Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Centro de Investigación en Comunicación y Redes

DESARROLLO DE APLICACIÓN MÓVIL “ALZRASTREO” PARA EL ACOMPAÑAMIENTO DE PACIENTES CON LA ENFERMEDAD DE ALZHEIMER

Trabajo Especial de Grado

presentado ante la Ilustre

Universidad Central de Venezuela

por la Bachiller

Carla Isabel Tellería Bonilla, C.I. 19.367.883

para optar al título de

Licenciada en Computación

Tutores:

Profa. Ana Morales y Prof. Antonio Silva Sprock

Caracas, junio de 2017

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Acta del Veredicto

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la Bachiller Carla I. Tellería B., C.I.: 19.367.883 con el título "Desarrollo de aplicación móvil "AlzRastreo" para el acompañamiento de pacientes con la enfermedad de Alzheimer", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 8 de Junio de 2017 a las 8:00 am, para que su autora lo defendiera en forma pública, en PB III en la Escuela de Computación, mediante una exposición oral de su contenido, y luego de la cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo. En fe de lo cual se levanta la presente acta, en Caracas el 8 de Junio de 2017, dejándose también constancia de que actuaron como coordinadores del Jurado la Profesora Tutora Ana Morales y el Profesor Tutor Antonio Silva.

Prof. Ana Morales
(Tutora)

Prof. Antonio Silva
(Tutor)

Prof. Héctor Arrechedera
(Jurado Principal)

Prof. Miguel Astor
(Jurado Principal)

DEDICATORIA

Para mi familia,

quienes siempre me acompañaron y brindaron todo su apoyo.

AGRADECIMIENTOS

A mi papá y mamá, por estar conmigo en cada paso.

A Andrea, por su apoyo incondicional.

A Pablo, por su ánimo siempre.

A Caye, Lucy y Anya, por su confianza en mí.

A la U.C.V. por las segundas oportunidades.

A mis tutores, Ana y Antonio, por sus consejos.

Gracias.

RESUMEN

La desorientación espacial es uno de los síntomas más característicos asociados a la enfermedad de Alzheimer, la cual se manifiesta como una intensa necesidad en la persona de irse a otro lugar, generalmente sin tener idea de a dónde. Esto, junto con las dificultades de reconocer lugares familiares, pone en evidencia los peligros que pueden enfrentar estos pacientes y, por consiguiente, la dificultad de sus guardianes o cuidadores de cuidar de ellos. En este trabajo especial de grado, se implementa el desarrollo de una aplicación móvil multiplataforma como una solución a esta problemática, y de esta manera asistir y mejorar la calidad de vida de personas con la enfermedad de Alzheimer, y sus guardianes o cuidadores.

En este sentido, la aplicación móvil desarrollada está basada en geolocalización y tiene como principal objetivo permitir conocer la ubicación de estos pacientes por medio de los servicios de ubicación de los dispositivos móviles, además de la definición de zonas geográficas consideradas seguras, como su casa o trabajo, con el fin de notificarles a los guardianes pertinentes sus paraderos en caso de que los pacientes se encuentren fuera de algún lugar seguro, y de esta manera proveer asistencia en caso de una posible emergencia. La aplicación móvil implementada sigue la metodología ágil llamada Mobile-D y está construida bajo la plataforma Android, utilizando Apache Cordova y Ionic para su desarrollo.

Palabras claves: aplicación móvil, enfermedad de Alzheimer, Mobile-D, geolocalización, Apache Cordova.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La enfermedad de Alzheimer es una de las principales causas de muerte en adultos mayores en el mundo y uno de los problemas más serios de salud de hoy en día [1].Está caracterizada por comenzar con un deterioro lento sobre las funciones de memoria y otras funciones cerebrales, hasta terminar en la muerte. Debido a que afecta directamente al cerebro causando daños en importantes procesos cognitivos [1, 2], la enfermedad de Alzheimer no afecta únicamente a aquellos que la padecen, sino también a sus familiares y cuidadores.

Bajo este contexto, la introducción de tecnologías como una forma de solventar y aliviar los problemas derivados de esta enfermedad, ha sido de gran beneficio para los cuidadores de pacientes con Alzheimer. De esta manera, emergen nuevas tecnologías que tienen como principal objetivo proveer asistencia y mayor independencia a estas personas, para tratar de aportar mayor calidad de vida a sus cuidadores, respondiendo a dificultades asociadas a las necesidades de esta enfermedad.

Este trabajo especial de grado se enfocó al problema de la desorientación espacial y dificultad en reconocer lugares familiares, que forman parte de los síntomas asociados con la enfermedad de Alzheimer. La desorientación espacial puede manifestarse en el paciente en diferentes etapas de la enfermedad, como una intensa necesidad de la persona de irse a otro sitio, a veces sin tener idea de a dónde, y con dificultades en reconocer los lugares conocidos. En este sentido, son evidentes los riesgos que una persona mayor puede padecer bajo un episodio de desorientación espacial; perderse, tener accidentes e incluso morir, son algunos de los peligros que enfrentan.

Por esta razón, este trabajo tiene como objetivo el desarrollo de solución a esta problemática haciendo uso de la tecnología como forma de proveer asistencia, y aprovechándose de la masificación del uso de teléfonos celulares. De este modo, se implementó el desarrollo de una aplicación móvil multiplataforma basada en geolocalización, que permite conocer la ubicación de los pacientes que padecen la enfermedad de Alzheimer a sus cuidadores, notificándoles de sus paraderos en caso de que éstos se encuentren fuera de algún lugar seguro, y de esta manera proveer asistencia en caso de una potencial emergencia.

Con este propósito, este trabajo de investigación está estructurado en tres capítulos principales. El primer capítulo representa el planteamiento del problema que se abarcó, donde se abordó la formulación del problema, una propuesta de solución, su correspondiente justificación, y el alcance. Asimismo, se presenta el objetivo general y los objetivos específicos, así como la metodología de desarrollo utilizada.

El segundo capítulo presenta el marco teórico de la investigación, que tiene como finalidad definir los conceptos y tecnologías claves utilizadas a lo largo de la implementación de la aplicación móvil; en donde se encuentra una definición de la enfermedad de Alzheimer y conceptos sobre las tecnologías móviles, las tecnologías web, las bases de datos NoSQL y, finalmente, las tecnologías geoespaciales.

En el tercer capítulo se presenta el marco aplicativo de este trabajo especial de grado. Se detallan las fases de la metodología Mobile-D utilizada, para la definición de la implementación realizada en forma iterativa e incremental, que dio como resultado la aplicación móvil.

Por último, se encuentran los posibles trabajos futuros, las conclusiones de este trabajo especial de grado y las referencias bibliográficas.

CAPÍTULO I PLANTEAMIENTO DEL PROBLEMA

En este capítulo, se presenta la formulación del problema que aborda este trabajo en detalle, junto con una propuesta de cómo podría ser solventado y su justificación correspondiente. Asimismo, se exponen los objetivos de la investigación, el alcance de ésta y la metodología utilizada.

1. FORMULACIÓN DEL PROBLEMA

En esta sección, se formula la problemática con respecto a la enfermedad de Alzheimer que se pretende resolver. Se describe una propuesta de solución, su correspondiente justificación y el alcance de este trabajo.

1.1.Problema

La enfermedad de Alzheimer es una de las principales causas de muerte en adultos mayores en el mundo y la más común forma de demencia [1]. Es una enfermedad neurodegenerativa, caracterizada por comenzar con un deterioro lento sobre las funciones de memoria y, progresivamente, otras funciones cerebrales. Como consecuencia de esta pérdida de funciones cognitivas, la enfermedad de Alzheimer culmina en la muerte para aquellos que la padecen, usualmente en un plazo de 3 a 9 años [2] desde su diagnóstico. Parte de los síntomas incluyen pérdida de memoria, problemas conductuales, alucinaciones, delirios y tendencias a deambular, y perderse [3]. Si bien científicos continúan estudiando la enfermedad en búsqueda de sus causas y cura, muchas otras investigaciones se han realizado con respecto a cómo proporcionar un mejor cuidado y calidad de vida a estos pacientes [1, 2, 3, 4].

En este sentido, la tecnología ha proveído un gran alivio no sólo para las personas que poseen esta enfermedad, sino también para sus cuidadores o guardianes. Asimismo, avances tecnológicos han propiciado un aumento de dispositivos y aplicaciones que tienen como principal propósito brindar a estos pacientes una vida más segura e independiente, y a sus guardianes de herramientas que asistan con el trabajo de cuidarlos [1, 4, 5]. Estas tecnologías de asistencia prometen ayudar con una gran diversidad de problemas; entre los que se destacan problemas de lenguaje, audición y vista, además de dar apoyo a estas personas en actividades cotidianas, como lo son sentirse seguros al caminar, encontrar su camino sin perderse, y les sea posible notificar a sus guardianes de cualquier emergencia, y de esta manera ayudar a estos guardianes, mejorando su calidad de vida al proveer formas de cuidar a sus pacientes.

De esta manera, este trabajo tiene como principal objetivo proponer e implementar el desarrollo de una solución al problema de la desorientación espacial que padecen las personas con la enfermedad de Alzheimer. Para este problema, varias aplicaciones han sido desarrolladas como una solución, las cuales tienen como objetivo brindar apoyo basado en geolocalización para el monitoreo de la ubicación de estos pacientes.

En efecto, utilizar este tipo de tecnologías representa una gran ayuda para estos pacientes y sus guardianes, ofreciendo capacidades de monitoreo, seguimiento, avisos y recordatorios automatizados, herramientas de localización y comunicación, y hasta de participación social [6]. No obstante, el uso de este tipo de tecnologías de asistencia puede llegar a ser bastante costoso, especialmente si se tratan de dispositivos complejos y sofisticados.

Con relación a este último punto presentado, vale acotar que, en Venezuela, donde la economía de hoy en día cada vez se agrava más y el poder adquisitivo es limitado, conseguir estos tipos de dispositivos dedicados y diseñados para resolver necesidades específicas para personas con la enfermedad de Alzheimer, se vuelve más difícil. Por esta razón, se deben utilizar soluciones análogas que contribuyan a asistir con este tipo de dificultades, dando especial consideración al menor consumo de recursos posible, y aprovechando la situación socioeconómica del país.

Bajo este contexto, y teniendo en cuenta que en Venezuela el 45% de la población es dueña de un teléfono inteligente [7], el desarrollo de aplicaciones para dispositivos móviles, especialmente para teléfonos inteligentes, solventaría de alguna manera esta problemática. Asimismo, ¿es posible el diseño de una aplicación que provea los mismos servicios que un dispositivo físico dedicado a un menor costo? En efecto, mediante la utilización de aplicaciones móviles que aprovechen las capacidades de los dispositivos, es posible encontrar una asistencia equivalente para las dificultades que acarrea la enfermedad de Alzheimer.

El problema de la desorientación espacial es uno de los temas que más ha tratado de resolverse a través de la tecnología. Si bien innovaciones en los dispositivos con respecto a tecnologías geospaciales ha propiciado la creación de dispositivos localizadores dedicados [4, 5], también ha favorecido el desarrollo de aplicaciones que utilizan unidades de geolocalización integradas en los dispositivos móviles como una manera de monitorear y rastrear a los pacientes. De esta manera, el teléfono inteligente se convierte en una solución a un problema a través de una aplicación móvil.

Bajo estas consideraciones, ciertas interrogantes emergen en lo referente a una aplicación móvil que permita al usuario la libertad que necesita, pero que aun así provea un monitoreo y rastreo constante como asistencia a los guardianes y familiares. En este aspecto, ¿cómo se percata la aplicación de que su usuario se encuentra fuera de un lugar seguro? Y derivado a esto, ¿cómo advertir al guardián que el paciente está perdido?

1.2.Propuesta

Como solución a la problemática planteada, este trabajo propone el desarrollo de una aplicación móvil que tenga como propósito servir como acompañante a las personas que padecen la enfermedad de Alzheimer. La aplicación desarrollada llamada “AlzRastreo”, está basada en información geoespacial obtenida mediante los servicios de ubicación integrados en los sistemas operativos de los dispositivos móviles. De esta manera, permite dar a conocer la ubicación de aquellas personas con Alzheimer a sus guardianes a través de notificaciones y, por consiguiente,

provee asistencia en caso de una potencial emergencia e impactando de manera positiva en la calidad de vida de sus guardianes o cuidadores.

Para lograr esto, la aplicación define dos tipos diferentes de usuarios: usuarios pacientes y usuarios guardianes, que el usuario debe seleccionar al momento de registrarse. Cada tipo de usuario posee su propio conjunto de funcionalidades inherentes al rol seleccionado. Los usuarios guardianes agregan a los usuarios pacientes que desean monitorear, quienes a su vez deben aceptar esta invitación como forma de mantener su privacidad. De esta forma, y una vez un usuario paciente acepta la invitación de un usuario guardián, se crea una relación entre ambos equivalentes a una relación de "amistad". Como es posible para un usuario paciente tener múltiples guardianes, se define el concepto de "Guardián Primario", quien es aquel que es aceptado como el primer guardián por el paciente. El guardián primario tiene derechos especiales sobre la cuenta de sus usuarios pacientes, como lo es registrar más guardianes a un usuario paciente en específico y ceder su rol de guardián primario.

Por lo tanto, la aplicación móvil obtiene la ubicación geoespacial de los usuarios pacientes de forma continua, con el fin de notificar esta información a aquellos usuarios guardianes pertinentes, en caso de que el usuario paciente no se encuentre en algún lugar seguro. La forma en la que la aplicación se percata de si el paciente está o no en un lugar seguro, es a través de las "Zonas Seguras". Una Zona Segura es una zona geográfica con un radio específico que demarca una ubicación real en la que los usuarios pacientes se encuentren seguros, y la cual es definida por los usuarios guardianes primarios para sus pacientes (Ej. hogares, trabajos, etc.).

