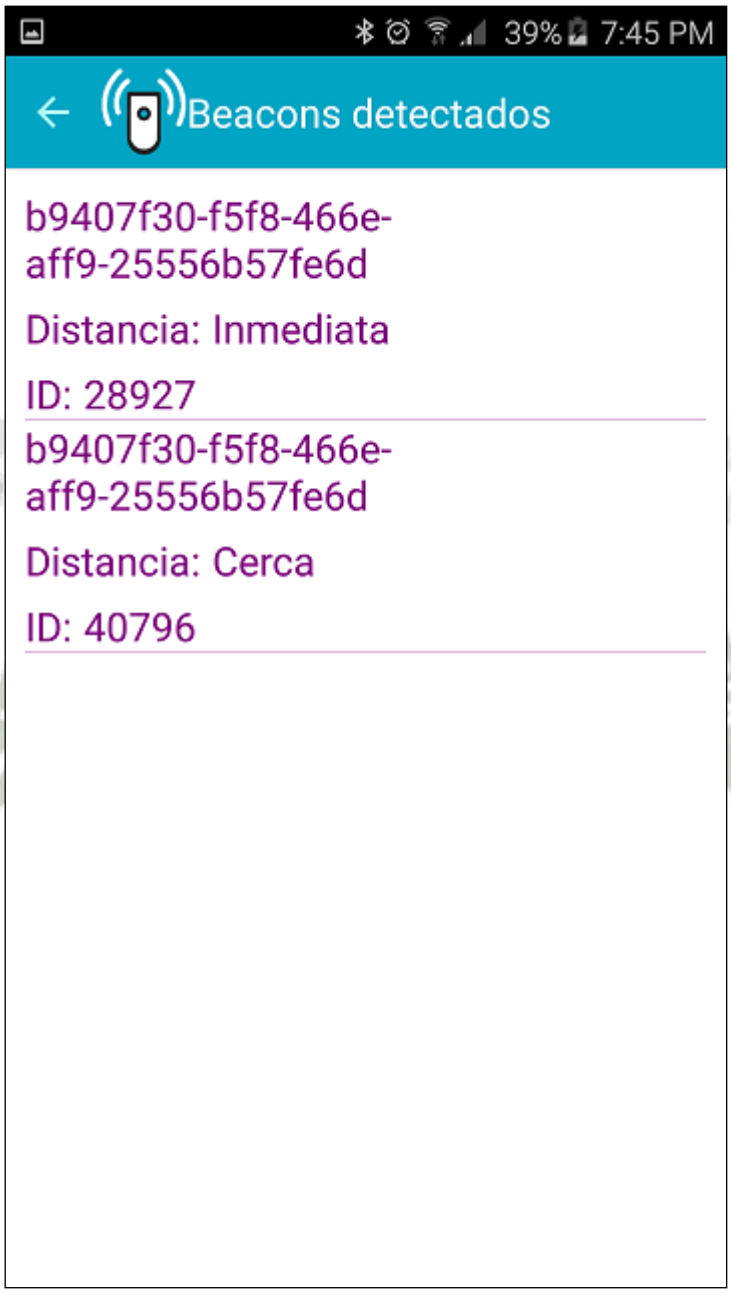

Ilustración 23: Interfaz BeaconsView

3.6.6. PatientAlertView

Es la interfaz que se le muestra al usuario cuando se ha detectado una posible emergencia. El paciente tiene la opción de marcar la emergencia como falso positivo (“estoy bien”) o sino de llamar a su Cuidador o a emergencias. Si no se obtiene respuesta del paciente se crea una emergencia que será notificada al Cuidador.

Ilustración 24: Interfaz PatientAlertView

3.6.7. CarerPatientsView

La interfaz principal del Cuidador una vez que ha iniciado sesión. Muestra todos los pacientes de los cuales se encarga y también permite añadir pacientes si se desea, para este último se usa el mismo formulario de creación de usuario con la diferencia de que el campo Paciente ha sido checkeado y en nombre de Cuidador está el del Cuidador actual.

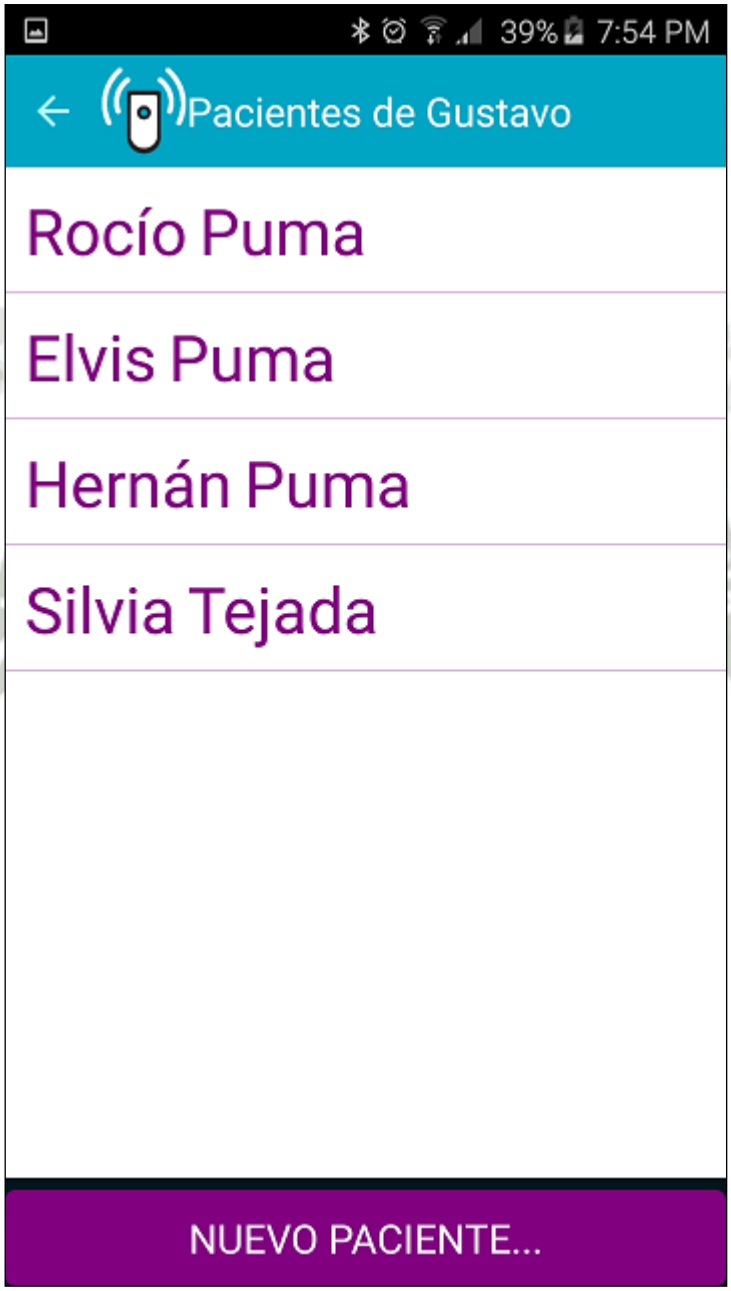

Ilustración 25: Interfaz CarerPatientsView

3.6.8. CarerAlertView

Esta es la interfaz de alerta del Cuidador. Se le comunica un mensaje detallando la emergencia y se le da la opción de llamar al Paciente o a emergencias.