De esta manera, si los usuarios pacientes se encuentran dentro de estas zonas seguras, la aplicación se abstiene del envío de notificaciones a los guardianes correspondientes. Por el contrario, si los usuarios pacientes salen de los límites de estas zonas seguras, entonces la aplicación se encarga de notificar a los usuarios guardianes de esto.

Asimismo, los usuarios guardianes también podrán visualizar de forma gráfica y sobre un mapa, la última ubicación almacenada del usuario paciente, a través de la interfaz de programación de aplicaciones de Google Maps. Además de esto también serán capaces de agregar a otros usuarios guardianes a un mismo usuario paciente, como forma de compartir el compromiso de cuidar de él. De este modo, la aplicación es capaz de asistir a los guardianes y pacientes en caso de un posible episodio de desorientación espacial, notificándoles a todos sus guardianes en qué lugar se encuentra el paciente.

Conjuntamente con la aplicación móvil, fue necesario también el desarrollo de un servidor Web que tiene el objetivo de establecer una conexión entre la aplicación y la base de datos central, donde los datos generados por los usuarios serán almacenados, como forma de monitoreo. De esta manera, la arquitectura final del sistema puede observarse en forma gráfica en la Figura 1.

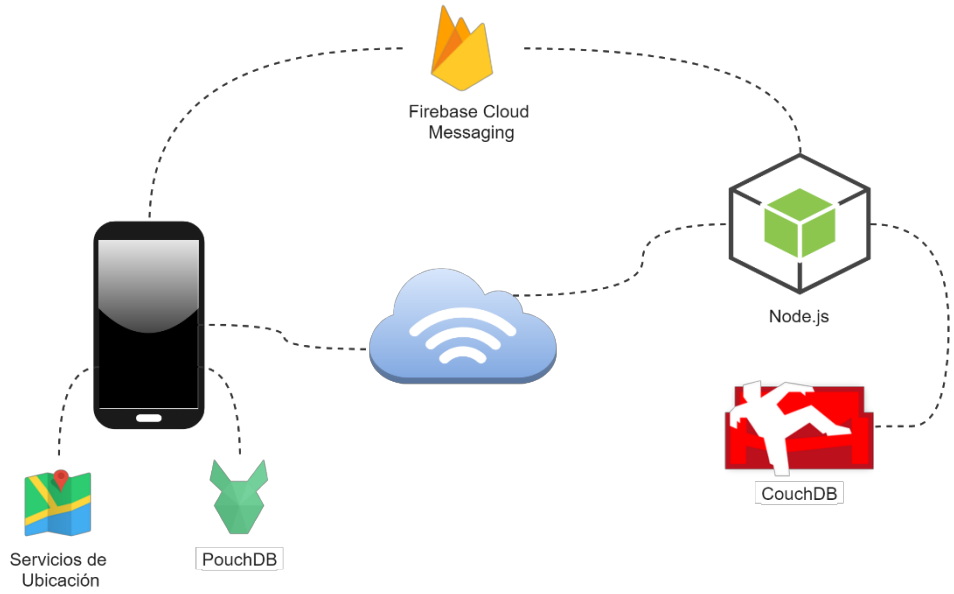

***Figura 1.*Arquitectura completa del sistema propuesto**

Como puede verse en la Figura 1, la aplicación móvil propuesta trabaja junto con un servidor Web Node.js que actúa como intermediario entre la aplicación móvil y la base de datos CouchDB. Asimismo, entre las funcionalidades del servidor Web también se encuentra la autenticación de usuarios, el monitoreo de las bases de datos de los usuarios y el envío de notificaciones a los usuarios pertinentes. La aplicación móvil establece esta conexión con el servidor a través del Internet, y usa los servicios de ubicación del teléfono, así como una base local PouchDB. Los mecanismos de notificación *push* se hacen a través del servidor y la conexión con Firebase Cloud Messaging de Google.

1.3. Justificación

La importancia de utilizar la tecnología como una forma de asistencia para aquellas personas que padecen de la enfermedad de Alzheimer es evidente. El diseño de tecnologías que promueven mayor independencia y un aumento de la calidad de vida en pacientes, y sus cuidadores, es un compromiso significativo, además de un impulso hacia una sociedad más avanzada. En virtud de ello, resulta de gran importancia contribuir en lo posible; en especial con herramientas adaptadas que pueden ser utilizadas por la mayor cantidad de personas bajo un menor costo. De esta manera, la aplicación móvil propuesta pretende dar una solución, específicamente, al presente problema de la desorientación espacial, y asistir en lo posible en el bienestar mental y físico de sus guardianes, cuidadores, y familiares.

1.4. Alcance

Con el proyecto propuesto se pretende desarrollar una aplicación móvil que obtiene la información geoespacial del usuario del dispositivo móvil, con el objetivo de notificarle a su guardián, guardianes o cuidadores de su ubicación geográfica, cuando éste salga de una zona

geográfica segura, previamente definida, impactando de manera positiva en la calidad de vida del cuidador. Dentro de la arquitectura propuesta, también se contempla el desarrollo de un servidor Web.

Asimismo, se define como alcance de este trabajo los siguientes puntos:

- Desarrollo de una aplicación móvil multiplataforma para el sistema operativo Android, que utiliza los servicios de ubicación integrados a los dispositivos móviles.
- Desarrollo de un servidor Web que funcione como intermediario entre las peticiones de los usuarios y la base de datos, el cual pueda ser accedido a través del internet.
- Implementación de mecanismos de notificación por correo electrónico y a través de notificaciones *push*.
- Implementación de funcionalidades de agregación y eliminación de usuarios pacientes para los usuarios tipo guardianes.

2. OBJETIVOS

En esta sección, se presenta el objetivo general y los objetivos específicos de la solución propuesta.

2.1. Objetivo General

Desarrollar una aplicación móvil para el acompañamiento de personas con la enfermedad de Alzheimer, basada en la geolocalización.

2.2. Objetivos Específicos

- Diseñar una aplicación móvil para el acompañamiento de personas con la enfermedad de Alzheimer, basada en la geolocalización.
- Configurar el ambiente de desarrollo de la aplicación móvil.
- Desarrollarlos componentes necesarios para la implementación de la solución; tanto para la aplicación móvil como para el servidor Web.
- Implementar un servidor que funcione como intermediario entre el servidor de base de datos y la aplicación móvil, y tenga funcionalidades de monitoreo.
- Realizar las pruebas de usabilidad y funcionalidad a la aplicación móvil para el acompañamiento de pacientes con Alzheimer, basada en geolocalización.

3. METODOLOGÍA DE DESARROLLO

En este trabajo, se propone utilizar una metodología de desarrollo ágil diseñada por Pekka Abrahamsson y otros investigadores del VTT en Finlandia [9], llamada Mobile-D para el desarrollo de la aplicación móvil para el acompañamiento de pacientes que padecen la enfermedad de Alzheimer. Esta decisión se fundamenta en las propiedades únicas de las metodologías ágiles, que la hacen especialmente aplicable como una solución natural para el desarrollo móvil, garantizando la realización de proyectos en corto plazo [10] por razón de sus

características de flexibilidad, agilidad, principios de diseños simples [11] y de procesos iterativos e incrementales.

En este sentido, Mobile-D provee de las herramientas y prácticas que mejor se adaptan a la planificación y el diseño de la aplicación móvil propuesta en este trabajo, debido a que está especialmente enfocada en superar los desafíos asociados a los desarrollos móviles [9].

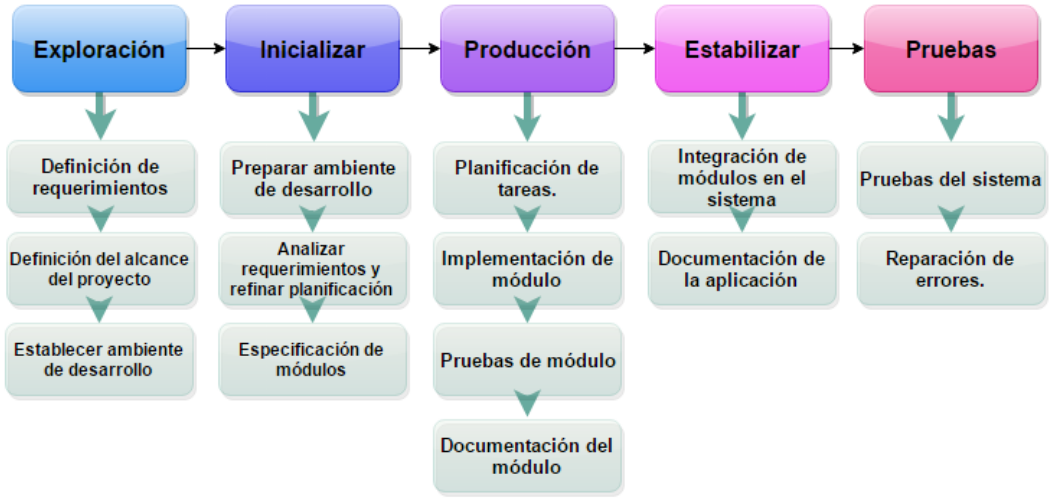

Figura 2. **Metodología utilizada para el desarrollo de la aplicación móvil basada en geolocalización Adaptación de [12]**

En la Figura 2 pueden observarse las cinco fases y sus iteraciones, descritas por la metodología Mobile-D. Estas fases son: exploración, inicializar, producción, estabilizar y pruebas [12].

En la fase de exploración, se propone la planificación y el establecimiento inicial del proyecto [13]. Es aquí donde se definirán los requerimientos, el alcance y el ambiente de desarrollo de la aplicación móvil.

La fase de inicializar tiene como propósito preparar el ambiente de desarrollo para la aplicación y predecir posibles problemas críticos que podrían ocurrir [13]. También es aquí donde se analizarán los requerimientos y refinará la planificación, además se especificarán los módulos que conforman la aplicación móvil.

La fase de producción es donde se implementan las funcionalidades y requerimientos definidos en las fases anteriores, aplicando un ciclo de desarrollo iterativo e incremental [13], y utilizando un desarrollo dirigido por pruebas (TDD) [12].

En la fase de estabilizar, es donde se integran todos los módulos que conforman la aplicación en único producto [13] y donde se define la documentación final de la aplicación móvil.

Finalmente, en la fase de pruebas es donde se toman las acciones que confirman que la aplicación móvil producida implementa las funcionalidades requeridas de forma correcta [13]. También es aquí donde se reparan los posibles errores que puedan existir.

CAPÍTULO II MARCO CONCEPTUAL

En este capítulo se definen ciertos conceptos e ideas tecnologías que permitirán abordar el problema de forma estructurada. De esta manera, se pasarán a exponer las investigaciones y definiciones que permiten organizar y conceptualizar el trabajo.

1. ANTECEDENTES

Las tecnologías de asistencia que están enfocadas al rastreo y la localización de personas que padecen Alzheimer no es algo nuevo. Tampoco es sorprendente encontrar aplicaciones móviles que tratan de proveer estos mismos servicios. En este aspecto, varios enfoques de desarrollo han sido utilizados y diferentes aplicaciones han salido al mercado para proveer asistencia, y proporcionar información precisa sobre la ubicación de estos pacientes. Por esta razón, es importante estudiar el estado del arte en cuanto a aplicaciones móviles basadas en geolocalización, enfocadas a monitorear y rastrear personas con la enfermedad de Alzheimer.

1.1. Tweri

Tweri es una solución móvil desarrollada por la empresa española Solusoft, que provee tranquilidad a los guardianes de las personas que padecen la enfermedad de Alzheimer, empoderando a las personas afectadas, al diseñar una aplicación de monitoreo posicional que permite a los pacientes con Alzheimer salir de la casa de forma autónoma [14].

Permite establecer límites seguros que están basados en el tiempo máximo que el usuario puede estar fuera de casa, o el radio de distancia máxima que tiene permitido alejarse de casa [15]. Antes de salir, se debe activar la aplicación, y cuando el tiempo o la distancia máxima son excedidos, se alerta al guardián registrado de forma automática vía correo electrónico, con la más reciente posición geográfica obtenida por el dispositivo. Adicionalmente, posee un botón de emergencia que el paciente puede presionar en cualquier momento [15].

La versión actual de la aplicación es la 2.4 y fue actualizada por última vez en diciembre del 2014. Posee un tamaño de 34,75 mega bytes, y requiere la versión Android 2.3 o versiones superiores [15] hasta la fecha. En la actualidad, Tweri no es compatible con las nuevas versiones de Android y trabaja en conjunto con la aplicación Web para el registro de los usuarios, por lo que no es una aplicación autónoma.

1.2. Alzheimer Patient y Alzheimer Caretaker

Alzheimer Patient y Alzheimer Caretaker son dos aplicaciones separadas que trabajan juntas proveyendo servicios de asistencia a aquellos guardianes y pacientes. Fue desarrollada como un proyecto por el departamento de Information and Communication Technology (ICT) de la Prince of Songkla University ubicada en Tailandia [16]. Provee un sistema de rastreo y monitoreo para los usuarios que tienen instalado Alzheimer Patient, y un sistema de notificaciones para los que tienen instalado Alzheimer Caretaker [17]. La versión actual de la aplicación Alzheimer

Caretaker es la 0.0.1, y fue actualizada por última vez en junio del 2015. Posee un tamaño de 1.5 mega bytes y requiere la versión de Android 2.3.3 o versiones superiores [17] hasta la fecha. La versión actual de la aplicación Alzheimer Patient es la 1.0.0, y fue actualizada por última vez en junio del 2015. Posee un tamaño de 2 mega bytes y requiere la versión de Android 2.3.3 o versiones superiores [18] hasta la fecha.

En este sentido y considerando lo antes expuesto, este trabajo pretende introducir nuevas funcionalidades que traigan mejoras a estas aplicaciones desarrolladas, de la siguiente manera:

- Autenticación de usuarios intrínseca a la aplicación móvil sin necesidad de ir a un sitio web externo para el registro.
- Cohesión de dos aplicaciones móviles separadas en una misma aplicación, que brinde capacidades de monitoreo y rastreo a los usuarios que tienen la enfermedad de Alzheimer. De esta forma, no es necesario tener una aplicación móvil para los pacientes y una diferente para sus guardianes, sino una única aplicación que provea funcionalidades para ambos tipos de usuarios, con sus respectivas restricciones.
- Capacidad de registrar más de un guardián o protector a un mismo usuario paciente y, de forma análoga, de registrar más de un paciente a un mismo guardián.
- Permitir la definición de usuarios guardianes primarios, que tienen el objetivo principal de manejar y definir las zonas seguras de sus pacientes, así como la habilidad de agregar otros guardianes que compartan el compromiso de cuidar a un mismo paciente.
- Notificaciones *push* y envío de correos electrónicos como modo de alertar a los usuarios guardianes de que un o uno de sus pacientes se encuentra fuera de un lugar seguro.

2. ENFERMEDAD DE ALZHEIMER

La enfermedad de Alzheimer es una de las principales causas de muerte en adultos mayores en el mundo y la más común forma de demencia. En sí misma, la demencia no es una enfermedad, sino un término general utilizado para englobar un gran grupo de síntomas que están asociados con una grave disminución de los procesos mentales, particularmente aquellos que se relacionan con procesos de memoria y de pensamiento. Para el 2015, alrededor de 46.8 millones de personas sufren de demencia a nivel global, y su mayoría está diagnosticada con la enfermedad de Alzheimer [1], lo que la convierte en uno de los problemas más serios de salud de hoy en día.

Caracterizada por comenzar con un deterioro lento sobre las funciones de memoria y progresivamente otras funciones cerebrales, la enfermedad de Alzheimer culmina en la muerte para quien la padece, debido a que en la actualidad no existe cura alguna, y aún se desconocen muchas de sus causas. Sin embargo, los científicos han llegado a un acuerdo de que, para la mayoría de las personas, la enfermedad es causada por una combinación de genética, factores ambientales y estilo de vida que afectan al cerebro a través del tiempo. Aún ahora, muchos estudios se están realizando con el objetivo de determinar los factores y las causas que llevan a una persona a padecer esta enfermedad, y de esta manera encontrar la cura.

A pesar de esto, cabe señalar que nuevas innovaciones tecnologías y emergentes han aliviado por mucho las dificultades asociadas a las necesidades de estos pacientes. Avances tecnológicos han permitido otorgarles mayor seguridad e independencia, y como consecuencia directa, una mejor calidad de vida [1 4, 5]. Aplicaciones de monitoreo, socialización, e incluso informativas son sólo algunas de las muchas formas que la tecnología provee ayuda a los que padecen la enfermedad y podría hasta salvarles la vida [4] y [6].

2.1.Definición de la enfermedad de Alzheimer

La enfermedad de Alzheimer fue descrita por primera vez en noviembre de 1906, por Alois Alzheimer [1], un médico alemán conocido por sus investigaciones en lo referente a la demencia de origen degenerativo, psicosis, psiquiatría forense y epilepsia. En la 37ma Conferencia de Psiquiatras del Sur-Oeste de Alemania en Tübingen [4], describió a un paciente de 51 años llamada Auguste D., la cual mostraba deterioro cognitivo progresivo, síntomas focales, alucinaciones, delirios y problemas psicosociales. No fue hasta 1910, cuando un psiquiatra alemán llamado Emil Kraepelin nombra esta enfermedad la “enfermedad de Alzheimer”, en la octava edición de su libro [4].

La enfermedad de Alzheimer es una enfermedad cerebral progresiva e irreversible que está caracterizada por un deterioro cognitivo que afecta las capacidades de memoria y pensamiento en las personas que la padecen. La razón de esto se debe a que causa que grandes cantidades de células nerviosas mueran, que el tejido cerebral se atrofie y que el cerebro, de forma anormal, comience a encogerse.

El cerebro está formado por miles de millones de neuronas y otras células cerebrales que tienen la responsabilidad de procesar y transmitir información entre las diferentes partes dentro del cerebro, y entre el cerebro y el resto del cuerpo. Esta información es transmitida por señales eléctricas o químicas. El cerebro posee cerca de 100 mil millones de células nerviosas, también llamadas neuronas. Una neurona está compuesta por tres partes principales: cuerpo, dendritas y un axón.

La comunicación es una de las funciones principales de las que son responsables las neuronas. Las dendritas y los axones juegan un papel fundamental en la comunicación neuronal. Las dendritas son las encargadas de recibir las señales de entrada enviadas por otras neuronas, lo que trae como consecuencia la generación de un impulso nervioso o acción potencial, que viaja por el axón donde se liberan finalmente los neurotransmisores, o mensajes químicos, cuando el axón entra en contacto con las dendritas de otra neurona [24]. Esta unión es llamada sinapsis y una neurona puede tener hasta 7.000 conexiones sinápticas con otras neuronas [25].

Para que las neuronas no mueran y realicen sus funciones correctamente, es necesario que exista un suministro de energía que las mantenga. El metabolismo es el proceso en el que las células extraen energía al descomponer químicos y nutrientes de la sangre que circula alrededor del cerebro. Las neuronas son las responsables del mayor consumo de energía en el cerebro,

aproximadamente el 90% [21], y los nutrientes indispensables que extraen son la glucosa y el oxígeno.

La supervivencia de las neuronas también depende largamente en sus capacidades de reparación, remodelación y regeneración. Continuamente, las neuronas se reparan a sí mismas y remodelan sus conexiones sinápticas dependiendo de la estimulación que reciben. El proceso de la remodelación de conexiones sinápticas es importante y vital, ya que es necesario para el aprendizaje, la memoria, el proceso de planificación y otras actividades mentales complejas.

La enfermedad de Alzheimer afecta mortalmente a las neuronas. Aun cuando es normal que por el pasar del tiempo el cerebro se encoja por el envejecimiento natural de los seres humanos, no es normal que pierda neuronas en grandes cantidades. La enfermedad de Alzheimer interrumpe los procesos que mantienen a las neuronas saludables: el metabolismo, la comunicación, y la reparación y remodelación neuronal, lo que afecta directamente a las neuronas; resultando en la detención de las funciones, pérdida de conexiones y la muerte neuronal.

Estas interrupciones de procesos neuronales tienen su origen en dos estructuras llamadas placas amiloideas y ovillos neurofibrilares, las cuales se encuentran en abundancia en aquellas personas que padecen la enfermedad de Alzheimer, a un volumen mucho mayor del que una persona saludable posee. Ambas estructuras son proteínas malformadas que se localizan en las neuronas y entre las células cerebrales, causando el mal funcionamiento y la muerte de las neuronas y sus conexiones sinápticas, y por consiguiente el deterioro de importantes procesos biológicos.

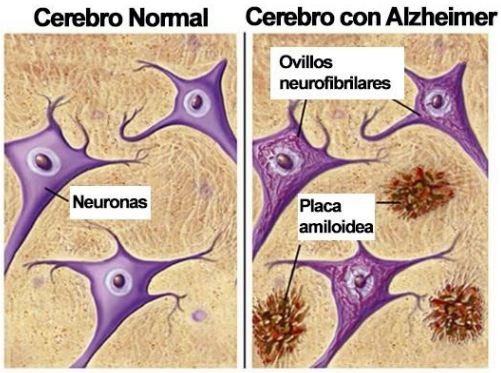

***Figura 3*. Cerebro normal Vs. Cerebro con la enfermedad de Alzheimer. Adaptación de [22]**

En la Figura 3, puede verse de forma gráfica cómo la formación de las placas amiloideas y los ovillos neurofibrilares contribuyen con la degradación de los procesos cerebrales y, por consiguiente, con el normal funcionamiento del cerebro.

Como el daño que causa la enfermedad es inicialmente en el hipocampo, la cual es una de las partes del cerebro que involucra la creación de recuerdos, la memoria es uno de los primeros factores que se ve afectado. Progresivamente, la enfermedad se va esparciendo por todo el cerebro y continúa dañando otras partes que involucran actividades responsables por el lenguaje, razonamiento, pensamiento y comportamiento.

Finalmente, una persona que padece la enfermedad de Alzheimer tiene tantas áreas del cerebro dañadas que se convierte en alguien indefenso y sin capacidades de responder a estímulos exteriores.

2.2.Síntomas del Alzheimer

Aunque las causas de la enfermedad de Alzheimer continúan siendo un misterio para los científicos de hoy en día, los síntomas tanto físicos como mentales son muy bien conocidos y documentados. Es importante destacar que el curso de la enfermedad varía para cada persona, aun cuando existen ciertos estados o etapas que son compartidos por la mayoría de los individuos.

Para describir los diferentes síntomas que ocurren a las personas con la enfermedad de Alzheimer, se separa el curso de la enfermedad en etapas. Cada etapa presenta signos típicos de los procesos que ocurren en el cerebro como consecuencia de la evolución de la enfermedad en el tiempo. En esta sección, se explicará más detalladamente las etapas de la enfermedad de Alzheimer, junto con sus síntomas asociados. Estas etapas son: etapa pre-clínica, etapa leve, etapa moderada y etapa grave.

2.2.1. Etapa Pre Clínica

Antes de que algún síntoma de pérdida de memoria se manifieste, se cree que los cambios cerebrales que involucran el esparcimiento de la enfermedad en el cerebro y, por consiguiente, el deterioro y la muerte de las células cerebrales, comienza hasta 10 ó 20 años antes de que el primer síntoma aparezca [23]. El estudio y reconocimiento de esta etapa es muy importante para futuras investigaciones y la creación de nuevos, y más eficaces tratamientos.

2.2.2. Etapa Leve

En esta etapa los problemas de pérdida de memoria y de otras habilidades cognitivas comienzan a manifestarse en los pacientes. Es cuando, normalmente, se diagnostica la enfermedad debido a que los primeros síntomas clínicos comienzas a aparecer. Algunos de estos síntomas incluyen [24]:

- Pérdida de memoria y otras dificultades cognitivas.
- Confusión sobre la ubicación de lugares familiares, por lo que la persona puede perderse y tender a deambular.
- Problemas en el manejo del dinero y el pago de facturas.
- Pérdida de espontaneidad e iniciativa.
- Cambios de humor y de personalidad, aumento de la ansiedad y, en algunos casos, de la agresividad.

2.2.3. Etapa Moderada

En esta etapa, el daño cerebral se ha esparcido en áreas del cerebro que están involucradas en procesos como el control del lenguaje, razonamiento y procesamiento sensorial. Esto se debe a que ahora la corteza cerebral se está viendo afectada y el cerebro continúa encogiéndose. Problemas como perderse o deambular se agravian por el deterioro de la orientación espacial, al igual que los problemas con pérdida de memoria y de comportamiento. Comienza a ocurrir que el paciente empiece a olvidar familiares y seres queridos. Otros síntomas incluyen [24]:

- Aumento de arranques de ira.
- Dificultad con el lenguaje, problemas con leer y escribir.
- Incapacidad de aprender nuevas cosas.
- Inquietud, agitación, ansiedad, llanto, entre otros.
- Oraciones o movimientos repetitivos.
- Alucinaciones, delirios, suspicacia y paranoia.
- Dificultad en realizar actividades como vestirse, hacer café, u otras actividades cotidianas.

2.2.4. Etapa Grave

La última etapa de la evolución de la enfermedad de Alzheimer es la etapa grave o severa. Las placas amiloideas y los ovillos neurofibrilares se han esparcido por la mayor parte del cerebro, afectando todo tipo de procesos mentales y neuronales. En esta etapa es imposible para las personas reconocer a sus seres queridos o familiares y pierden toda capacidad de comunicación. En este momento, la persona que padece la enfermedad es completamente dependiente de otros. Los síntomas finales incluyen:

- Pérdida de peso.
- Convulsiones.
- Infecciones cutáneas.
- Aumento de sueño. Las personas están la mayoría del tiempo durmiendo.

2.3. Tipos de Alzheimer

Aun cuando no se entiende del todo la enfermedad de Alzheimer, investigaciones realizadas han dado a conocer el rol importante de los genes en el desarrollo de la enfermedad.

Los genes son segmentos contenidos en los cromosomas que contienen ADN y que se encuentran localizados en casi todas las células del cuerpo. Son considerados como la unidad básica de herencia. Los genes son los que poseen las instrucciones para la construcción de proteínas, por lo que incluso pequeñas alteraciones o mutaciones en un gen pueden producir proteínas malformadas, lo que a su vez resulta en el mal funcionamiento celular y, eventualmente, en enfermedades. Una de estas enfermedades es la enfermedad de Alzheimer;

científicos han identificado ciertas regiones del genoma que pueden ser factores de riesgos para la enfermedad.

Hay dos tipos de enfermedad de Alzheimer: de inicio precoz y de inicio tardío. Descubrimientos realizados han demostrado que para ambos tipos existen componentes genéticos involucrados. En esta sección se hablará con más detalles de estos tipos.

2.3.1. Alzheimer de Inicio Precoz

El más raro tipo de enfermedad de Alzheimer es el de inicio precoz, el cual solamente cuenta con el 5% de todos los casos diagnosticados [24], y normalmente ocurre en personas entre los 30 y los 60 años. Aunque algunos casos de inicio precoz no tienen una causa conocida, la mayoría se debe a rasgos heredados de sus familias por medio de los genes.