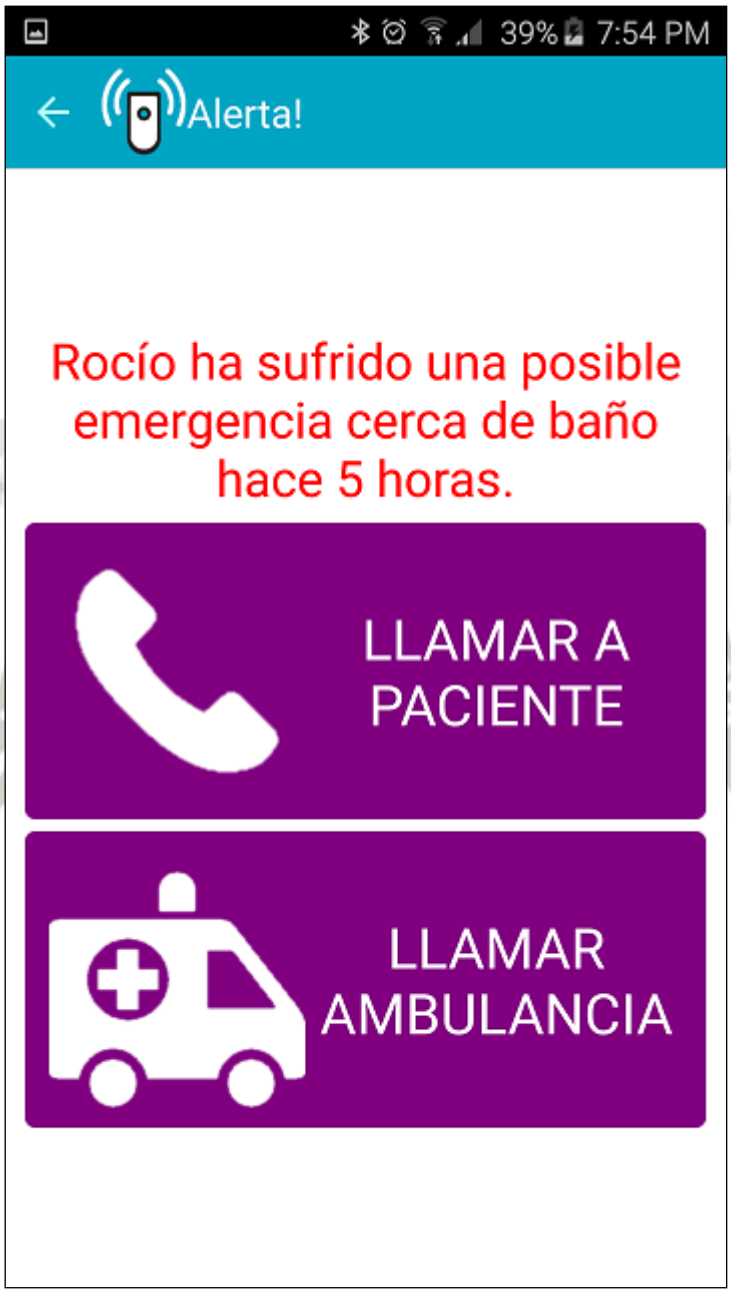

Ilustración 26: Interfaz CarerAlertView

CAPÍTULO 4 - IMPLEMENTACIÓN DEL SISTEMA

4.1. Base de datos

La base de datos fue implementada en SQL Server 2016 con el nombre de CareDB. En las siguientes ilustraciones se puede apreciar los scripts de creación correspondientes.

```
CREATE TABLE [dbo].[Usuario]
(
        [Nombre] NVARCHAR(50) NOT NULL,
        [Apellido] NVARCHAR(50) NOT NULL,
        [Username] NVARCHAR(50) NOT NULL PRIMARY KEY,
        [Password] NVARCHAR(128) NOT NULL,
        [Tipo] BIT NOT NULL,
        [Cuidante] NVARCHAR(50) NULL,
        [Telefono] NVARCHAR(20) NOT NULL,
        CONSTRAINT [FK_Usuario_Usuario_Cuidante] FOREIGN KEY ([Cuidante])
        REFERENCES [Usuario]([Username])
)
```

Ilustración 27: Script de creación de la Tabla Usuario

```
CREATE TABLE [dbo].[TipoEmergencia]
(
        [Id] INT NOT NULL PRIMARY KEY,
        [Descripcion] NVARCHAR(50) NOT NULL,
        [Mensaje] NVARCHAR(100) NOT NULL
)
```

Ilustración 28: Script de creación de la Tabla TipoEmergencia

```
CREATE TABLE [dbo].[Configuracion]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
        [Paciente] NVARCHAR(50) NOT NULL,
        [Tipo] INT NOT NULL,
        [BeaconId1] INT NOT NULL,
        [BeaconId2] INT NULL,
        [Rango] INT NOT NULL,
        [Tiempo] INT NOT NULL,
        [Nombre] NVARCHAR(50) NOT NULL,
        [Hora] DATETIME NULL,
         CONSTRAINT [FK_Configuracion_Usuario_Paciente]
         FOREIGN KEY ([Paciente]) REFERENCES [Usuario]([Username]),
         CONSTRAINT [FK_Configuracion_TipoEmergencia_Tipo]
         FOREIGN KEY ([Tipo]) REFERENCES [TipoEmergencia]([Id])
)
```

Ilustración 29: Script de creación de la Tabla Configuracion

```
CREATE TABLE [dbo].[Emergencia]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
        [Tipo] INT NOT NULL,
        [ConfigId] INT NOT NULL,
        [Timestamp] DATETIME NOT NULL,
        [Estado] BIT NOT NULL,
        [Paciente] NVARCHAR(50) NOT NULL,
        [Cuidante] NVARCHAR(50) NOT NULL,
        [Lugar] NVARCHAR(50) NOT NULL,
        CONSTRAINT [FK_Emergencia_TipoEmergencia_Tipo]
         FOREIGN KEY ([Tipo]) REFERENCES [TipoEmergencia]([Id]),
        CONSTRAINT [FK_Emergencia_Configuracion_ConfigId]
         FOREIGN KEY ([ConfigId]) REFERENCES [Configuracion]([Id]),
        CONSTRAINT [FK_Emergencia_Usuario_Paciente]
         FOREIGN KEY ([Paciente]) REFERENCES [Usuario]([Username]),
        CONSTRAINT [FK_Emergencia_Usuario_Cuidante]
         FOREIGN KEY ([Cuidante]) REFERENCES [Usuario]([Username])
)
```

Ilustración 30: Script de creación de la Tabla Emergencia

4.2. Xamarin.Forms

La parte aplicativa del prototipo fue desarrollado en Xamarin.Forms: el módulo del framework Xamarin que permite desarrollar aplicaciones multiplataforma utilizando una misma base de código (Xamarin Inc., 2016). El lenguaje de programación utilizado es C# y el lenguaje de marcado, el cual es usado para crear las interfaces de usuario, es XAML, una versión modificada del estándar XML. Aunque se puede crear una aplicación enteramente en C#, el uso de

XAML refuerza el concepto de separación de intereses (en inglés *separation of concerns)* ya que crea una división explícita entre interfaz gráfica y lógica, o *Vista* y *Controlador* en términos de la arquitectura MVC.

4.3. Modelos

Los modelos son las entidades que representan la estructura de los objetos en nuestra fuente de datos, en este caso la BD. Sirven para crear un enlace entre la capa de negocios y la de datos. Los modelos con los que cuenta el prototipo se muestran a continuación. Los atributos con la etiqueta *[JsonIgnore]* no corresponden al modelo directamente, pero son usados para almacenar datos de interés. Estos atributos son de interés en el proceso de **Serialización**.

```
public class Usuario
{
    public string Nombre { get; set; }
    public string Apellido { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
    public bool Tipo { get; set; }
    public string Cuidante { get; set; }
    public string Telefono { get; set; }

    [JsonIgnore]
    public string TelCuidante { get; set; }
    [JsonIgnore]
    public List<Usuario> Pacientes { get; set; }
    [JsonIgnore]
    public List<EmergencyConfig> Configuraciones { get; set; }
}
```

Ilustración 31: Modelo Usuario

```
public class EmergencyConfig
{
    public int Id { get; set; }
    public string Nombre { get; set; }
    public int Tipo { get; set; }
    public ushort BeaconId1 { get; set; }
    public ushort BeaconId2 { get; set; }
    public int Rango { get; set; }
    public int Tiempo { get; set; }
    public string Paciente { get; set; }
}
```

Ilustración 32: Modelo Configuración de Emergencia

```
public class Emergencia
{
    public int Id { get; set; }
    public int Tipo { get; set; }
    public int ConfigId { get; set; }
    public DateTime Timestamp { get; set; }
    public bool Estado { get; set; }
    public string Paciente { get; set; }
    public string Cuidante { get; set; }
    public string Lugar { get; set; }
}
```

Ilustración 33: Modelo Emergencia

```
public class Beacon
{
    public string Uuid { get; set; }
    public ushort Major { get; set; }
    public ushort Minor { get; set; }
    public string RelativeDistance { get; set; }
    public string Type { get; set; }
}
```

Ilustración 34: Modelo Beacon. No corresponde a ninguna entidad en la base de datos pero es usado por los algoritmos de detección.

4.4. Servicios Web REST

Representational State Transfer (REST) es una arquitectura que define una serie de principios sobre los cuales podemos diseñar servicios web que nos permiten utilizar un amplio número de clientes, escritos en diferentes lenguajes, sin problema alguno, ya que utiliza HTTP como medio de transporte y JSON como lenguaje interoperacional ("RESTful Web services", 2015). JSON (ECMA International, 2016) es un formato abierto expresado en texto plano para transmitir objetos representados como pares de atributos y llaves, actualmente ha logrado reemplazar casi completamente a XML como el lenguaje de elección para comunicación de cliente y servidor a través de la Web. El prototipo implementado en este trabajo utiliza Servicios REST para la comunicación entre cliente y servidor.

4.4.1. sandman2

sandman2 es un programa que permite generar una API de servicio REST a partir de una base de datos existente, sin escribir una sola línea de código ("jeffknupp/sandman2", s/f). Una vez instalado el programa y con nuestra base de datos **CareDB** puesta en marcha, el siguiente comando fue ejecutado para generar nuestra API REST:

```
sandman2ctl
mssql+pyodbc://@localhost\SQLEXPRESS/CareDB?driver=SQL+Server+Native+Client+11.0
```

Ilustración 35: Comando para crear una API REST utilizando sandman2 sobre una BD en SQL Server con nombre CareDB

Una vez ejecutado el comando, podemos acceder a una representación gráfica de nuestra API REST a través de un navegador como podemos ver en la **Ilustración 34.** Para acceder al contenido de nuestra base de datos basta con especificarlo en la URL como vemos en las **Ilustraciones 35 y 36**, en estas ilustraciones podemos observar que los

datos son representados como texto plano en el lenguaje JSON, lo que hace conveniente diseñar un programa en cualquier lenguaje que haga interfaz con este tipo de APIs.

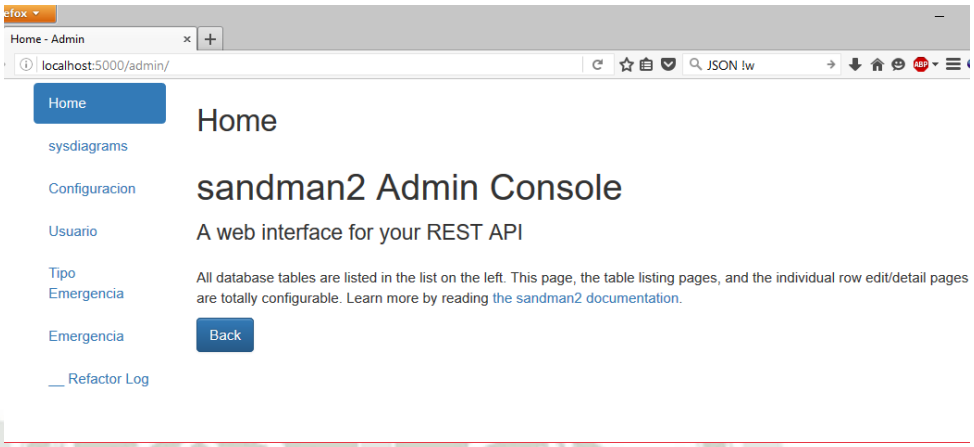

Ilustración 36: Interfaz gráfica de la API REST generada por sandman2

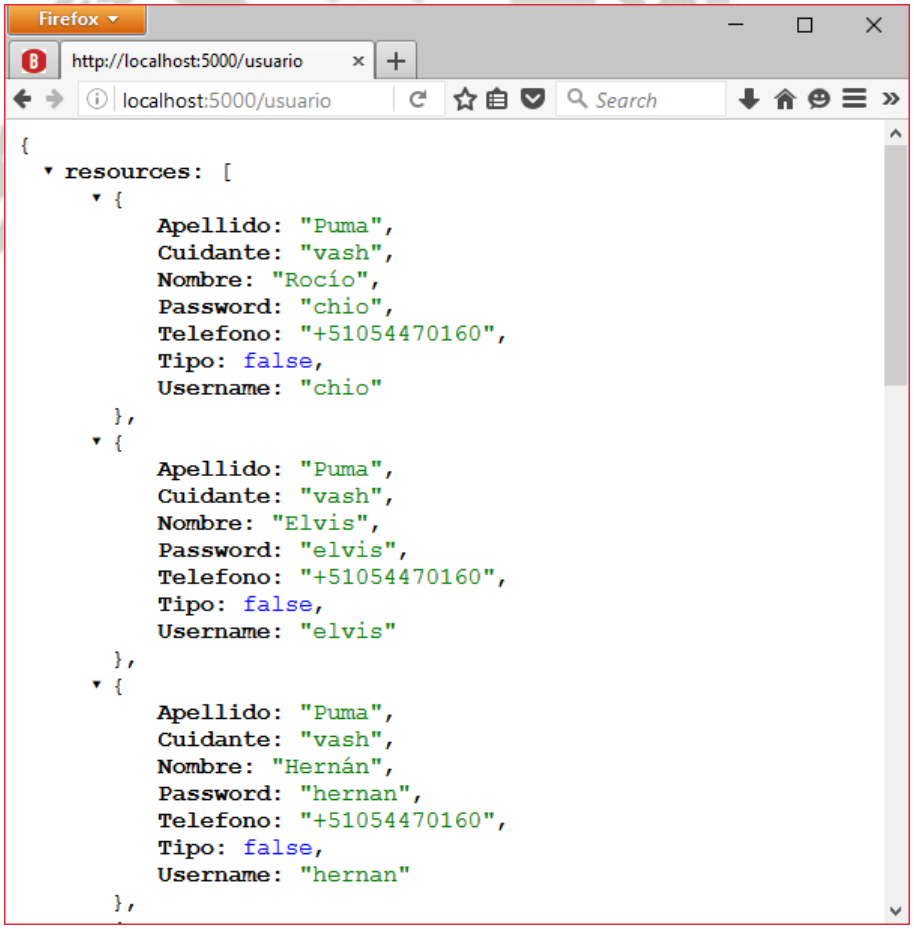

Ilustración 37: Tabla Usuario vista desde la API REST.

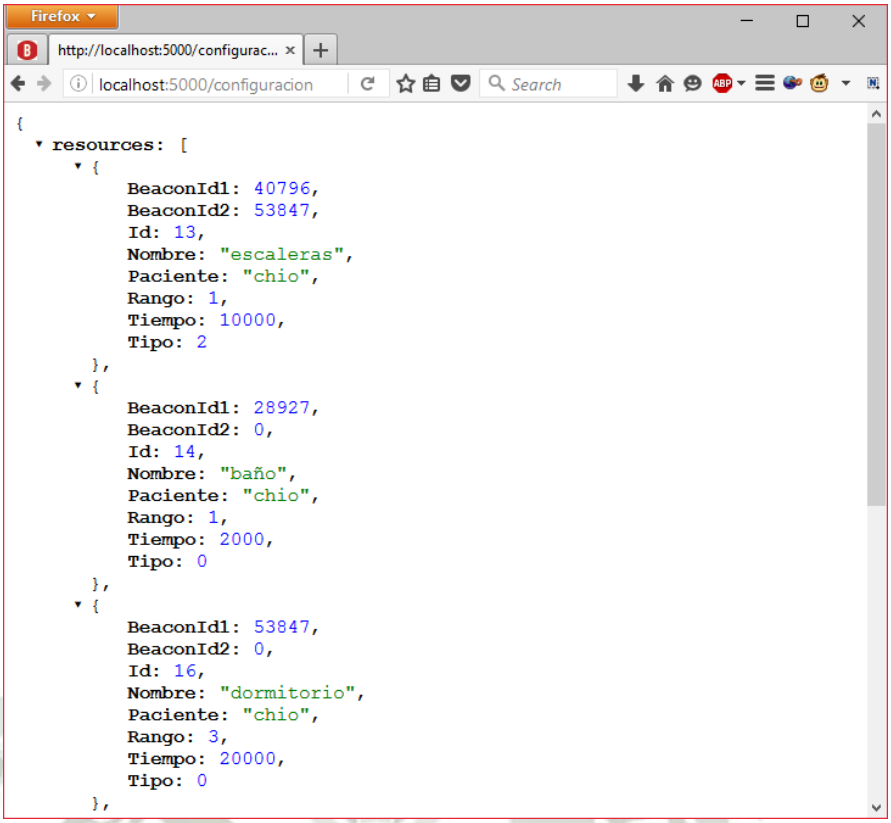

Ilustración 38: Tabla Configuracion vista a través de la API REST.

4.4.2. Comunicación de la aplicación con REST

Para la comunicación entre nuestra aplicación móvil y la API REST se creó una clase llamada **RESTService.cs**, la cual representa a nuestra capa de datos y al punto central por el cual cualquier módulo que quiera leer o escribir a la BD debe pasar. El procedimiento estánadar de comunicación consiste en **Serializar** y **Deserializar**. Serializar consiste en convertir un objeto representado en un formato binario, usualmente entendible solo por el lenguaje origen (en nuestro caso C#, a un formato interoperacional; entendible por otros lenguajes (en nuestro caso JSON). Deserializar es la operación inversa. En las **Ilustraciones 37 y 38** se muestran recortes de las funciones utilizadas en nuestro programa para leer y escribir usuarios, respectivamente. La acción de leer **GetUsers()** realiza una deserialización ya que nuestra información, obtenida a través de una acción HTTP **GET** dirigida a nuestra API REST, se encuentra en

formato JSON y queremos convertirla a una representación en memoria, más específicamente, a una lista de objetos del tipo **Usuario** para poder manipularla dentro de nuestro programa. En la acción de escribir hacemos lo inverso: Tenemos un objeto del tipo **Usuario**, el cual es serializado a formato JSON y enviado a través de una acción HTTP **POST** a nuestra API REST.

```
public async Task<List<Usuario>> GetUsers()
{
    List<Usuario> users = null;
    var uri =  "http://192.168.0.100:5000/usuario";
    try
    {
        var response = await client.GetAsync(uri);
        if (!response.IsSuccessStatusCode)
            return null;
        var jsonString = await response.Content.ReadAsStringAsync();
        var parsedJson = JObject.Parse(jsonString);
        var objResponse = parsedJson[propertyName];
        users = JsonConvert.DeserializeObject<List<Usuario>>(
                objResponse.ToString());
    }
    catch (Exception ex)
    {
        Debug.WriteLine(@"
         ERROR {0}", ex.Message);
    }
    return users;
}
```

Ilustración 39: Función de lectura de los Usuarios de la BD

```
public async Task<bool> SaveUser(Usuario usr)
{
    var uri =  "http://192.168.0.100:5000/usuario";
    try
    {
        var json = JsonConvert.SerializeObject(usr);
        var content = new StringContent(json, Encoding.UTF8,
                                        "application/json");
        HttpResponseMessage response;
        response = await client.PostAsync(uri, content);
        if(response.IsSuccessStatusCode)
            Debug.WriteLine(@"User saved");
        else
            Debug.WriteLine(response.StatusCode.ToString());
        return response.IsSuccessStatusCode;
    }
    catch(Exception ex)
    {
        Debug.WriteLine(@"
         ERROR {0}", ex.Message);
        return false;
    }
}
```

Ilustración 40: Función de Escritura de Nuevo Usuario

4.5. Beacons

Un Beacon es una minicomputadora que está constantemente transmitiendo información de distintos tipos, sin importar si alguien está escuchando o no. La tecnología estándar utilizada actualmente es Bluetooth Low Energy (BLE) (también conocida como Bluetooth Smart), que es la última iteración de Bluetooth, la cual ha sido diseñada para utilizar considerablemente menos recursos y mantener un rango de comunicación similar a las versiones anteriores (Bluetooth, s/f), además de que no necesita que ocurra un emparejamiento entre los dispositivos para poder comenzar a comunicarse.