La razón de esto se debe a que algunas familias tienen una mutación en los cromosomas 21, 14 y 1 en su ADN. La mutación en el cromosoma 21 causa la producción anormal del precursor de las amiloideas. En los cromosomas 14 y 1, la mutación causa una producción de una estructura llamada presenilina 1 y 2, correspondientemente, que ayuda a la creación de placas amiloideas.

Estas mutaciones juegan un rol primordial en la creación de las dañinas placas amiloideas, una de las dos principales razones que induce la muerte de neuronas en el cerebro. En estas familias, los hijos tendrán hasta un 50% de riesgo de padecer la enfermedad de Alzheimer [23].

2.3.2. Alzheimer de Inicio Tardío

El Alzheimer de tipo tardío ocurre en personas luego de los 60 años, y es el más común tipo de enfermedad de Alzheimer. Las causas de este tipo siguen siendo un misterio, pero se cree que tienen que ver por una combinación de factores genéticos, ambientas y estilo de vida que afectan al cerebro a través del tiempo. Aunque las causas y los factores de riesgo siguen siendo, en gran parte, desconocidos, se ha podido identificar un componente genético que no tiene relación con las mutaciones de tipo de inicio precoz, pero sí con la aparición de la enfermedad de Alzheimer en las personas.

El gen apolipoproteína E, encontrado en el cromosoma 19 del ADN humano se considera implicado en la enfermedad de Alzheimer [23]. Este gen define el proceso de la creación de una proteína que ayuda a llevar el colesterol y otros tipos de grasa en el flujo sanguíneo. Tiene tres diferentes formas, también conocidos como alelos, estos son: ε2, ε3, y ε4.

La apolipoproteína E ε2 es la forma más rara del alelo, sólo manifestándose entre un 5% a un 10% de la población [25], y a diferencia de los otros alelos, puede que reduzca el riesgo, ya que la enfermedad de Alzheimer se desarrollaría más tarde en una persona con este gen que con la forma ε4.

La apolipoproteína E ε3 es la más común de las formas, manifestándose entre un 70% a un 80% de la población [25], y se piensa que es una forma que juega un rol neutral en el desarrollo de la enfermedad, ni aumentando ni reduciendo los riesgos.

Por otro lado, la apolipoproteína E ε4 está presente entre un 10% a 15% de la población [25] y se considera un gen de riesgo, ya que incrementa el riesgo de padecer la enfermedad de Alzheimer. Sin embargo, la razón por la que esto sucede sigue sin conocerse.

No obstante, es importante mencionar que, aunque las genéticas pueden explicar parte del desarrollo y causas de la enfermedad de Alzheimer, no lo explica todo, por lo que aún se continúan realizando investigaciones para encontrar otras posibilidades que puedan causar esta enfermedad, y en particular este tipo.

2.4. Comportamientos de Pacientes con Alzheimer

Como se describió en secciones anteriores, la enfermedad de Alzheimer causa que las personas que la padecen, se comporten diferente a lo normal y de formas poco predecibles. Comportamientos como aumento de la agresividad, paranoia, alucinaciones, confusión, ansiedad, deseos de deambular y aumento de las posibilidades de perderse son comunes, y con el paso de tiempo y el desarrollo progresivo de la enfermedad, la gravedad de estos comportamientos va aumentando. Entre el 70 y 90% de las personas que padecen la enfermedad de Alzheimer desarrollan uno o más síntomas conductuales [24]. La identificación y el manejo de estos síntomas son vitales para el cuidado de estos individuos por parte de sus guardianes.

Ciertos comportamientos están presentes en todas las etapas de la enfermedad, y en casi todos los pacientes a lo largo de la enfermedad. Entre los comportamientos más frecuentes se encuentran: la apatía, depresión y ansiedad. Hiperactividad, que a su vez engloba agitación, desinhibición e irritabilidad, también son observables en gran parte de los pacientes. Delirios y paranoias pueden presentarse en la forma de un convencimiento del individuo de que su casa no es su casa, que otras personas están robando, que los familiares o guardianes son impostores, entre otros. Con frecuencia, alucinaciones visuales, auditivas y olfativas también están presentes, y son parte de los cambios de comportamientos asociados a la enfermedad de Alzheimer.

Vagar, deambular y perderse son conductas habituales para estas personas. Una persona que padece la enfermedad puede que no recuerde su nombre o dirección, lo que lleva a desorientación, incluso en lugares familiares. Un paciente perdido por más de 12 horas tiene un 50% de probabilidad de estar herido o incluso, de morir [25].

Es importante identificar las razones de por qué estos comportamientos y síntomas ocurran, ya que abren el camino a estudios de nuevos tratamientos farmacológicos y no farmacológicos. Tratar estos comportamientos ha probado ser importante a la hora de ayudar a las personas que padecen la enfermedad y habilitan la posibilidad de hacer más fácil para los guardianes el cuidarlas.

2.5. Cuidados para los Pacientes con Alzheimer

La enfermedad de Alzheimer puede tener costos muy elevados para los familiares y guardianes de aquellos que la padecen [1], [23]. Desgastes tanto físicos como mentales son muy normales y

algo con lo que se debe lidiar diariamente [24]. El cuidado de estas personas presenta desafíos, sobre todo en la última etapa de la enfermedad, cuando el final de la vida está cerca.

Es crucial mejorar el cuidado y proveer apoyo emocional, y práctico para estos pacientes. Nuevos tratamientos que proveen una manera de retardar el proceso de la enfermedad pueden ayudar mucho a mantener el funcionamiento de las personas y la realización de actividades cotidianas, lo que a su vez alivia el estrés físico y emocional de los guardianes.

Debido al progreso degenerativo neuronal de la enfermedad, muchos guardianes y familiares terminan realizando actividades del día a día para las personas que padecen la enfermedad, como lo es bañarlas y darles de comer, entro otros trabajos que pueden llegar a ser difíciles y tristes.

La introducción de la tecnología como forma de mejorar el manejo y la calidad de vida a personas que sufren de Alzheimer es una realidad. Con los avances tecnológicos en la actualidad, dispositivos nuevos y mejores entran al mercado con la finalidad de mejorar las vidas de estas personas.

Dispositivos móviles para el monitoreo de la temperatura, movimiento y ritmo cardíaco están siendo diseñados y construidos. Asimismo, dispositivos que triangulan la ubicación de personas pérdidas que olvidaron dónde quedaban sus hogares son utilizados cada vez con mayor frecuencia [4], [6], como una solución a la desorientación espacial. El uso de estos sistemas de información y de comunicaciones se hace cada vez más popular para el cuidado para los pacientes con Alzheimer, y a su vez ayuda a mantenerlos con una mejor calidad de vida y alivia la dificultad de los guardianes de protegerlos y darles el mejor cuidado posible.

3. TECNOLOGÍAS MÓVILES

Los dispositivos móviles, especialmente los teléfonos inteligentes, se han ido convirtiendo con el pasar de los años en un accesorio imprescindible para las personas. Según el Pew Research Center en un estudio publicado en febrero del 2016, en Venezuela el 45% de la población es dueña de un teléfono inteligente [19]. Tomando en cuenta esto y con el rápido crecimiento de las tecnologías móviles, no es una sorpresa que exista un auge en el desarrollo de aplicaciones para estos dispositivos.

Vale acotar que el diseño de una aplicación móvil está íntimamente relacionado con el sistema operativo o plataforma en la que se quiere desarrollar la aplicación. En este sentido, se conoce como sistema operativo a un programa de software que está diseñado para administrar el hardware de un dispositivo, actuando como un intermediario entre el usuario y el hardware. El propósito de un sistema operativo es proveer un ambiente en el que los usuarios puedan ejecutar programas o aplicaciones de manera conveniente y eficiente [20]. Los sistemas operativos que operan sobre sistemas móviles como teléfonos inteligentes y tabletas, son llamados sistemas operativos móviles, o SO móvil.

El diseño de un SO móvil tiene restricciones específicas en cuanto a los recursos limitados que poseen los dispositivos móviles, a diferencia de los sistemas operativos para computadoras de

escritorio. Entre las características más representativas de los dispositivos móviles, se encuentra el tamaño de la memoria, poder de procesamiento, capacidad de la batería y limitada cantidad de capacidades de cómputo, y comunicaciones. Por esta razón, un SO debe estar diseñado para resolver este tipo de necesidades que son íntegramente diferentes a el desarrollo de programas tradicional.

Existen varios sistemas operativos móviles que soportan dispositivos como teléfonos inteligentes y tabletas, entre los que se encuentran Android, iOS, Windows Phone, Firefox OS, BlackBerry 10 y Ubuntu TouchOS. La batalla de los SO móviles para liderar el mercado ha sido difícil, pero en los últimos años se ha establecido una tendencia; según cifras publicadas por la International Data Corporation (IDC) en agosto de 2015, Android domina el mercado en un 82,8%, seguido de iOS con un 22,3% y Windows Phone con un 4.2% [21].

Como consecuencia de lo antes descrito, para este trabajo especial de grado se decidió trabajar bajo la plataforma Android, dado que esta plataforma fue la que mejor se adaptaba a los recursos disponibles y por su gran penetración en el mercado venezolano.

3.1. Sistema Operativo Android

Android es una pila de *software* de fuente abierta creada para un amplio rango de dispositivos móviles con diferentes factores de forma [22]. Asimismo, Android tiene dos objetivos principales; el primero es crear una plataforma de *software* disponible para fabricantes y desarrolladores, que haga realidad la creación de sus ideas innovadoras; y el segundo es introducir un producto exitoso y real que mejore la experiencia móvil de los usuarios[22]. En la actualidad, Android es la plataforma móvil más popular del mundo [23].

Originada por un grupo de compañías conocidas como la Open Handset Alliance, lideradas por Google, Android está basado en el *kernel* de Linux. Su primera versión pública fue distribuida el 12 de noviembre de 2007, y el primer teléfono inteligente con el sistema operativo Android salió al mercado el 23 de septiembre de 2008 [24]. A diferencia de iOS, Andorid corre en una gran variedad de plataformas móviles incluyendo tabletas y hasta PCs; como consecuencia de esto, Android es bastante atractivo a los fabricantes del *hardware*, lo que explica en parte su gran popularidad en los últimos años y lo que lo convierte en el líder en el mercado de SO móviles en la actualidad.

La pila de capas de *software* que compone Android, proporciona un rico conjunto de *frameworks* para el desarrollo de aplicaciones móviles. En la Figura. 4 se muestra la arquitectura de Android con más detalle.

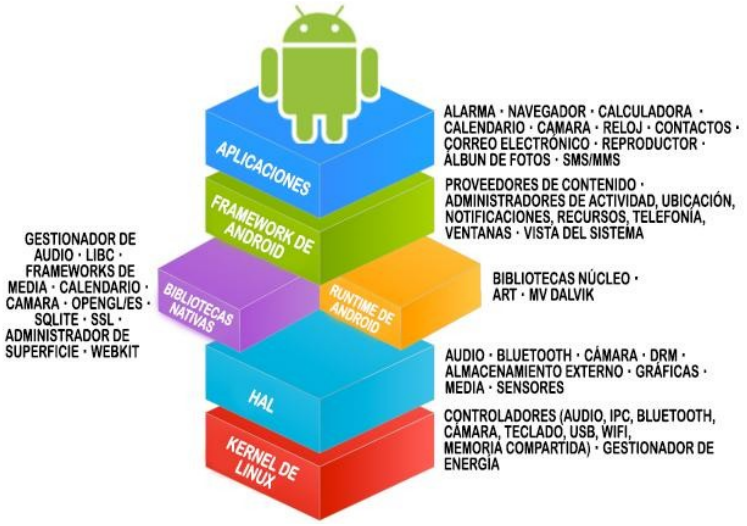

***Figura 4*. Arquitectura de Android. Adaptado de [22]**

Android puede separarse en cinco capas de abstracción principales: una capa de aplicaciones, una capa *framework* de aplicaciones, una de bibliotecas nativas que contiene también el Android Runtime, una capa HAL y finalmente, el *kernel* de Linux. El *kernel* de Linux es un *kernel* modificado por Google y se encuentra fuera de las distribuciones normales de las versiones de Linux [20], Android depende de éste para servicios básicos del sistema como el acceso a los controladores del dispositivo, manejo de memoria y manejo de procesos, así como el administrador de energía.

La capa conocida como Hardware Abstraction Layer, o HAL por sus siglas en inglés, define las interfaces estándares que los proveedores de *hardware* implementan y permite a Android ser agnóstico sobre las implementaciones de controladores de nivel inferior, que dependen de los diferentes fabricantes. HAL es el que permite implementar funcionalidades sin necesidad de afectar o modificar el sistema a nivel superior [25].

El Android Runtime incluye las bibliotecas núcleo que contienen la Máquina Virtual Dalvik la cual se encarga de ejecutar las aplicaciones. Google reemplazó la máquina virtual Dalvik por Android Runtime desde Android versión 5.0 (Lollipop) [26], por lo que Dalvik sólo se encuentra en la versión 4.4 de Android (KitKat) y distribuciones anteriores.

Android provee un conjunto de bibliotecas núcleo de C/C++, las cuales se exponen a los desarrolladores a través del *framework* de aplicaciones [19], el cual también se encarga de proporcionar los API que las aplicaciones utilizan para la compartición de datos, recepción de las notificaciones y el acceso al sistema telefónico [27].