Un Beacon, al igual que cualquier dispositivo inalámbrico, está sujeto a las limitaciones físicas que nos impone la tecnología. Dentro del Beacon existe una antena Bluetooth que se encarga de la transmisión; como cualquier antena, su precisión y rango depende en gran parte del posicionamiento de esta y de las interferencias que puede haber alrededor: un Beacon que se encuentre al lado de

un Router inalámbrico va a experimentar un rendimiento reducido debido a que ambos transmiten a la misma frecuencia: 2.4 GHz (además que el Router probablemente tiene más potencia). Asimismo, el posicionamiento del Beacon también es crucial, un Beacon que se encuentra posicionado cerca al suelo tiene su rango afectado negativamente comparado con uno que está posicionado a una altura elevada. Adicionalmente, la distancia reportada por un Beacon es una estimación que se basa en el procesamiento de la señal recibida, no es en definitiva un valor exacto, por lo tanto, también está sujeta a los factores mencionados arriba. Es por esta razón que utilizamos distancias relativas: cerca, lejos e inmediata, en vez de cantidades escalares. Estimote habla en detalle sobre la física y matemática involucrada en el funcionamiento de los Beacons (Estimote Inc., 2015).

Actualmente existen dos protocolos de beacons: el primero es **iBeacon**, diseñado por Apple en el 2013 (Apple Inc., 2016) y el segundo y más reciente es **Eddystone**, diseñado por Google y lanzado del 2015. Ambos protocolos son similares en funcionamiento; la gran diferencia siendo que Eddystone es un protocolo abierto, razón por la cual este último está desplazando poco a poco a iBeacon (Pulsate, 2016).

Independientemente del protocolo usado, el proceso de reconocimiento es el siguiente: cada Beacon tiene un intervalo regular de transmisión (por defecto un segundo), el dispositivo de destino (el teléfono en nuestro caso) empieza a escuchar la información de los beacons al mismo intervalo que ellos transmiten, generando un evento por cada intervalo. Entonces, para unos beacons configurados por defecto, el dispositivo escucha a cada segundo y en cada segundo puede procesar el evento que ha sido generado, el cual contiene una lista de los Beacons encontrados con sus **identificadores y la distancia calculada al dispositivo.** Existe información adicional transmitida que varía de dispositivo a dispositivo, como campos adicionales que puede almacenar información, temperatura, poder de transmisión, intervalo, etc.

El prototipo implementado actualmente solo soporta el protocolo iBeacon, sin embargo, añadir soporte para Eddystone es una tarea trivial gracias a

Xamarin.Forms y a la librería de detección que estamos usando. Los beacons usados para los pruebas son Sticker Beacons de la marca Estimote (Estimote Inc., 2016a), los cuales tienen un rango máximo de **setenta metros**, soportan ambos protocolos y tienen amplias opciones de configuración. Vale la pena recalcar que dado a que el programa utiliza protocolos de comunicación estándar para la detección, en teoría cualquier Beacon que tenga soporte para iBeacon o Eddystone funcionaría sin problemas.

4.5.1. Algoritmos

La detección de los beacons se encuentra en la clase llamada **BeaconManager.cs**. Esta es una clase estática que actúa de wrapper sobre la clase base de gestión de Beacons y es aquí donde se encuentran nuestros algoritmos de detección. La clase funciona en segundo plano y tiene dos métodos **Start()** y **Stop()** que inician y detienen la detección de beacons, respectivamente. Estos métodos son usados a discreción según el objeto que los necesite; en nuestro prototipo la detección se da en la pantalla principal del Paciente: **ConfigsView**. Cuando el paciente pasa a otra interfaz se detiene la detección y se resume cuando regresa a la anterior. El algoritmo de detección ocurre en cada intervalo que los Beacons transmiten información, de acuerdo a lo que se habló en la sección anterior, dentro de este intervalo se itera sobre los Beacons y se verifica si es que coinciden con las configuraciones del Usuario, si es así entonces se itera sobre cada configuración correspondiente para detectar emergencias. A continuación se muestra la implementación de los algoritmos especificados en la sección de **Diseño**, para poder contar el tiempo transcurrido de múltiples configuraciones se utilizó un diccionario de cronómetros (timers).

```
case EmergencyType.ProximityForPeriod:
   TargetBeacon = new Tuple<ushort, int>(econfig.BeaconId1, econfig.Rango);
   if (!beacons.Contain(TargetBeacon))
   {
       currentTimer.Reset();
       continue;
   }
   currentTimer.Start();
   if (currentTimer.ElapsedMilliseconds >= econfig.Tiempo)
   {
       EnableRanging = false;
       currentTimer.Stop();
       //mostramos una notificación
       Notifier.Inform(String.Format("detectada proximidad de {0} seg.",
                       currentTimer.ElapsedMilliseconds / 1000.0));
       currentTimer.Reset();
       await Parent.Navigation.PushAsync(
             new Views.PatientAlertView(User, econfig));
       EnableRanging = true;
   }
   break;
```

Ilustración 41: Implementación de algoritmo de Proximidad Por Periodo

```
case EmergencyType.ProximityForPeriodAtTime:
    TargetBeacon = new Tuple<ushort, int>(econfig.BeaconId1, econfig.Rango);
    if (!beacons.Contain(TargetBeacon))
    {
        currentTimer.Reset();
        continue;
    }
    currentTimer.Start();
    if (currentTimer.ElapsedMilliseconds >= econfig.Tiempo &&
        DateTime.Now.Hour >= econfig.Hora.Hour)
    {
        EnableRanging = false;
        currentTimer.Stop();
        //mostramos una notificación
        Notifier.Inform(String.Format("detectada proximidad de {0} seg.",
                        currentTimer.ElapsedMilliseconds / 1000.0));
        currentTimer.Reset();
        await Parent.Navigation.PushAsync(
              new Views.PatientAlertView(User, econfig));
        EnableRanging = true;
    }
    break;
```

Ilustración 42: Implementación de algoritmo de Proximidad Por Periodo A Hora

```
case EmergencyType.FastCross:
    StartBeacon = new Tuple<ushort, int>(econfig.BeaconId1, econfig.Rango);
    EndBeacon = new Tuple<ushort, int>(econfig.BeaconId2, econfig.Rango);

    //comienzo del cruce
    if (CROSS == "NO" && beacons.Contain(StartBeacon))
    {
        currentTimer.Restart();
        CROSS = "BEGAN";
        Notifier.Inform("COMIENZO CRUCE");
        return;
    }
    if (CROSS == "BEGAN" && beacons.Contain(EndBeacon))
    {
        //cruce satisfactorio (emergencia
        if (currentTimer.Elapsed.TotalSeconds > econfig.Tiempo / 1000.0)
        {
            //se pasó del tiempo límite para completar el cruce
            CROSS = "NO";
            continue;
        }
        currentTimer.Stop();
        Notifier.Inform("CRUCE DETECTADO");
        CROSS = "NO";
        await Parent.Navigation.PushAsync(
              new Views.PatientAlertView(User, econfig));
        EnableRanging = true;
    }
    break;
```

Ilustración 43: Implementación de algoritmo de Cruce Rápido

```
case EmergencyType.IncompleteCross:
    StartBeacon = new Tuple<ushort, int>(econfig.BeaconId1, econfig.Rango);
    EndBeacon = new Tuple<ushort, int>(econfig.BeaconId2, econfig.Rango);

    //comienzo del cruce
    if (CROSS == "NO" && beacons.Contain(StartBeacon))
    {
        currentTimer.Restart();
        CROSS = "BEGAN";
        Notifier.Inform("COMIENZO CRUCE");
        return;
    }
    if (CROSS == "BEGAN" && !beacons.Contain(EndBeacon))
    {
        if (currentTimer.Elapsed.TotalSeconds <= econfig.Tiempo / 1000.0)
            continue;
        currentTimer.Stop();
        Notifier.Inform(String.Format("CRUCE INCOMPLETO en {0}
seg", currentTimer.Elapsed.TotalSeconds));
        CROSS = "NO";
        await Parent.Navigation.PushAsync(
              new Views.PatientAlertView(User, econfig));
        EnableRanging = true;
    }
    break;
```

Ilustración 44: Implementación de algoritmo de Cruce Incompleto

4.6. Librerías utilizadas

- **Estimotes.Xplat**

  Permite el desarrollo multiplataforma usando el SDK de Estimote (aritchie, 2016).

- **Humanizer**

  Muestra fechas y horas en un formato más amigable para los humanos, e.g. “hace dos horas”, “hace unos momentos, etc. (“Humanizr/Humanizer”, s/f).

- **Toast.Forms.Plugin**

  Permite mostrar notificaciones al estilo Toast en Android y iOS (“EgorBo/Toasts.Forms.Plugin: A plugin for Xamarin and Windows - it unites Crouton (Android), TWMessageBarManager (iOS) and my toast notificator for WP8.”, s/f).

- **Newtonsoft.JSON**

  Framework robusto de manipulación de JSON para .NET (“Json.NET - Newtonsoft”, s/f).

- **VibratePlugin**

  Librería multiplataforma para acceder a las capacidades de vibración del teléfono ("jamesmontemagno/VibratePlugin", s/f).

4.7. Repositorio de código fuente

El código fuente del prototipo desarrollado está disponible, en su totalidad, en el sitio de almacenamiento de proyectos de código abierto Github. Adicionalmente, todo el proceso de desarrollo se encuentra documentado y puede ser revisado a discreción. El código fuente ha sido liberado bajo la licencia Apache 2.0 ("Licenses", s/f). La URL es la siguiente:

https://github.com/vash47/care-app

4.8. Factibilidad Económica

Para el presupuesto requerido por la implementación del sistema se ideó una casa "promedio", que consiste de:

- **Dos pisos**, donde habría **una escalera**, la cual requeriría **dos beacons**. Lamentablemente no existen estadísticas sobre el número de niveles o "pisos" promedio dentro de los censos poblacionales; solo es un estimado.
- **Dos baños** por casa, un beacon por cada uno.
- **Un dormitorio**, donde descansa el paciente

Adicionalmente, se consideró el costo de un smartphone promedio, aunque se podría usar uno existente si cumple los requisitos.

En la siguiente tabla se puede apreciar que el costo total requerido para desplegar el sistema en el hogar promedio peruano sería de **S/. 413.50**. Los precios son actuales a noviembre del 2016.

item	precio (S/.)	cantidad	subtotal (S/.)
beacon	40.9[1]	5	204.5
smartphone	209.0[2]	1	209.0
		total	**413.5**

Tabla 2 - Presupuesto del sistema

[1] http://www.ebay.com/itm/2sets-3-5years-Bluetooth4-0-BLE-Beacon-with-iBeacon-Eddystone-Tech-/262086661172?hash=item3d05951434:g:yQEAAOSwsB9V9prW

[2] http://www.entel.pe/producto/oferta-huawei-y360/

CAPÍTULO 5 - PRUEBAS Y VALIDACIÓN

El objetivo de las pruebas fue determinar la precisión de detección de emergencias del prototipo. Se creó un nuevo usuario y se le asignaron cuatro configuraciones de emergencia. Para esto se realizaron veinte pruebas (observaciones) por cada configuración. Dentro de estas veinte, las pruebas han sido divididas equitativamente según la posición del teléfono relativa al cuerpo: en la mano y en el bolsillo (en la repisa para los algoritmos de **Proximidad**); como vimos en la sección **Beacons**, la naturaleza de transmisión de las redes inalámbricas, más específicamente, Bluetooth hace a estas bastante sensibles al posicionamiento de la antena (en este caso, la antena del teléfono). Es importante indicar los detalles de las posiciones; cuando hablamos del teléfono en la mano nos referimos a la posición usual en la que una persona interactúa con su teléfono: con la pantalla orientada al rostro a una posición elevada, cuando hablamos del bolsillo nos referimos al bolsillo del pantalón, en posición vertical y cuando hablamos de repisa hablamos del teléfono recostado horizontalmente a una altura de aproximadamente un metro.

Cada observación cuenta con los siguientes datos: tipo de algoritmo o detección, posición relativa al cuerpo y los datos relevantes de la configuración de emergencia (tiempo, distancia y hora) y un valor binario que indica el criterio de éxito, donde 0 indica que el sistema falló en detectar la emergencia y 1, que la emergencia se detectó cuando se esperaba.

Para los algoritmos de proximidad se utilizó un beacon que fue pegado en la pared central de la habitación de prueba (en este caso un baño), a una **altura de 160 cm**, cada observación fue realizada entrando a la habitación durante el tiempo programado por la configuración (en el caso de nuestras pruebas, **cinco segundos**) y saliendo de ésta después de dicho tiempo, tomando nota si hubo la detección correspondiente.

Para los algoritmos de cruce se utilizaron dos beacons, uno al comienzo de las escaleras y otro al final, con el objetivo de simular a un paciente que se encuentra bajando las gradas. Ambos beacons fueron posicionados también en la pared a una **altura de 160cm**, con una duración de **quince segundos** y **diez segundos** para el cruce rápido e

incompleto, respectivamente. En el caso del cruce rápido se bajaron las gradas rápidamente, simulando a alguien que se ha tropezado y en el caso del cruce incompleto se interrumpió el cruce en la mitad de las gradas por un tiempo mayor al configurado. El sujeto de pruebas fue un joven de veintidós años, debido al problema a la salud que plantearía el hecho de simular caídas con personas mayores.

El dispositivo de prueba fue un teléfono HUAWEI G6-L22 corriendo Android 4.3, con soporte para BLE.

Los resultados de las pruebas se validaron mediante una rutina adicional que se le programó al módulo de detección la cual indica el tipo de emergencia detectada y los beacons involucrados, más el tiempo de detección (si aplica). El detalle y resultados de las pruebas pueden observarse en la **Ilustración 33 y 34,** respectivamente.

n	Algoritmo	Posición	Tiempo (seg.)	Distancia	Detección
1	Cruce Rápido	bolsillo	15	Cerca	0
2	Cruce Rápido	bolsillo	15	Cerca	1
3	Cruce Rápido	bolsillo	15	Cerca	1
4	Cruce Rápido	bolsillo	15	Cerca	1
5	Cruce Rápido	bolsillo	15	Cerca	0
6	Cruce Rápido	bolsillo	15	Cerca	1
7	Cruce Rápido	bolsillo	15	Cerca	0
8	Cruce Rápido	bolsillo	15	Cerca	0
9	Cruce Rápido	bolsillo	15	Cerca	0
10	Cruce Rápido	bolsillo	15	Cerca	1
1	Cruce Rápido	mano	15	Cerca	1
2	Cruce Rápido	mano	15	Cerca	0
3	Cruce Rápido	mano	15	Cerca	1
4	Cruce Rápido	mano	15	Cerca	1
5	Cruce Rápido	mano	15	Cerca	1
6	Cruce Rápido	mano	15	Cerca	1
7	Cruce Rápido	mano	15	Cerca	0
8	Cruce Rápido	mano	15	Cerca	1
9	Cruce Rápido	mano	15	Cerca	1
10	Cruce Rápido	mano	15	Cerca	0
1	Cruce Incompleto	bolsillo	10	Cerca	0
2	Cruce Incompleto	bolsillo	10	Cerca	0
3	Cruce Incompleto	bolsillo	10	Cerca	1
4	Cruce Incompleto	bolsillo	10	Cerca	1
5	Cruce Incompleto	bolsillo	10	Cerca	0
6	Cruce Incompleto	bolsillo	10	Cerca	1
7	Cruce Incompleto	bolsillo	10	Cerca	1
8	Cruce Incompleto	bolsillo	10	Cerca	1
9	Cruce Incompleto	bolsillo	10	Cerca	0
10	Cruce Incompleto	bolsillo	10	Cerca	1
1	Cruce Incompleto	mano	10	Cerca	1

2	Cruce Incompleto	mano	10	Cerca	1
3	Cruce Incompleto	mano	10	Cerca	1
4	Cruce Incompleto	mano	10	Cerca	0
5	Cruce Incompleto	mano	10	Cerca	0
6	Cruce Incompleto	mano	10	Cerca	1
7	Cruce Incompleto	mano	10	Cerca	1
8	Cruce Incompleto	mano	10	Cerca	0
9	Cruce Incompleto	mano	10	Cerca	1
10	Cruce Incompleto	mano	10	Cerca	1
1	Proximidad Por Periodo	repisa	5	Cerca	1
2	Proximidad Por Periodo	repisa	5	Cerca	1
3	Proximidad Por Periodo	repisa	5	Cerca	0
4	Proximidad Por Periodo	repisa	5	Cerca	1
5	Proximidad Por Periodo	repisa	5	Cerca	0
6	Proximidad Por Periodo	repisa	5	Cerca	0
7	Proximidad Por Periodo	repisa	5	Cerca	0
8	Proximidad Por Periodo	repisa	5	Cerca	1
9	Proximidad Por Periodo	repisa	5	Cerca	1
10	Proximidad Por Periodo	repisa	5	Cerca	1
1	Proximidad Por Periodo	mano	5	Cerca	1
2	Proximidad Por Periodo	mano	5	Cerca	0
3	Proximidad Por Periodo	mano	5	Cerca	1
4	Proximidad Por Periodo	mano	5	Cerca	1
5	Proximidad Por Periodo	mano	5	Cerca	0
6	Proximidad Por Periodo	mano	5	Cerca	1
7	Proximidad Por Periodo	mano	5	Cerca	1
8	Proximidad Por Periodo	mano	5	Cerca	1
9	Proximidad Por Periodo	mano	5	Cerca	1
10	Proximidad Por Periodo	mano	5	Cerca	1
1	Proximidad Por Periodo y Hora	repisa	5	Cerca	0
2	Proximidad Por Periodo y Hora	repisa	5	Cerca	0
3	Proximidad Por Periodo y Hora	repisa	5	Cerca	1
4	Proximidad Por Periodo y Hora	repisa	5	Cerca	0

5	Proximidad Por Periodo y Hora	repisa	5	Cerca	1
6	Proximidad Por Periodo y Hora	repisa	5	Cerca	0
7	Proximidad Por Periodo y Hora	repisa	5	Cerca	1
8	Proximidad Por Periodo y Hora	repisa	5	Cerca	1
9	Proximidad Por Periodo y Hora	repisa	5	Cerca	1
10	Proximidad Por Periodo y Hora	repisa	5	Cerca	0
1	Proximidad Por Periodo y Hora	mano	5	Cerca	1
2	Proximidad Por Periodo y Hora	mano	5	Cerca	1
3	Proximidad Por Periodo y Hora	mano	5	Cerca	0
4	Proximidad Por Periodo y Hora	mano	5	Cerca	0
5	Proximidad Por Periodo y Hora	mano	5	Cerca	1
6	Proximidad Por Periodo y Hora	mano	5	Cerca	1
7	Proximidad Por Periodo y Hora	mano	5	Cerca	1
8	Proximidad Por Periodo y Hora	mano	5	Cerca	1
9	Proximidad Por Periodo y Hora	mano	5	Cerca	1
10	Proximidad Por Periodo y Hora	mano	5	Cerca	1

Tabla 3: Tabla de pruebas. Para el caso del algoritmo de Proximidad Por Periodo a Hora se utilizó la hora a la que se realizaron las pruebas.