3.2.Estrategias de desarrollo

El desarrollo de aplicaciones para dispositivos móviles difiere en gran medida al desarrollo tradicional, dada las particularidades específicas que son nativas a estos dispositivos y que deben ser consideradas a la hora de desarrollar aplicaciones móviles. De esta manera, entre los

elementos a tomar en cuenta, se encuentra la gran diversidad de plataformas que existen en el ámbito móvil y la variedad de tecnologías, estándares, y protocolos de red utilizados hoy en día entre las diferentes plataformas y fabricantes. Acuñado a esto, las limitaciones físicas inherentes de los dispositivos móviles como la capacidad de la batería, el procesador, la capacidad de la memoria principal y la fragmentación, también juegan un papel importante en el diseño de una aplicación.

Por esta razón, la decisión de una estrategia de desarrollo es un factor importante puesto que permite minimizar y acelerar el desarrollo multiplataforma. El proceso de escoger una estrategia de desarrollo para una aplicación móvil conlleva ciertos parámetros como tiempo del proyecto, público objetivo, funcionalidades, entre otros. Cada estrategia trae consigo beneficios y limitaciones específicas; encontrar la que mejor se adapte puede ser desafiante.

Como consecuencia de esto, y para satisfacer las diferentes necesidades existentes, se han definido varias estrategias de desarrollo que pueden ser escogidas en el momento de desarrollar una aplicación móvil, entre las que se encuentra el desarrollo nativo, el web y el desarrollo multiplataforma (o híbrido). A continuación, se describirá brevemente en qué consiste cada estrategia.

3.2.1. Desarrollo Nativo

El desarrollo de aplicaciones nativas está enfocado específicamente para un dispositivo en particular y su sistema operativo en específico y, por consiguiente, están escritas con el lenguaje de programación por defecto de cada plataforma. Este tipo de aplicaciones son descargadas desde una tienda de aplicaciones web e instaladas en el dispositivo móvil.

Para el desarrollo de una aplicación nativa, generalmente es necesario un entorno de desarrollo o IDE dependiente de cada plataforma, entre los que se encuentran Android Studio, para Android; xCode, para iOS; y Microsoft Visual Studio, para Windows 10 [28]. De esta manera, utilizando el *software development kit* (SDK) que provee cada plataforma, una aplicación es capaz de acceder a los datos de los dispositivos y/o cargar datos desde un servidor externo [29].

Sin embargo, y aun cuando un desarrollo nativo conlleva una aplicación que explota las capacidades de la plataforma móvil para la que está diseñada, el código escrito para una plataforma no puede ser usado para otra, por lo que el desarrollo y el mantenimiento de estas aplicaciones en diferentes sistemas operativos es una empresa larga y costosa [30].

3.2.2. Desarrollo Web

Como una aplicación web tradicional, el desarrollo para aplicaciones web en el contexto móvil también está basado en tres tecnologías claves: HTML para la estructura de los documentos, CSS para el formato y estilo, y JavaScript para el comportamiento, interacciones y animaciones. Para este tipo de aplicaciones, la fragmentación sigue siendo un problema, pero suele ser menor que en el resto [28].

Las aplicaciones creadas bajo esta estrategia, están diseñadas para ser independientes de la plataforma y de los dispositivos, capaces de ser ejecutadas en cualquier teléfono inteligente o tableta. Normalmente, estas aplicaciones son descargadas desde un servidor web central cada vez que es ejecutada [31]. Como desventaja para este tipo de estrategia de desarrollo, se tienen limitaciones en cuanto a ciertas funcionalidades como el control de la cámara y control directo del GPS.

3.2.3. Desarrollo Multiplataforma

Finalmente, las aplicaciones desarrolladas bajo una estrategia multiplataforma combinan el desarrollo nativo con tecnologías web. Esta estrategia permite a los desarrolladores escribir porciones significativas de la aplicación con tecnologías web multiplataforma, manteniendo acceso directo a los API nativos cuando sea requerido [30], lo que permite que utilicen características nativas de los dispositivos como el GPS o la cámara.

Para el desarrollo de este tipo de aplicaciones, existen soluciones en el mercado como Cordova Apache y Appcelerator, las cuales proveen una interfaz que permite seleccionar capacidades de los dispositivos móviles que son consistentes a través de los diferentes sistemas operativos [30]. La estrategia multiplataforma trata de conseguir y unificar lo mejor de las dos estrategias anteriores, reduciendo tiempos de desarrollo y proveyendo menos limitaciones en cuanto a las funcionalidades de la aplicación.

Para este trabajo, se decidió utilizar la estrategia de desarrollo híbrida, ya que permite la implementación de la aplicación a una gran variedad de plataformas con el mismo código fuente, lo que disminuye la repetición de código, el tiempo de desarrollo, mantenimiento e incrementa la productividad; sin perder funcionalidades nativas como el control de unidades GPS y considerando que era la que mejor se adaptaba a los recursos disponibles.

3.3. Soluciones Multiplataforma (*Frameworks*)

Para el desarrollo de aplicaciones multiplataforma se utilizan *frameworks* de aplicaciones, también conocidos como entornos de desarrollo de aplicaciones, los cuales tienen dos principales objetivos. El primer objetivo es proveer un navegador embebido, como lo es Web View en Android, que ejecuta el código Web de la aplicación. Y el segundo es suministrar “puentes” que permitan al código Web acceder a los recursos locales del dispositivo [32].

De esta manera, estos entornos de desarrollo permiten a los desarrolladores la creación de aplicaciones móviles portables y la conversión de aplicaciones Web existentes, a aplicaciones móviles [32]. Sus plataformas objetivo incluyen sistemas operativos móviles como Android, iOS, Windows 10, entre otros; y entre las herramientas tecnológicas más populares para esta estrategia de desarrollo se encuentran Apache Cordova y Appcelerator Titanium.

Apache Cordova es un *framework* de desarrollo móvil de fuente abierta que permite el uso de tecnologías Web estándares como HTML5, CSSS3 y JavaScript, para el desarrollo de aplicaciones móviles multiplataforma. De esta manera, las aplicaciones son ejecutadas dentro de

envoltorios, en inglés son conocidos como *wrappers*, que están dirigidos a cada plataforma de manera individual, y se basan en API estandarizadas para acceder a las capacidades de cada dispositivo como sensores, datos, estado de red, entre otros [33].

Como contraste, Appcelerator Titanium es un ambiente de desarrollo de fuente abierta, que permite la creación de aplicaciones nativas a través de diferentes dispositivos móviles, así como el desarrollo de aplicaciones multiplataforma.

Para este trabajo especial de grado, se decidió utilizar el entorno de desarrollo Apache Cordova, por la gran documentación que posee y los recursos disponibles; así como por su amplia aceptación en la comunidad de desarrollo móvil.

3.3.1. Apache Cordova

Apache Cordova, anteriormente conocido como PhoneGap, es un entorno de desarrollo móvil de fuente abierta, el cual permite el uso de tecnologías Web como HTML5, CSS3 y JavaScript, para el desarrollo de aplicaciones móviles multiplataforma. Fue llamado originalmente PhoneGap y fue creado por la compañía Nitobi Software, la cual fue luego adquirida por Adobe en octubre del 2011 [34]. En la actualidad, el desarrollo se realiza en el proyecto Apache Cordova de la Apache Software Foundation [35]. A continuación, se describirá de forma más detallada la arquitectura de Apache Cordova.

3.3.1.1. Arquitectura

Existen varios componentes que forman la arquitectura de una aplicación construida sobre Cordova. En la Figura 5 se describe de forma gráfica cuáles son y cómo se ubican cada uno de estos componentes en el sistema de Apache Cordova.

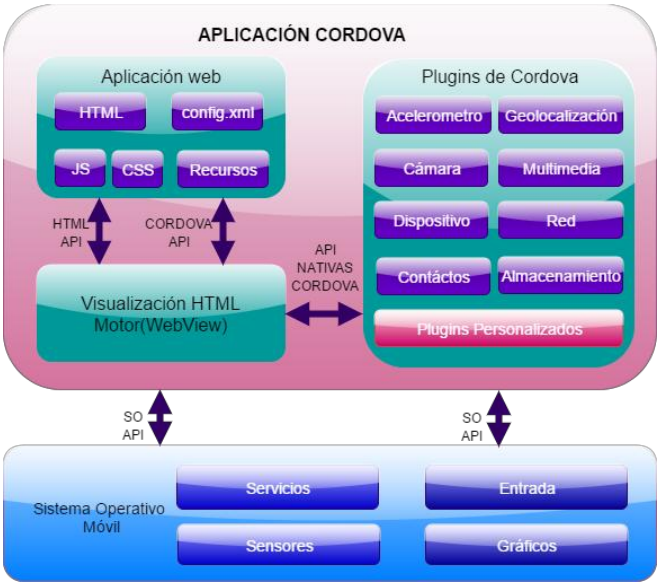

***Figura 5*. Arquitectura de una aplicación Cordova. Adaptado de [33]**

3.3.1.2. WebView

La WebView, habilitada por Cordova, se encarga de proveer la interfaz de usuario de la aplicación. Asimismo, y en algunas plataformas, puede también ser un elemento dentro de una aplicación multiplataforma más compleja que mezcla la WebView con componentes nativos, como es el caso del proyecto que describe este trabajo [33]. De esta forma, aplicaciones desarrolladas con Cordova son implementadas generalmente dentro de una WebView, que funciona como un contenedor, dentro de una plataforma móvil nativa, como Android, iOS y Windows Phone.

3.3.1.3. Web App

La capa de Web App es donde reside el código de la aplicación. La aplicación en sí misma es implementada como una página web, que referencia archivos de tipo CSS, JavaScript, imágenes, archivos de media y otros recursos necesarios para su ejecución [33]. La aplicación se ejecuta en una WebView dentro de un ambiente de aplicación nativo, la cual puede ser distribuida en las diferentes tiendas de aplicaciones como el Play Store, de Android.

Dentro de la Web App, existe un archivo crucial llamado *config.xml*, el cual provee información sobre la aplicación y especifica parámetros que afectan cómo funciona ésta; por ejemplo, si responde o no a cambios de orientación [33].

3.3.1.4. Plugins

Los *plugins*tienen un papel sumamente importante dentro del ecosistema Cordova, ya que son los que proveen una interfaz para que Cordova y los componentes nativos se puedan comunicar los unos con los otros, además de proveer también enlaces a las API de los dispositivos móviles estándar[33]. De esta manera, permite la invocación de código nativo desde JavaScript.

El proyecto de Apache Cordova mantiene un conjunto de *plugins* llamados Plugins Núcleo. Estos son los *plugins* que proveen a la aplicación el acceso a las capacidades del dispositivo móvil [33] como los son la batería, cámara, contactos, unidades de ubicación, entre otros. Además de los *plugins* núcleo, Cordova también cuenta con *plugins* de terceras partes que proveen enlaces adicionales a características no necesariamente disponibles en todas las plataformas móviles.

Dado que Apache Cordova no provee de herramientas para el desarrollo de Interfaces de Usuario o entornos de desarrollo para los componentes web, es necesario utilizar algún tipo de herramienta o *framework*que provea estos mecanismos y servicios optimizados para el desarrollo de aplicaciones móviles.

3.3.2. Ionic Framework

Ionic Framework es un SDK de fuente abierta y un entorno de desarrollo de aplicaciones multiplataforma, que fue desarrollado por Drifty en noviembre del 2013 [36]. Ionic permite a los desarrolladores la construcción de aplicaciones móviles de alta calidad y con buen rendimiento,

utilizando tecnologías web familiares entre las que se encuentran HTML5, CSS3 y JavaScript [37]; en otras palabras, su intención es enfocarse en el desarrollo de las interfaces de la aplicación y cómo los usuarios interactúan con ésta, con el objetivo de simplificar el proceso de desarrollo que involucra el *front-end* de la aplicación móvil.

Bajo este contexto, Ionic provee numerosos componentes que permiten a los usuarios deslizarse entre las diferentes interfaces que componen una aplicación móvil, entre los que se encuentran botones, pestañas y menús. Ionic también incluye una biblioteca de *fonts* que permite la utilización de iconos en las aplicaciones y aumenta la experiencia de los usuarios de la aplicación.

Actualmente, Ionic está construido sobre AngularJS, el cual es el que se encarga de proveer funcionalidades esenciales como el enrutamiento, enlace de dos vías (conocido en inglés como *two way data binding*) [37] y servicios de validación en cuanto a los formularios y los datos de entrada. Por esta razón, se dice que una aplicación Ionic requiere de AngularJS con el fin de utilizar todo su potencial [37]. Sin embargo, trabajar con AngularJS es opcional, y es posible desarrollar en Ionic sin utilizar AngularJS. Asimismo, las aplicaciones desarrolladas utilizando Ionic son desplegadas a través de Apache Cordova.

4. TECNOLOGÍAS WEB

El concepto de la web como idea, tiene sus inicios en los años 40s, cuando Murray Leinster escribió en su obra “A Logic Named Joe” un mundo en el que las computadoras se encontraban en cada hogar, proveyendo el acceso a un dispositivo central del que podían recuperar información [38]. Sin embargo, no fue hasta 1989 que esta idea se convierte en una realidad, y en una entidad que siguió creciendo de manera exponencial, hasta finalmente convertirse en la World Wide Web que conocemos hoy en día.

Tom Berners-Lee, conocido como el padre fundador de la web, delineó el concepto de una plataforma computacional que permitiría facilitar la colaboración entre investigadores que estaban localizados en diferentes partes del mundo, a través de una base de datos centralizada [38]. De esta forma, el lenguaje HTML tuvo su génesis, y rápidamente se convirtió en el bloque de construcción básico de la World Wide Web, lo que aún ahora se mantiene; convirtiéndolo en el núcleo de su código e infraestructura.