Algoritmo	**Posición**	**Precisión**
Cruce Rápido	bolsillo	0.5
Cruce Rápido	mano	0.7
Cruce Incompleto	bolsillo	0.6
Cruce Incompleto	mano	0.7
Proximidad	repisa	0.6
Proximidad	mano	0.8
Proximidad a Hora	repisa	0.5
Proximidad a Hora	mano	0.8

Tabla 4: Resultados de las pruebas. La precisión está expresada en escala del 0 al 1, donde 1 vendría a ser una precisión del 100%.

5.1. Pruebas de aceptación

En adición a las pruebas de precisión, también se llevaron a cabo pruebas de aceptación, las cuales consistieron de una encuesta de ocho preguntas acerca de la experiencia del usuario con el sistema. Se realizó dicha encuesta a **cuatro sujetos** mayores de sesenta años: tres mujeres y un varón, todos profesionales y jubilados. Para cada sujeto se le creó un usuario y una configuración de emergencia de prueba (proximidad de cinco segundos). Los detalles de la encuesta se encuentran en el **ANEXO B**. A cada sujeto se le proporcionó un teléfono con el aplicativo desplegado y con la sesión iniciada para su respectivo usuario, el razonamiento detrás de esto es que las tareas de creación, configuración e inicio de sesión serían llevadas a cabo por el **Cuidante**; al **Paciente** solo se le proporcionaría el teléfono una vez que esté todo listo y desde ahí solo interactuaría con él cuando se detecta una emergencia. La simulación de la emergencia se dio bajo el siguiente procedimiento: el sujeto entró al baño (con el beacon posicionado a una altura de 160cm en el centro de la habitación) y permaneció por más de cinco segundos, para luego salir, repitiendo el procedimiento cinco veces. Después de la interacción con el sistema se procedió a hacer la encuesta y recolectar las respuestas.

DISCUSIÓN DE LOS RESULTADOS

Como podemos observar en la **Ilustración 34** cuando se trata de hacer la detección con el teléfono en el bolsillo o en la repisa el grado de precisión es casi aleatorio (entre 50% y 60%), sin embargo, cuando el teléfono se tiene en la mano la precisión aumenta considerablemente, llegando a un 70% para los algoritmos de cruce y hasta un 80% para los algoritmos de proximidad. Esto tiene sentido si consideramos las limitaciones tecnológicas de la comunicación inalámbrica: cuando el teléfono está en una posición vertical y apuntando hacia arriba la antena cuenta con su mayor rango de comunicación (Estimote Inc., 2015), la elevación física también incrementa su capacidad de alcance; el sostener el teléfono en la mano cómo se describió en la sección anterior cuenta con todas estas características, mientras tenerlo en el bolsillo o yaciendo en una superficie horizontalmente tienen a la antena del teléfono en posiciones no óptimas.

También podemos observar que los algoritmos de Proximidad tienen una precisión más alta (80%) comparada con la de los algoritmos de Cruce. Esto se debe al hecho de que los algoritmos de proximidad son los más simples: solo involucran un Beacon, mientras que los algoritmos de Cruce involucran dos; por el concepto de entropía es de esperar de que algo que emplea más cosas tenga más probabilidades de fallar que algo que menos.

Sobre la aceptación del dispositivo podemos ver que fue positiva para todos los sujetos: todos están de acuerdo en que la interfaz es legible y entendible, resaltando el tamaño de los botones y del texto como una ventaja. A su vez, todos concuerdan en que la opción de que el sistema hable al usuario y vibre el teléfono cuando detecta una emergencia es útil. Solo una persona no estaría dispuesta a usar su teléfono colgando del cuello debido a que, en sus palabras: “mi celular es grande y pesa”. Similarmente, solo una persona recalcó que no estaría muy dispuesto a usar un smartwatch porque escuchó que dichos aparatos “son caros”. Todos recomendarían el sistema a sus amigos y/o familiares, mientras que también concuerdan en que el precio presupuestado es factible, considerando que, aparte de no ser muy elevado, por ser relevante a su salud valdría la inversión.

CONCLUSIONES

1. El prototipo del sistema diseñado fue implementado exitosamente, siendo el único requerimiento, aparte de los beacons, tener un smartphone con una versión de Android 4.3 o superior; básicamente la mayoría de smartphones Android vendidos desde el 2014 (Sarah Silvert, 2013). El costo de instalación del sistema requerido para un hogar promedio está por debajo de los quinientos nuevos soles, incluyendo un smartphone, equivaliendo a menos de la mitad del sueldo mínimo peruano al 2016 (TuSalario.org, 2016). Adicionalmente, según la encuesta realizada a los usuarios finales, todos concuerdan en que es un precio asequible.

2. El tiempo de respuesta del sistema está limitado solo por la conexión a internet de ambos Paciente y Cuidador, sin embargo, gracias a la compacidad de la información transmitida: máximo 2KB, ya que el prototipo utiliza cabeceras HTTP (Google Inc., 2008), dicho tiempo de respuesta no presentaría problema alguno en conexiones lentas, ni tampoco resultaría en consumo significante de ancho de banda.

3. La arquitectura del sistema y de la información fue definida en torno al lenguaje de modelado UML (OMG, 2016), mientras que otros aspectos del diseño (como la interfaz de usuario) fueron definidos teniendo en base a los principios de creación de interfaces humano-computador (Sutcliffe, 1988) y las pautas de diseño para Android (Google Inc., s/f)

4. Se diseñaron cuatro algoritmos pensando en las situaciones más comunes en las que se podrían encontrar los pacientes y de una manera en la que se minimizó, en lo posible, el uso de los recursos del sistema.

5. El prototipo implementado fue desarrollado utilizando una metodología ágil, con iteraciones cortas y teniendo en cuenta las buenas prácticas de programación, en especial en lo referente al lenguaje de programación C# (Microsoft Inc., 2016).

6. Se realizó una serie de pruebas sobre la precisión en la detección de emergencias, cuyos detalles se pueden ver en la sección **Pruebas y Validación**

7. El posicionamiento nos presenta un problema en términos de uso práctico; obviamente no podemos hacer que los pacientes estén asiendo su teléfono constantemente, eso derrotaría la usabilidad del sistema y terminaría con el mismo destino de muchos dispositivos técnicamente hábiles pero imprácticos: el rechazo por parte de los usuarios (Mubashir et al., 2013). Una solución simple sería atar el teléfono a una especie de hilo y tenerlo como collar, debido a la propensión de las personas mayores a olvidar cosas no sería difícil de implementar y tampoco sería una sorpresa si ya hubiera personas cargando su teléfono de esa manera. Los resultados de las pruebas de aceptación apoyan esta posición ya que solo una persona no está muy dispuesta a utilizar el teléfono de esta manera, por razones de comodidad (el peso del dispositivo).

8. Aunque los grados de precisión de 70% y 80% son considerables bajo los estándares de sistemas de clasificación y predicción (Sandro Saitta, 2010), como tratamos con el objetivo de detectar adecuadamente emergencias que podrían significar la vida o la muerte para nuestros pacientes, es necesario seguir investigando maneras de incrementar esta precisión, ya sea ayudándonos de otras tecnologías a disposición en un smartphone (acelerómetro, GPS) o afinando los algoritmos en la medida posible.

9. El prototipo implementado utiliza un método de comunicación con los beacons llamado Ranging. Dicho método actualmente solo funciona en primer plano en iOS (a diferencia de Android que puede seguir comunicándose aun cuando la pantalla se ha apagado) esto limitaría prácticamente nuestro sistema a dispositivos Android. Desafortunadamente no existe información sobre si es que Apple planea cambiar esto en futuras versiones. Una alternativa sería utilizar el otro método de comunicación llamado Monitoreo: este funciona en primer y segundo plano en ambos sistemas operativos y consume menos batería, sin embargo, la información que nos proporciona no es distancia al beacon (ya sea absoluta o relativa) sino regiones; nos puede indicar cuando un usuario ha entrado o salido de una región (definida por uno o más beacons). Esto requería un cambio significativo de los algoritmos definidos en este trabajo.

10. Todavía existe mucho qué hacer en el campo de detección de caídas; es remarcable como reconocer una caída nos es una tarea tan trivial como humanos, pero hasta ahora nos sigue presentando grandes retos cuando se trata de programar máquinas que lo hagan automáticamente.