Poco después de su nacimiento, la World Wide Web comenzó a ganar gran popularidad con el desarrollo del primer navegador web, lo que introdujo nuevos tipos de usuarios diferentes a los que hasta ese momento existían y, a consecuencia de esto, introdujo diferentes tipos de necesidades que necesitaban ser satisfechas. Por esta razón, los objetivos de la web cambiaron drásticamente de los previstos inicialmente hace más de 20 años [38], lo que dio pie al nacimiento de nuevas demandas con respecto a lo que los usuarios querían ver en las páginas web. Como resultado, nuevas tecnologías enfocadas específicamente a la web y al desarrollo en la web, comenzaron a emerger. En esta sección, se describirán algunas de las tecnologías más impacto tienen en lo referente al proyecto descrito en este trabajo especial de grado.

4.1.HTML5

HyperText Markup Language, o HTML por sus siglas en inglés, y traducido al español como lenguaje de marcado de hipertexto, es el bloque de construcción más básico y el lenguaje de marcado más utilizado en la web. HTML es el lenguaje que se encarga de describir y definir el contenido, y la estructura de una página web. En otras palabras, HTML es el código utilizado para estructurar una página web y su contenido, por ejemplo, como un conjunto de párrafos, listas, imágenes o tablas de datos, entre otros elementos [39].

HTML fue creado por Berners-Lee a finales del año 1991, pero no fue hasta la creación de HTML 2.0 que se convierte es un estándar con la primera especificación HTML publicada en 1995. En la actualidad, HTML se encuentra en la versión 5, por lo que es llamado HTML5, y su especificación fue publicada en el año 2012 [40].

Es importante puntualizar que HTML no es un lenguaje de programación, sino un lenguaje de marcado. De esta manera, es el que se encarga de decirle al navegador cómo mostrar las páginas web y su contenido a los usuarios. Con este fin, HTML consiste en una serie de elementos que son utilizados para contener o encerrar diferentes partes del contenido [39] que se encuentra alojado en las páginas web. En HTML, un elemento está compuesto por cuatro partes principales: las etiquetas de apertura y de cierre, el contenido, y el elemento., las cuales pueden observarse de manera gráfica en la Figura 6:

***Figura 6*. Elemento HTML. Adaptado de [39]**

Las etiquetas de apertura son las que consisten en el nombre del elemento, en este caso la “p” designa un elemento párrafo, y determinan dónde comienza este elemento; mientras que las etiquetas de cierre son las que establecen dónde termina. Por otro lado, el contenido del elemento es lo que se encuentra entre la etiqueta de apertura y la de cierre; y, finalmente, el elemento HTML es el que engloba todos los componentes antes mencionados. De esta manera, las etiquetas que contienen elementos HTML proveen de instrucciones de cómo la información será procesada y mostrada [41].

Los elementos en HTML pueden tener atributos. Un atributo es información adicional sobre el elemento que no aparece ante el usuario de forma explícita, y está compuesto por un nombre y un valor. En la Figura 7 puede verse de forma gráfica cómo corresponden los atributos a los elementos HTML.

***Figura 7*. Elemento HTML con atributo. Adaptado de [39]**

En este caso, el atributo que se agrega al elemento párrafo es la clase; la palabra “class” en este ejemplo representa el nombre del atributo y su valor está representado por la palabra “hello”.

4.1.1. Ejemplo de un documento HTML5

Un documento HTML está formado por dos secciones principales: la cabeza o *head*, y el cuerpo o *body*. El *body* es donde se define el contenido que los usuarios verán a través del navegador, mientras que la sección *head* contiene información acerca del propio documento HTML, que generalmente es invisible al usuario. Las secciones de *body* y *head* son anidadas a un elemento HTML padre.

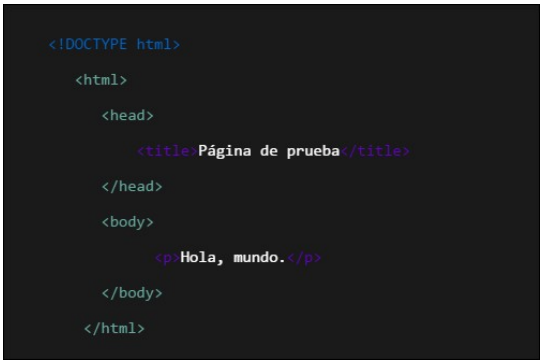

***Figura 8*. Ejemplo de documento HTML**

En la Figura 8, puede verse un ejemplo sencillo de un documento HTML. A continuación, se describirá este documento de forma más detallada:

La declaración `<!DOCTYPEhtml>` es lo primero que debe encontrarse en un documento HTML; esto se debe a que es lo que instruye al navegador de la versión HTML en la que el documento está escrito y permite su correcto despliegue.

El elemento `<html>` es el que engloba o abarca todo el contenido de la página web en su totalidad. También es conocido como el elemento raíz [39].

El elemento `<head>` es el que contiene información sobre el propio documento HTML. En esta sección es donde se ubica la descripción de la página, el título, y referencias a archivos CSS, entre otros.

El elemento `<body>` es donde se ubica el contenido a mostrar al usuario; desde párrafos, textos e imágenes, hasta enlaces, tablas y videos.

4.2.DOM

El Document Object Model, o DOM por sus siglas en inglés, es una interfaz de programación de aplicaciones, o API, para documentos HTML, XML y SVG. El DOM provee una representación estructurada del documento en forma de un árbol [42] y define métodos que permiten el acceso a este árbol, con el fin de proveer a los desarrolladores una manera de modificar la estructura, el estilo y el contenido de los documentos para la creación de páginas web dinámicas.

En otras palabras, el DOM es el modelo de un documento HTML que es cargando en el navegador para los usuarios, que está representado en forma de árbol de nodos, donde cada nodo simboliza un elemento del documento. En la Figura9 puede verse de forma gráfica el ejemplo descrito en la sección anterior de un documento HTML, representado como un árbol de nodos.

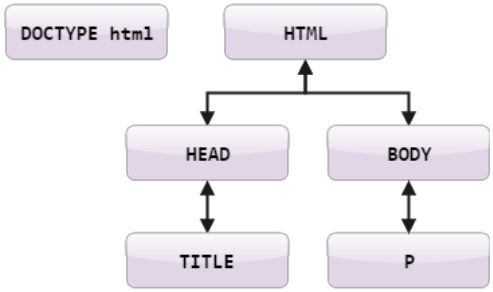

***Figura 9*. Ejemplo de árbol DOM**

Como puede verse en la Figura 9, el DOM provee una representación del documento como un grupo estructurado de nodos y objetos [42], y a su vez proporciona a los desarrolladores propiedades y métodos para su acceso. Además, los nodos que forman el árbol DOM permiten el registro de manejadores de eventos. De esta manera, y una vez un evento es disparado, los manejadores de eventos son ejecutados. El DOM define un evento como una acción producida por el usuario o por el mismo navegador, por ejemplo, si un usuario presiona un enlace, se genera un evento tipo "*onclick*". Un manejador de eventos captura esta información, y permite el manejo de este evento de varias maneras. Esencialmente, el DOM permite la conexión o el enlace entre las páginas web y los lenguajes de programación [42] para la creación de páginas dinámicas y que reaccionan ante acciones de los usuarios.

El DOM es la API más utilizada en la web, debido a que permite al código que se ejecuta en un navegador, acceder e interactuar con los nodos del árbol. Originalmente, DOM no fue creado como una especificación, sin embargo, con la aparición de lenguajes de scripting como JavaScript, fue necesaria la definición de su especificación, la cual ha sido liderada por la W3C. En la actualidad, la versión 4 de la especificación del DOM está siendo preparada para su publicación [43].

4.3.CSS

Cascading Style Sheets, o CSS, es un lenguaje de hojas de estilo en cascada que es utilizado para dar estilo y establecer el formato de las páginas o aplicaciones web. De esta manera, son las que

se encargan de alterar el tipo de letra, color, tamaño, opacidad y agregar animaciones a un documento HTML, entre otras características. Fue propuesto originalmente en 1994 por Håkon W Lie[44], quien trabajaba con Berners-Lee por ese tiempo, y fue finalmente publicado en 1996 por la W3C [45] como CSS nivel 1.

La forma en la que CSS afecta la presentación de los documentos HTML es a través del navegador. Los navegadores web aplican reglas CSS a un documento y, en consecuencia, afectan la forma en la que estos se presentan ante el usuario. Las reglas CSS están formadas de la siguiente manera [46]:

***Figura 10*. Ejemplo de declaración CSS. Adaptado de [46]**

En la Figura 10 puede verse una sencilla regla CSS, la cual consiste en dos partes principales: un selector, que en este caso sería “H1”, y una declaración, que en la figura correspondería con “color: pink” y la cual está rodeada de llaves. A su vez, una declaración está compuesta de dos elementos: una propiedad (“color”) y un valor (“pink”) [47]. El lenguaje CSS permite la agrupación de declaraciones para un selector, por lo que un selector puede poseer numerosas declaraciones, y una hoja de estilos CSS puede poseer numerosos selectores. El conjunto de reglas CSS contenidas dentro de una hoja de estilo determina la presentación final de la página web [46].

4.3.1. Contención en HTML

Para que las hojas de estilo puedan influenciar la presentación de los documentos HTML a través de reglas CSS definidas, debe existir una manera para que el navegador conozca que estas reglas CSS existen en primer lugar. Puede haber tres formas diferentes en la que un documento HTML y una hoja de estilo se enlazan o combinan: usando un elemento HTML de enlace hacia una hoja de estilo externa, un elemento *style* dentro de la sección *head* del documento HTML y un atributo llamado “style” dentro de algún elemento HTML del documento.

En la figura 10 puede verse de forma gráfica cómo se combina HTML, DOM y CSS para presentarle al usuario la página web final.

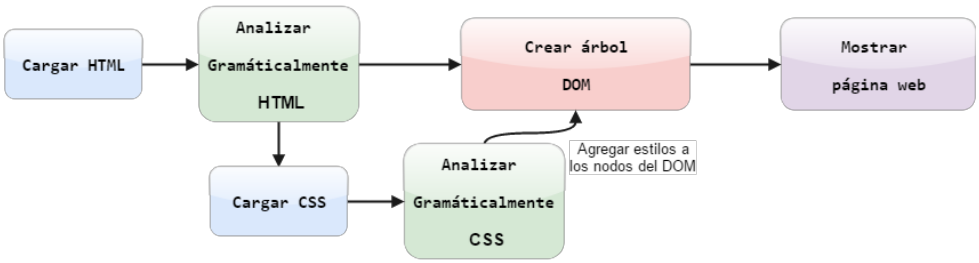

***Figura 11*. Combinación de HTML y CSS en el DOM de una página web. Adaptado de [46]**

4.4. JavaScript

JavaScript, a veces también llamado JS, es un lenguaje de programación de scripting interpretado, basado en objetos, débilmente tipado que puede realizar cómputo y manipular objetos computacionales en un ambiente anfitrión. En la actualidad, JavaScript es mejor conocido como el lenguaje de scripting de la web. Sin embargo, sigue siendo muy utilizado en otros ambientes no dependientes de navegadores web [48], ya que es un lenguaje bastante poderoso y con grandes capacidades. Asimismo, JavaScript es un lenguaje basado en prototipos y es multiparadigma, lo que lo convierte en un lenguaje dinámico que soporta estilos de programación orientada a objetos, imperativa y funcional.

Es importante destacar que JavaScript es mayormente ejecutado del lado del cliente; por lo que es sumamente utilizado en el diseño de aplicaciones o páginas web, definiendo su comportamiento de acuerdo a la ocurrencia de algún evento definido por el DOM. Así como HTML representa la estructura y CSS la presentación de una página web, JavaScript representa el comportamiento.

JavaScript fue creado por Brendan Eich en 1995, mientras trabajaba como ingeniero en la compañía Netscape. Fue liberado al público a principios de 1996, junto con el navegador Netscape 2, lo que influyó directamente con la popularidad de la web. Originalmente, JavaScript tenía como nombre LiveScript. Sin embargo, y tratando de capitalizar la gran popularidad del lenguaje de programación Java, fue renombrado JavaScript, aun cuando ambos lenguajes tienen muy pocas cosas en común [48]. Unos meses después de su creación, Netscape le entregó JavaScript a Ecma International, una organización de estandarización europea, para el desarrollo de su estandarización y especificación, que fue llamada ECMA-262.

A diferencia de muchos otros lenguajes de programación, JavaScript no maneja los conceptos de datos de entrada (*input*) o datos de salida (*output*), ya que fue diseñado para ser ejecutado como un lenguaje de scripting en un ambiente anfitrión, y es la responsabilidad de los ambientes anfitriones de proveer mecanismos de comunicación externa [49]. En este sentido, y en contexto de los navegadores web, un programa de scripting escrito en JavaScript es ejecutado por los navegadores web cuando una página es descargada o en respuesta a un evento [50]. Sin embargo, y como se dijo con anterioridad, JavaScript también se encuentra en el código núcleo de otras aplicaciones, como por ejemplo Adobe Photoshop, Node.js y Apache CouchDB.

Para que un navegador pueda ejecutar código escrito en JavaScript, necesita un motor JavaScript que se encargue de la interpretación. Entre los más populares de la actualidad, se encuentra Spider Monkey de Mozilla, V8 de Google, el cual también es utilizado por Opera, JavaScript Core de Apple Safari y Chakra es el utilizado por Internet Explorer [48]. Además de navegadores web, otra aplicación popular que se basa en código JavaScript es el servidor web. Un servidor web JavaScript se encarga de exponer los objetos que representan peticiones y respuestas HTTP a los ambientes anfitriones, para luego manipularlos por un programa JavaScript con el objetivo

de generar páginas web de forma dinámica. Un ejemplo de un servidor web basado en JavaScript es Node.js.

4.5. AngularJS

AngularJS es un *framework* JavaScript del lado cliente para el desarrollo de páginas web dinámicas. Permite el uso de HTML y provee la extensión de su sintaxis, con el objetivo de expresar los componentes de la aplicación de forma clara y sucinta [51]. Creado en el 2009 por Miško Hevery y Adam Abrons, AngularJS es un proyecto de Google de fuente abierta y bastante popular para desarrolladores de aplicaciones web.

Promoviendo una experiencia de desarrollo web de alta productividad, AngularJS está construido sobre la creencia de que la programación declarativa es la mejor opción para construir las interfaces de usuario, mientras que, por otro lado, la programación imperativa se adapta mejor y es preferible de implementar en la lógica de negocio de una aplicación.

Por esta razón, AngularJS empodera el lenguaje HTML al extender su sintaxis; lo que resulta en el desarrollo de componentes de aplicación expresivos, reusables y mantenibles. En este sentido, AngularJS también provee una clara separación entre las capas principales que conforman una aplicación AngularJS; la vista, el modelo y el controlador, e introduce nuevos términos igual de importantes como los son las directivas, los filtros y los servicios. En consecuencia, se dice que AngularJS sigue un patrón arquitectónico llamado MVC (Modelo-Vista-Controlador) que permite la separación de las unidades lógicas de una aplicación [51], lo que es particularmente pertinente en el desarrollo de aplicaciones con grandes cantidades de código. El patrón de arquitectura MVC que provee AngularJS divide una aplicación en tres partes modulares que serán detalladas con mayor precisión.

4.5.1. La Vista

Las vistas en AngularJS, también llamadas plantillas, están escritas enteramente en HTML. De esta manera, una vista es la interfaz de usuario con la que el usuario interactúa [52]. Las vistas son generadas dinámicamente, aprovechando los mecanismos de directivas que permiten la extensión de la sintaxis de HTML; en consecuencia, AngularJS permite realiza tareas de lenguajes de programación como lo es iterar sobre un arreglo o evaluar una expresión condicionalmente [52]. En este sentido, las vistas son las que manejan la estructura del documento y se encargan de enviar información al controlador, así como generar contenido basado en el modelo de la aplicación.

4.5.2. El Controlador

Detrás de la vista, se encuentra el controlador [52]. El controlador es la capa que contiene toda la lógica del negocio presentada por la vista y realiza acciones como recuperación de datos y toma decisiones sobre cómo presentar el modelo definido por la aplicación. La conexión entre la vista y el controlador es dada por un objeto que define AngularJS llamado `$scope` Con el objeto

`$scope`, los controladores establecen los estados iniciales de la aplicación y la comunicación entre la vista y el modelo.

4.5.3. El Modelo

En AngularJS, un modelo es un objeto JavaScript llamado Plain-Old-JavaScript-Object (POJO), y representa los datos de la aplicación. Al ser implementado como un objeto JavaScript, les facilita a los desarrolladores su dominio y uso. En general, todo dato que se le es presentado al usuario a través de alguna interfaz de la aplicación AngularJS, es entregado por el modelo.

4.6.Node.js

Node.js es una plataforma de software que es utilizada para la construcción y el desarrollo de aplicaciones de red escalables. Node.js utiliza un modelo de entrada y salida sin bloqueos, también llamado E/S sin bloqueo, y es impulsada por eventos (en inglés conocido como *event-driven*), lo que la convierte en una plataforma ligera y eficiente, perfecta para el desarrollo de aplicaciones de tiempo real, que utilicen datos de forma intensiva, y que sean ejecutadas a través de dispositivos distribuidos. [53]

Fue creada en 2009 por Ryan Dahl y patrocinada por Joyent, la compañía en la que Dahl trabajaba para ese momento. Aunque inicialmente sólo tenía soporte para Linux, con el pasar del tiempo ha ido evolucionando como un ambiente de ejecución JavaScript multiplataforma para el desarrollo de una gran variedad de aplicaciones. Node.js permite la ejecución del código JavaScript debido a que está construida sobre el motor de Google Chrome llamado V8. El V8 es un motor de ejecución de código abierto desarrollado por Google en el 2008 y escrito en C++, que se encarga de compilar código JavaScript a código máquina nativo en vez de interpretarlo a tiempo real [54].

Node.js permite la creación de servidores web y el desarrollo de herramientas de red usando JavaScript y una colección de módulos que manejan varias funcionalidades núcleo; en otras palabras, permite la ejecución de JavaScript en el lado del servidor. Los módulos que Node.js maneja son los que proveen mecanismos de E/S del sistema de archivos, conexiones de redes, data binaria, funciones de criptografías y otras funciones núcleo.

Bajo alta carga y alta concurrencia, Node.js mantiene alto rendimiento y baja latencia. Esto lo logra por tres importantes factores que son claves en el diseño de Node.js: el motor de JavaScript V8 que es optimizado por Google, quien continúa trabajando en aumentar su rendimiento; el uso de eventos en vez de hilos, que lo hacen liviano y con un rendimiento mayor en comparación a otros diseños monolíticos; y su arquitectura de contención, lo que la simplifica su migración a la nube y otros servicios [55].

5. NoSQL

Desde su creación, los sistemas manejadores de base de datos relacionales, también conocidos como SMDBR por sus siglas en español, han sido el método de almacenamiento de datos

escogido para el desarrollo de aplicaciones que necesitan almacenar datos de forma persistente. Dominando el mercado desde los años 80 [56], los SMDBR son reconocidos por implementaciones como la base de datos Oracle, MySQL, PostgresSQL y Microsoft SQL Server. Sin embargo, en años recientes y en parte por la expansión de la computación en la nube [57], conocido en inglés como *cloud computing*, y el uso masivo de dispositivos móviles, se ha incrementado la necesidad de procesar mayores volúmenes de datos, a mayores velocidades y considerando una mayor variedad de tipos de datos [58]. En este sentido, para compañías como Amazon, Facebook y Google, entre muchas otras, la web se ha convertido en un repositorio de datos distribuido, en el cual el procesamiento de estos datos a través de los SMDBR ha quedado corto y ha demostrado no ser suficiente [58], tanto por problemas en la modelación de los datos como por las restricciones a nivel de escalabilidad horizontal a través de varios servidores y grandes cantidades de datos [57]. En este sentido, salen a la luz dos factores que están directamente relacionados con lo declarado anteriormente. El primero involucra el crecimiento exponencial de los datos generados por usuarios, sistemas y sensores, y la concentración de grandes partes de estos volúmenes en grandes sistemas distribuidos como Amazon, Google y otros servicios en la nube. Mientras que el segundo factor involucra la creciente independencia y complejidad de los datos en consecuencia de la web, redes sociales y acceso abierto, y estandarizado a fuentes de datos por un largo número de diferentes sistemas y dispositivos [56] en la web. En consecuencia, y para satisfacer las nuevas necesidades que procesar grandes volúmenes de datos en sistemas distribuidos genera, aparecen las bases de datos NoSQL.

Las bases de datos NoSQL, por sus siglas en inglés Not Only SQL [56], se refiere a un grupo ecléctico y cada vez más familiar de sistemas manejadores de datos no relacionales; donde las bases de datos no están principalmente construidas sobre tablas, y que generalmente no hacen uso del lenguaje SQL para la manipulación de datos [56]. Los sistemas manejadores de bases de datos NoSQL permiten trabajar sobre grandes volúmenes de datos cuando la naturaleza de estos datos no requiere de un modelo relacional. De esta manera, las NoSQL son sistemas de base de datos distribuidos y no relacionales, diseñados para el procesamiento de datos masivo y paralelo a través de variados números de servidores, los cuales no utilizan lenguajes PLSQL/SQL para interactuar con los datos [56].

Las bases de datos NoSQL se han vuelto particularmente relevantes en la actualidad, en lo concerniente con el desarrollo de aplicaciones móviles. Parte de la razón de esto se debe a que los modelos de base de datos relacionales no son lo suficientemente, o idealmente, dinámicos para las necesidades de estas aplicaciones como consecuencia de sus esquemas de datos fijos y estrictos. Esta particularidad de los SMDBR causa problemas en los entornos móviles dado que existen muchos requerimientos situacionales comunes a las aplicaciones móviles que deben ser contemplados, y los cuales los desarrolladores solucionan haciendo cambios que terminan consumiendo mucho tiempo de desarrollo por las contantes ediciones en el esquema de base de datos [59]. Otro de los problemas que pueden encontrarse al implementar una base de datos relacional en los entornos móviles, es que éstas no están construidas para manejar todos los diferentes casos de uso que las aplicaciones necesitan, debido a consideraciones que involucran

la fragmentación, tipo de dispositivo, sistema operativo móvil, plataforma y hasta los tipos de datos que son almacenados; así como las versiones de estas plataformas y la posibilidad de utilizar software no actualizado.

En contraste, las bases de datos NoSQL están diseñadas para manejar las necesidades dinámicas de las aplicaciones móviles [59], en parte porque no utilizan esquemas de datos fijos. Esto permite que realizar cualquier cambio necesario a la base de datos, no demande tiempos de edición prolongados y una reestructuración del modelo de datos. Además, son particularmente útiles cuando se utilizan tipos de datos diferentes a los habituales, como lo son los datos de geolocalización.

Otra de las preocupaciones propias a las aplicaciones móviles, que también son satisfechas a través de las bases de datos NoSQL, es la sincronización de los datos móviles y el desarrollo de aplicaciones con capacidades de trabajar sin conexión, en inglés conocidas como *offline applications*. Una aplicación sin conexión permite al usuario seguir utilizando ciertos módulos de la aplicación sin necesidad de tener conexión internet, y en cuanto esté establecida esta conexión, sincroniza la información generada por el usuario en el dispositivo móvil al servidor de base de datos. Como consecuencia de esto, las aplicaciones móviles que siguen este modelo proveen una mejor experiencia de usuario y cada vez se hacen más populares, al evitar una dependencia a la red y dando capacidades de desconexión, y sincronización.

Este trabajo especial de grado, implementa una solución NoSQL por las razones antes dichas, escogiendo la base de datos CouchDB para el servidor web, y PouchDB, una implementación de CouchDB escrita en JavaScript, en los dispositivos móviles, ya que ambas fueron diseñadas con el propósito principal de proveer sincronización de datos.

5.1. CouchDB

CouchDB es una base de datos NoSQL basada en documentos de código abierto, que provee sincronización fluida y que permite escalar desde un entorno móvil hasta un entorno de Big Data. Posee una API HTTP/JSON intuitiva y diseñada para proporcionar confiabilidad [60]. CouchDB define el Protocolo de Replicación Couch, en inglés llamado Couch Replication Protocol, que es implementado por ambientes de computación que van desde clúster de servidores distribuidos, a dispositivos móviles y navegadores web.

CouchDB comenzó a ser implementada en el 2005 en IBM como un proyecto para ayudar con la replicación de datos sin conexión, y se convirtió en un proyecto independiente de código abierto en 2008 [61] cuando fue entregado a la Apache Software Foundation.

Uno de los objetivos de diseño de CouchDB es en tratar de proveer facilidad de uso; por esta razón, aprender CouchDB y entender sus conceptos núcleos debería sentirse natural para cualquier desarrollador que tenga experiencia con trabajar en la web [62]. Con una arquitectura interna tolerante a fallos, provee una forma en la que los problemas que raramente ocurren no se contagien en forma de cascada a través del servidor, si no que permanezcan aislados en

solicitudes individuales [62]. De esta manera, el diseño de CouchDB emplea grandes característicos de la arquitectura web y el concepto de recursos, métodos y representaciones, proveyendo además formas poderosas de consultar, mapear, combinar y filtrar sus datos [62]. Anidado a lo anterior la tolerancia a fallos, escalabilidad extrema y replicación incremental, CouchDB define es una de las mejores opciones en cuanto a bases de datos documentales.

5.1.1. Almacenamiento de documentos

Un servidor CouchDB hospeda bases de datos que a su vez contienen documentos. Los documentos en una base de datos CouchDB deben poseer un único identificador o nombre que lo diferencie de todos los demás documentos almacenados en la misma base de datos. Además de esto, CouchDB provee de los API necesarios para la lectura y la actualización (agregación, edición y eliminación) de los documentos de la base de datos [63]. De esta manera, un documento en CouchDB es la unidad primaria de datos y consiste en cualquier número de campos y archivos adjuntos.

Es importante notar que el modelo de actualización de documentos que proporciona CouchDB es optimista; es decir, las ediciones y actualizaciones de los documentos son realizadas por las aplicaciones cuando cargan algún documento, le aplican cambios y lo guardan nuevamente en la base de datos. Por esta razón, si otro usuario con la misma aplicación hace cambios al mismo documento, pero lo guarda primero, el segundo usuario recibe un error de conflicto. Para resolver esta situación, la última versión del documento debe cargarse, y los cambios deben ser aplicados nuevamente para que pueda ser actualizado [63].

Para CouchDB, las operaciones que involucran las actualizaciones de los documentos o son exitosas por completo o fallan por completo. Es decir, una base de datos CouchDB nunca almacena documentos parcialmente editados o actualizados. Asimismo, los documentos son indexados en Árboles-B, en inglés llamados B-trees, por sus nombres y un identificador de secuencia. De esta manera, cada nueva actualización a una base de datos genera un nuevo número secuencial, y el identificador de secuencia es el que permite encontrar los cambios realizados en la base de datos de manera incrementar [64].