RECOMENDACIONES Y TRABAJOS FUTUROS

1. Debido a la escasez de recursos y de tiempo solo se probó la aplicación en un teléfono de gama media. Sería interesante probar distintos dispositivos, podría darse el caso de que haya unos con transmisores Bluetooth mucho más precisos y que esto influya significativamente los resultados. Adicionalmente, aunque el sistema actual ha sido diseñado para correr en la mayor cantidad de dispositivos posibles, un método de trabajo alternativo sería utilizar los principios de detección de beacons ilustrados aquí pero con un dispositivo dedicado, la ventaja sería que al concentrar todos los esfuerzos en un solo hardware se podría afinar el dispositivo de manera que aproveche sus capacidades al máximo.

2. Una tecnología prometedora es Estimote Indoor Location (Estimote Inc., 2016b), la cual permite hacer un mapeo espacial completo de una o más habitaciones, el cual nos permitiría saber exactamente donde se encuentra el usuario. La desventaja es que requiere un mínimo de cuatro beacons para una habitación normal (cuatro paredes) o más dependiendo de los requerimientos. Adicionalmente esta tecnología solo está disponible a los beacons de marca Estimote, a diferencia de nuestro sistema que puede comunicarse con cualquier beacon que soporte el protocolo iBeacon o Eddystone.

3. En cuanto a las opciones que presenta el programa hay bastantes funcionalidades que se puede añadir: la más inmediata es habilitar el inicio de sesión con huella digital para los dispositivos que lo soportan. Otra opción de utilidad sería permitir al Cuidador monitorear el estado de la batería del paciente, su conectividad a internet y su geolocalización. En el caso de la batería sería de gran utilidad notificarle al paciente si es que se le está agotando para que vaya a recargar su teléfono.

4. Con la llegada de los smartwatches valdría la pena probar el sistema en uno de estos dispositivos, ya que han sido desarrollados con el Internet de las Cosas (IoT) en mente deben de contar con sistemas Bluetooth bastante robustos, los cuales podrían prestarse mejor a los tipos de algoritmos que hemos desarrollado. Además, Xamarin.Forms también está disponible en estas plataformas, haciendo la migración de nuestro prototipo mucho más fácil.

BIBLIOGRAFÍA

ACM. (2009). ACM SIGCHI Curricula for Human-Computer Interaction : 2. Definition and Overview of Human-Computer Interaction. Recuperado el 8 de abril de 2016, a partir de http://old.sigchi.org/cdg/cdg2.html

Adibi, S. (Ed.). (2015). *Mobile Health* (Vol. 5). Cham: Springer International Publishing. Recuperado a partir de http://link.springer.com/10.1007/978-3-319-12817-7

Apple Inc. (2016). About iBeacon on your iPhone, iPad, and iPod touch. Recuperado el 17 de julio de 2016, a partir de https://support.apple.com/en-gb/HT202880

aritchie. (2016). ACR Estimotes Plugin For Xamarin. Recuperado el 17 de julio de 2016, a partir de https://github.com/aritchie/estimotes-xplat

Bardales, E. (2015, mayo 20). El déficit hospitalario del Perú equivale a 1.5 camas por cada 1,000 habitantes. Recuperado el 7 de abril de 2016, a partir de http://gestion.pe/mercados/oferta-hospitalaria-peru-solo-15-camas-cada-1000-habitantes-2132300

Below, P., Hassan, Q. F., & Stark, G. (2011). Demystifying Cloud Computing. *CrossTalk*. Recuperado a partir de http://www.crosstalkonline.org/storage/issue-archives/2011/201101/201101-0-Issue.pdf

Blaya, J. A., Fraser, H. S. F., & Holt, B. (2010). E-Health Technologies Show Promise In Developing Countries. *Health Affairs*, *29*(2), 244–251. https://doi.org/10.1377/hlthaff.2009.0894

Bluetooth. (s/f). Bluetooth Low Energy | Bluetooth Technology Website. Recuperado el 8 de abril de 2016, a partir de https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

Bourke, A. K., Van de Ven, P. W., Chaya, A. E., OLaighin, G. M., & Nelson, J. (2008). Testing of a long-term fall detection system incorporated into a custom vest for the elderly. En *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE* (pp. 2844–2847). IEEE. Recuperado a partir de http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4649795

Della Mea, V. (2001). What is e-Health (2): The death of telemedicine? *Journal of Medical Internet Research*, *3*(2), e22. https://doi.org/10.2196/jmir.3.2.e22

Demiris, G., Rantz, M. J., Aud, M. A., Marek, K. D., Tyrer, H. W., Skubic, M., & Hussam, A. A. (2004). Older adults' attitudes towards and perceptions of "smart home" technologies: a pilot study. *Medical Informatics and the Internet in Medicine*, *29*(2), 87–94. https://doi.org/10.1080/14639230410001684387

ECMA International. (2016). JSON. Recuperado el 17 de julio de 2016, a partir de http://www.json.org/

EgorBo/Toasts.Forms.Plugin: A plugin for Xamarin and Windows - it unites Crouton (Android), TWMessageBarManager (iOS) and my toast notificator for WP8. (s/f). Recuperado el 17 de julio de 2016, a partir de https://github.com/EgorBo/Toasts.Forms.Plugin

Ericsson. (2016, febrero). Ericsson Mobility Report MWC Edition. Recuperado el 8 de abril de 2016, a partir de http://www.ericsson.com/res/docs/2016/mobility-report/ericsson-mobility-report-feb-2016-interim.pdf

Estimote Inc. (2015). How do beacons work? The physics of beacon tech. Recuperado el 6 de mayo de 2016, a partir de http://blog.estimote.com/post/106913675010/how-do-beacons-work-the-physics-of-beacon-tech

Estimote Inc. (2016a). Estimote Beacons — real world context for your apps. Recuperado el 7 de abril de 2016, a partir de http://estimote.com/

Estimote Inc. (2016b). Estimote Indoor Location. Recuperado el 17 de julio de 2016, a partir de http://estimote.com/indoor/

Futuro Labs. (2015). Índice del usuario móvil 2015/Q1 - Infogram, charts & infographics. Recuperado el 8 de abril de 2016, a partir de https://infogr.am/indice_del_usuario_movil_2015q1

Google Inc. (2008). SPDY: An experimental protocol for a faster web - The Chromium Projects. Recuperado el 6 de septiembre de 2016, a partir de http://dev.chromium.org/spdy/spdy-whitepaper

Google Inc. (s/f). Design | Android Developers. Recuperado el 6 de septiembre de 2016, a partir de https://developer.android.com/design/index.html

Humanizr/Humanizer. (s/f). Recuperado el 17 de julio de 2016, a partir de https://github.com/Humanizr/Humanizer

Inei. (2015). *Estado de la Población Peruana 2015*. Recuperado a partir de https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1251/Libro.pdf

Ipsos. (2015). *Aplicaciones Móviles - Perú Urbano*. Recuperado a partir de http://www.ipsos.pe/sites/default/files/marketing_data/Apps.pdf

jamesmontemagno/VibratePlugin. (s/f). Recuperado el 17 de julio de 2016, a partir de https://github.com/jamesmontemagno/VibratePlugin

jeffknupp/sandman2. (s/f). Recuperado el 17 de julio de 2016, a partir de https://github.com/jeffknupp/sandman2

Json.NET - Newtonsoft. (s/f). Recuperado el 17 de julio de 2016, a partir de http://www.newtonsoft.com/json

Kangas, M., Konttila, A., Lindgren, P., Winblad, I., & Jämsä, T. (2008). Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait & Posture*, *28*(2), 285–291. https://doi.org/10.1016/j.gaitpost.2008.01.003

Kaplan, W. A. (2006). Can the ubiquitous power of mobile phones be used to improve health outcomes in developing countries? *Globalization and Health*, *2*, 9. https://doi.org/10.1186/1744-8603-2-9

Kieser, H. (2014, mayo 29). What is a Mock-Up? Recuperado el 8 de abril de 2016, a partir de https://experience.sap.com/basics/what-is-a-mock-up/

Klasnja, P., & Pratt, W. (2012). Healthcare in the pocket: Mapping the space of mobile-phone health interventions. *Journal of Biomedical Informatics*, *45*(1), 184–198. https://doi.org/10.1016/j.jbi.2011.08.017

Korhonen, I., & Bardram, J. E. (2004). Guest Editorial Introduction to the Special Section on Pervasive Healthcare. *IEEE Transactions on Information Technology in Biomedicine*, *8*(3), 229–234. https://doi.org/10.1109/TITB.2004.835337

Leavitt, N. (2010). Will NoSQL databases live up to their promise? *Computer*, *43*(2), 12–14.

Licenses. (s/f). Recuperado el 18 de julio de 2016, a partir de http://www.apache.org/licenses/

Lindquist, A. M., Johansson, P. E., Petersson, G. I., Saveman, B.-I., & Nilsson, G. C. (2008). The Use of the Personal Digital Assistant (PDA) Among Personnel and Students in Health Care: A Review. *Journal of Medical Internet Research*, *10*(4), e31. https://doi.org/10.2196/jmir.1038

Mann, S. (1996). Smart Clothing: The Shift to Wearable Computing. *Commun. ACM*, *39*(8), 23–24. https://doi.org/10.1145/232014.232021

Microsoft Inc. (2016). C# Coding Conventions (C# Programming Guide). Recuperado el 6 de septiembre de 2016, a partir de https://msdn.microsoft.com/en-us/library/ff926074.aspx

Mohan, C. (2013). History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla. En *Proceedings of the 16th International Conference on Extending Database Technology* (pp. 11–16). ACM. Recuperado a partir de http://dl.acm.org/citation.cfm?id=2452378

Mubashir, M., Shao, L., & Seed, L. (2013). A survey on fall detection: Principles and approaches. *Neurocomputing*, *100*, 144–152. https://doi.org/10.1016/j.neucom.2011.09.037

Olivera, J., & Clausen, J. (2014). Las características del adulto mayor peruano y las políticas de protección social. *Economía*, *37*(73), 75–112.

OMEGA. (s/f). Accelerometer. Recuperado el 7 de abril de 2016, a partir de http://www.omega.com/prodinfo/accelerometers.html

OMG. (2016). Welcome To UML Web Site! Recuperado el 9 de julio de 2016, a partir de http://www.uml.org/

OMS. (2012). OMS | Caídas. Recuperado el 7 de abril de 2016, a partir de http://www.who.int/mediacentre/factsheets/fs344/es/

Perú 21. (2014, febrero 23). Essalud: Crisis en hospitales afecta a millones de peruanos. Recuperado el 7 de abril de 2016, a partir de http://peru21.pe/actualidad/essalud-crisis-hospitales-afecta-millones-peruanos-2171321

Pointr. (s/f). Beacons: Everything you need to know. | Pointr Blog. Recuperado a partir de http://www.pointrlabs.com/blog/beacons-everything-you-need-to-know/

Pulsate. (2016). Eddystone vs iBeacon: The Big Beacon Battle. Recuperado el 6 de mayo de 2016, a partir de http://academy.pulsatehq.com/eddystone-vs-ibeacon-the-big-beacon-battle

RESTful Web services: The basics. (2015, febrero 9). Recuperado el 17 de julio de 2016, a partir de http://www.ibm.com/developerworks/library/ws-restful/index.html

Sandro Saitta. (2010, noviembre 4). What is a good classification accuracy in data mining? | Data Mining Research - www.dataminingblog.com. Recuperado a partir de http://www.dataminingblog.com/what-is-a-good-classification-accuracy-in-data-mining/, http://www.dataminingblog.com/what-is-a-good-classification-accuracy-in-data-mining/

Sarah Silvert. (2013, julio 24). Android 4.3 Jelly Bean official: shipping with new Nexus 7, available OTA for select devices today. Recuperado el 6 de septiembre de 2016, a partir de https://www.engadget.com/2013/07/24/android-4-3-jelly-bean-official/

Schmidt, A. (s/f). Context-Aware Computing. Recuperado el 8 de abril de 2016, a partir de https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/context-aware-computing-context-awareness-context-aware-user-interfaces-and-implicit-interaction

Sutcliffe, A. G. (1988). *Human-computer interface design*. Springer.

terminology - What is the difference between wireframes and mockups? - Graphic Design Stack Exchange. (s/f). Recuperado el 8 de abril de 2016, a partir de http://graphicdesign.stackexchange.com/questions/30860/what-is-the-difference-between-wireframes-and-mockups

TuSalario.org. (2016, abril 29). Salario Minimo, Remuneración Mínima en Perú. Recuperado el 15 de noviembre de 2016, a partir de http://www.tusalario.org/peru/portada/salario/remuneracion-minima

Weiser, M. (1991). The computer for the 21st century. *Scientific american*, *265*(3), 94–104.

WhatIs.com. (s/f). What is sensor? - Definition from WhatIs.com. Recuperado el 8 de abril de 2016, a partir de http://whatis.techtarget.com/definition/sensor

Xamarin Inc. (2016). Mobile Application Development to Build Apps in C# - Xamarin. Recuperado el 14 de julio de 2016, a partir de https://www.xamarin.com/platform

ANEXO A – GLOSARIO DE TÉRMINOS

- **Algoritmo**

  Conjunto de instrucciones que realizan alguna tarea específica
- **CRUD**

  Las operaciones básicas en una Base de Datos: Crear, Leer, Actualizar y Borrar (**C**reate, **R**ead, **U**pdate, **D**elete).
- **Cuidador**

  La persona que está a cargo de cuidar al Paciente.
- **Dispositivo dedicado**

  Dispositivo que ha sido diseñado específicamente para cumplir una función
- **Emergencia**

  Cualquier incidente que podría resultar en una caída
- **Falso Negativo**

  En términos de clasificación, cuando un sistema clasifica a una observación como negativa, cuando debió haberla clasificado como positiva.
- **Falso Positivo**

  En términos de clasificación, cuando un sistema clasifica a una observación como positiva, cuando debió haberla clasificado como negativa.
- **Framework (Software)**

  Conjunto de herramientas, librerías y lenguajes que permiten el desarrollo de un software.
- **Interfaz REST**

  Interfaz de interacción entre cliente y servidor basada en operaciones HTTP.
- **Librería (Software)**

  Conjunto de funcionalidades centralizadas que permiten realizar una tarea específica.
- **Mac**

  Computadora personal que usa los sistemas operativos de escritorio de Apple.
- **Modelo (Diseño de software)**

  Representación abstracta de una entidad.
- **NoSQL**

  Todas las bases de datos que no usan el modelo relacional, sino, el de documentos.

- **Paciente**

  Persona que está bajo cuidado del Cuidador

- **Prototipo (Software)**

  Software desarrollado que implementa solo las funcionalidades claves del diseño de un sistema.

- **Repositorio (Software)**

  Ubicación centralizada donde se encuentra código

- **Seudocódigo**

  Cualquier lenguaje que imita el funcionamiento de un lenguaje de programación en concreto.

ANEXO B – ENCUESTAS DE ACEPTACIÓN

nro	pregunta
1	¿Entendió lo que hacen los botones de la interfaz de alerta a simple vista?
2	¿Qué opina del tamaño y color de los botones? ¿Le son legibles y distintivos?
3	¿Le parece útil que el teléfono vibre y le hable cuando haya una posible emergencia?
4	¿Cuán dispuesto estaría a cargar su teléfono en una funda especial que cuelgue de su cuello en una escala del 1 al 10, donde 1 es nada dispuesto y 10 es muy dispuesto?
5	¿Cuán dispuesto estaría a utilizar un smartwatch en vez del smartphone en una escala del 1 al 10, donde 1 es nada dispuesto y 10 es muy dispuesto?
6	¿Cuán probable es que recomiende esta aplicación a sus amigos o familiares en una escala del 1 al 10, donde 1 es nada probable y 10 es muy probable?
7	¿Qué opina del precio inicial de 400-500 nuevos soles?
8	¿Algunos comentarios que le gustaría añadir?

Tabla 5: Listado de preguntas de la encuesta

Sujeto	Edad	Sexo	P1	P2	P3	P4	P5	P6	P7	P8
1	**63**	**F**	Sí por supuesto	Sí para vieja como yo, son legibles, grandes. Muy bien	Me parece bien porque aparte de leer estoy escuchando.	10	10	10	Es un precio factible porque me lo pagarían mis hijos. Si son varios hermanos es más fácil. Si el hijo ha sido educado con valores no va a escatimar gastos.	Me parece que funciona para una persona de la tercera edad pero que esté en sus facultades mentales. Yo prefiero tener el teléfono colgado porque a veces uno se olvida; tenerlo en el bolsillo es difícil sacarlo. Lo recomendaría a las personas jubiladas que se quedan en la casa y sus hijos están trabajando. Sería una forma fácil de comunicarse. Con un smartwatch sería mejor porque puedo cocinar y hacer otras cosas.

2	**73**	**F**	No entendí al comienzo el botón del cuidante, debería ser el nombre de mi hija.	Está estupendo. Entre más grande mejor.	Sí me parece perfecto porque no estoy revisando mi teléfono a cada rato	9	9	10	Si se trata de salud, ahorraría si fuera necesario. Pero me parece muy factible con la pensión que recibo.	Sí lo recomiendo a otras personas pero para mis amigas de 70 deberían practicar al menos una semana para que se ajusten al sistema. Me parece fácil para comunicarme con mi hija. Si me caigo podría simplemente apretar un botón porque en ese caso puede que no pueda hablar. Por ejemplo a mi papá una vez se le bajó la glucosa y empezó a hacer movimientos descoordinados y no podía hablar. Con el aparatito simplemente hubiera apretado el botón.
3	**72**	**M**	Sí, fácil la comprensión	Se distingue bien, se mira bien.	Sí, está bien que vibre.	9	6	9	Creo que es un precio justo, además que no me gusta estar dando trabajo a mis hijos y esto me libraría de un peso	Se necesitaría hacer difusión en los medios sobre el sistema. Me gustaría que escuchara mi voz, es mucho más práctico en vez de tener que apretar a "Estoy bien". Me parece factible comprarlo como jubilado, pero la situación varía de persona a persona. He escuchado que los smartwatch son caros así que prefería usar mi celular.
4	**67**	**F**	Sí, aunque al apretar el botón con la figurita de la ambulancia no pasaba nada.	Se ve bien claro, me gusta que cada botón tenga su color.	Sí, aunque espero que haya la opción de desactivar la voz.	4	10	10	Yo no tengo problema con ese precio si me va a dar seguridad	Me gustó mucho el programita, está muy útil, siempre cargo el celular conmigo pero no me gustaría tenerlo en el cuello porque mi celular es grande y pesa.

Tabla 6: Respuestas a la encuesta