5.1.2. Modelo de Vista

El modelo de vista que CouchDB integra a su arquitectura, es el que permite agregar estructura a los datos no estructurados o semi-estructurados que almacena. En otras palabras, las vistas son el método que se utiliza para agregar e informar sobre los documentos en una base de datos, y son definidas bajo demanda para agregar y unir documentos [64]. Las vistas en CouchDB son construidas de forma dinámica, por lo que no afectan el documento subyacente, así que puede implementarse diferentes representaciones de vistas de los mismos datos.

Las vistas en CouchDB son definidas dentro de un documento de diseño. Los documentos de diseño son un tipo especial de documentos que contiene código de aplicación [65] que pueden ser escritos en JavaScript.

5.1.3. Actualizaciones distribuidas y replicación

CouchDB se destaca entre las bases de datos NoSQL porque es un sistema de base de datos distribuido basado en *peers*. De esta manera, provee a los usuarios y servidores con las herramientas necesarias para acceder y actualizar datos compartidos entre varios usuarios, sin la necesidad de que estos deban estar conectados entre sí, ya que los cambios realizados en los documentos de una base de datos CouchDB pueden ser replicados luego de una manera bidireccional.

Para CouchDB el proceso de replicación es incremental. Es otras palabras, si durante la replicación algo ocurre que interrumpe la conexión de red entre las bases de datos siendo replicadas, se retomará la próxima vez que la conexión pueda recuperarse. Además de esto, la replicación de CouchDB utiliza las mismas API que las aplicaciones utilizan [65], lo que hace más sencillo el desarrollo.

Debido a que CouchDB consigue la consistencia eventual entre múltiples bases de datos al utilizar replicación incremental, las bases de datos no necesitan mantenerse en comunicación constante [66], y de esta manera se consigue dar mayor flexibilidad en el desarrollo de aplicaciones móviles.

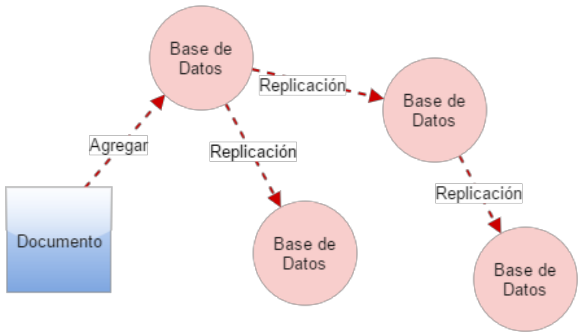

***Figura 12*. Modelo de replicación de CouchDB. Adaptado de [66]**

En la Figura 12 puede verse de forma gráfica cómo funciona la replicación incremental de CouchDB a través de una red. De esta manera, se genera un clúster de bases de datos donde cada nodo es independiente y autosuficiente [66], ya que, debido a la replicación, todas las bases de datos contienen los mismos documentos y datos que las demás.

5.1.4. MapReduce

A diferencia de las bases de datos relacionales, que te permiten ejecutar cualquier tipo de consulta mientras los datos estén estructurados de forma correcta, CouchDB utiliza funciones de mapeo y reducción, conocido en inglés como funciones MapReduce. Estas funciones proveen gran flexibilidad ya que pueden adaptarse a variaciones de la estructura del documento, y la indexación de cada documento puede ser ejecutada de forma independiente y en paralelo [67], lo que reduce los tiempos de respuesta.

A diferencia de las consultas tradicionales a tablas en los SMDBR, las consultas de reducción, o reduce, están basadas en solicitudes de rango simple contra los índices generados por las funciones de mapeo. Estas funciones de mapeo, o map, son ejecutadas una sola vez con cada documento como el argumento de la función. De esta manera, la función puede elegir saltarse un documento o emitir una o más filas de vistas como pares claves/valor. Dado que las vistas de CouchDB son almacenadas como listas que son ordenadas por la clave, la recuperación de los datos desde un conjunto de claves es extremadamente eficiente, incluso cuando hay miles o millones de documentos. Por esta razón, el objetivo principal al construir una función map es definir una indexación que almacena datos relacionados bajo claves cercanas [67].

5.2.PouchDB

PouchDB es una base de datos de código abierto de JavaScript inspirada en Apache CouchDB que está diseñada específicamente para que funcione en el navegador. PouchDB fue creada para ayudar a los desarrolladores web o móviles la construcción de aplicaciones sin conexión, que funcionen tan bien como lo hacen con conexión [68]. De esta forma, permite a las aplicaciones almacenar datos de forma local mientras la aplicación se encuentra sin conexión, para luego sincronizarla con una base de datos CouchDB remota en cuanto la aplicación regrese a estar conectada, manteniendo los datos de los usuarios sincronizados.

CAPÍTULO III MARCO APLICATIVO

En este capítulo se describen las cinco fases que conforman la adaptación de la metodología de desarrollo ágil Mobile-D que se utilizó para la implementación de la aplicación móvil basada en geolocalización, para este trabajo especial de grado. Estas fases fueron: la fase de exploración, inicializar, producción, estabiliza y, finalmente, la fase de prueba.

1. FASE DE EXPLORACIÓN

El principal propósito de la fase de exploración es la planificación y el establecimiento inicial del proyecto que será desarrollado. Es aquí donde se definen los requerimientos, el alcance y el ambiente de desarrollo de la aplicación móvil, lo que permite establecer la base para la implementación controlada del software.

1.1.Definición de requerimientos

En esta sección, se describe de manera general el sistema completo. La aplicación se muestra bajo un contexto de interacción con otros sistemas y se introducen las funcionalidades básicas. También se definen las restricciones de la aplicación y el ambiente de operación.

1.1.1. Perspectiva de la aplicación

La aplicación a desarrollar llamada “AlzRastreo”, estará basada en información geoespacial obtenida mediante los servicios de ubicación integrados en los dispositivos móviles. De esta manera, permitirá dar a conocer la ubicación de aquellas personas con Alzheimer a sus guardianes a través del envío de notificaciones. Para lograr esto, la aplicación define dos tipos diferentes de usuarios: usuarios pacientes y usuarios guardianes, que el usuario debe seleccionar al momento de registrarse. Cada tipo de usuario posee su propio conjunto de funcionalidades; los usuarios guardianes agregan a los usuarios pacientes, quienes a su vez deben aceptar esta invitación. De esta forma, y una vez un usuario paciente acepta a un usuario guardián, se crea una relación entre ambos.

La aplicación móvil obtendrá la ubicación geoespacial de los usuarios pacientes de forma continua, con el fin de notificar esta información a aquellos usuarios guardianes pertinentes, en caso de que el usuario paciente no se encuentre en algún lugar seguro. La forma en la que la aplicación conoce si el paciente está o no en un lugar seguro, es a través de las “Zonas Seguras”. Una Zona Segura es una zona geográfica con un radio específico que demarca una ubicación real en la que los usuarios pacientes se encuentren seguros, y la cual es definida por los usuarios guardianes para sus pacientes (Ej. hogares, trabajos, etc.).

De esta manera, si los usuarios pacientes se encuentran dentro de estas zonas seguras, la aplicación se abstendrá del envío de notificaciones a los guardianes correspondientes. Por el contrario, si los usuarios pacientes salen de los límites de estas zonas seguras, entonces la aplicación se encargará de notificar a los usuarios guardianes de esto. Los usuarios guardianes

también podrán visualizar de forma gráfica y sobre un mapa, la última ubicación almacenada del usuario paciente, y serán capaces de registrar a otros usuarios guardianes a un mismo usuario paciente como forma de compartir el compromiso.

Conjuntamente con la aplicación móvil, se planea el desarrollo de un servidor web que tiene como objetivo establecer una conexión entre la aplicación y la base de datos CouchDB remota, donde los datos generados por los usuarios y almacenados en cada dispositivo de forma local, serán replicados y almacenados en la base de datos remota. De esta manera, la arquitectura que se propone está representada de forma gráfica en la Figura 1.

La aplicación móvil propuesta trabajaría junto con un servidor web que actuaría como intermediario entre la aplicación y la base de datos. Asimismo, entre las funcionalidades del servidor web también se encuentra la autenticación de usuarios, el monitoreo de las bases de datos de los usuarios y el envío de notificaciones a los usuarios pertinentes.

1.1.2. Restricciones

La aplicación “AlzRastreo” está limitada por los servicios de ubicación del dispositivo móvil y la conexión a internet para la replicación de datos a la base de datos remota. Debido a que existen múltiples fabricantes de unidades GPS, la interfaz puede variar con respecto a ellos, de la misma manera con las funcionalidades que estos proveen. Una restricción también es el rango de la unidad GPS con respecto a la localización de los usuarios (túneles, subterráneos, etc.) y la posibilidad de triangulación celular provista por los servicios de Google dependiente de la señal celular.

1.2.Requisitos de Interfaces Externas

En esta sección se detallan algunas entradas y salidas de la aplicación móvil. También describe ciertas interfaces de los usuarios en forma de prototipos básicos, de software y de comunicación.

1.2.1. Interfaces de Usuario

Para un usuario que ingresa a la aplicación por primera vez, la aplicación debe desplegar la interfaz de iniciar sesión. Desde ésta, el usuario puede navegar a la interfaz de registrarse, como puede verse en la Figura 13.

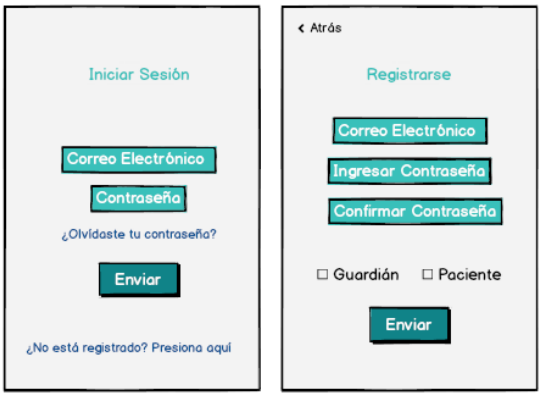

***Figura 13*. Interfaz de iniciar sesión y de registro de usuario**

Si el usuario ha olvidado su contraseña, podrá seleccionar el enlace adecuado y se le presentarán las interfaces mostradas en la Figura 14 para restablecer la contraseña.

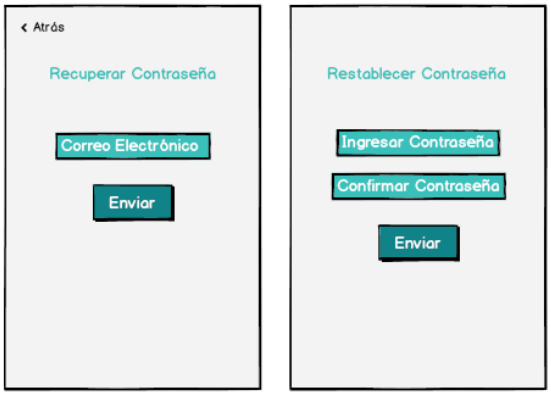

***Figura 14*. Interfaz de recuperar contraseña y de reestablecer contraseña**

Si el usuario ya mantiene una sesión, su pantalla de inicio depende del tipo de usuario que sea. Para los pacientes, la pantalla de inicio muestra un mapa con la ubicación del usuario en ese momento. Existen tres escenarios en este caso, si el paciente se encuentra fuera o dentro de una zona segura, o si no tiene una zona segura asociada. En la Figura 15 puede observarse un bosquejo de la interfaz de inicio para los usuarios pacientes que se encuentra en una zona segura.

***Figura 15*. Interfaz de inicio: paciente dentro de una zona segura**

Los pacientes pueden ver la lista de los guardianes que tienen agregados, además de las invitaciones recibidas de otros guardianes que aún no han sido aceptadas. Los pacientes también pueden ver en forma de lista, las zonas seguras que los guardianes han registrado a su cuenta, como se muestra en la Figura 16. Un paciente puede ver los detalles de sus Zonas Seguras en la vista Ver Zona, la cual lista el nombre de la zona y su descripción.

***Figura 16*. Interfaz de zonas seguras y ver detalle de zona**

Los usuarios guardianes tienen su propia interfaz de inicio diferente a la de los pacientes, y la cual puede verse de forma gráfica en la Figura 17.

***Figura 17*. Interfaz de inicio guardián y de rastrear paciente**

En la interfaz de inicio, los guardianes pueden ver un listado de los pacientes que tienen registrados. En ella, se permite navegar a una interfaz de rastrear paciente, que muestra la ubicación del paciente en esos momentos y la que puede visualizarse gráficamente en un mapa. Asimismo, la aplicación permite la navegación para la gestión de pacientes, la cual es análoga a la gestión de guardianes por parte de los pacientes, como puede verse en la Figura 18.

***Figura 18*. Interfaz de paciente y ver detalle de paciente**

1.2.2. Interfaces de Software

La aplicación móvil se comunica con los servicios de ubicación y localización para obtener la información geográfica del usuario en ese momento y la representación visual a través de un mapa. El sistema operativo en el que la aplicación se ejecuta será Android, el cual provee *frameworks* que serán utilizados en la implementación de la aplicación. La comunicación de la aplicación y la base de datos local y remota involucran operaciones tanto de lectura como de escritura.

1.2.3. Interfaces de Comunicación

La aplicación móvil podrá comunicarse utilizando notificaciones *push* y correos electrónicos hacia otros dispositivos. La aplicación también debe comunicarse con el servidor web para la replicación y la sincronización de los datos entre las diferentes bases de datos.

1.3.Requerimientos Funcionales

En esta sección se describen los requerimientos que especifican todas las acciones esenciales de la aplicación móvil.

1.3.1. Casos de Uso

En esta sección se describen de forma detallada los casos de uso definidos para la aplicación. En la Figura 19 puede verse el diagrama de casos de uso de nivel 0, y en la Figura 20 se muestra el diagrama de casos de uso de nivel 1.

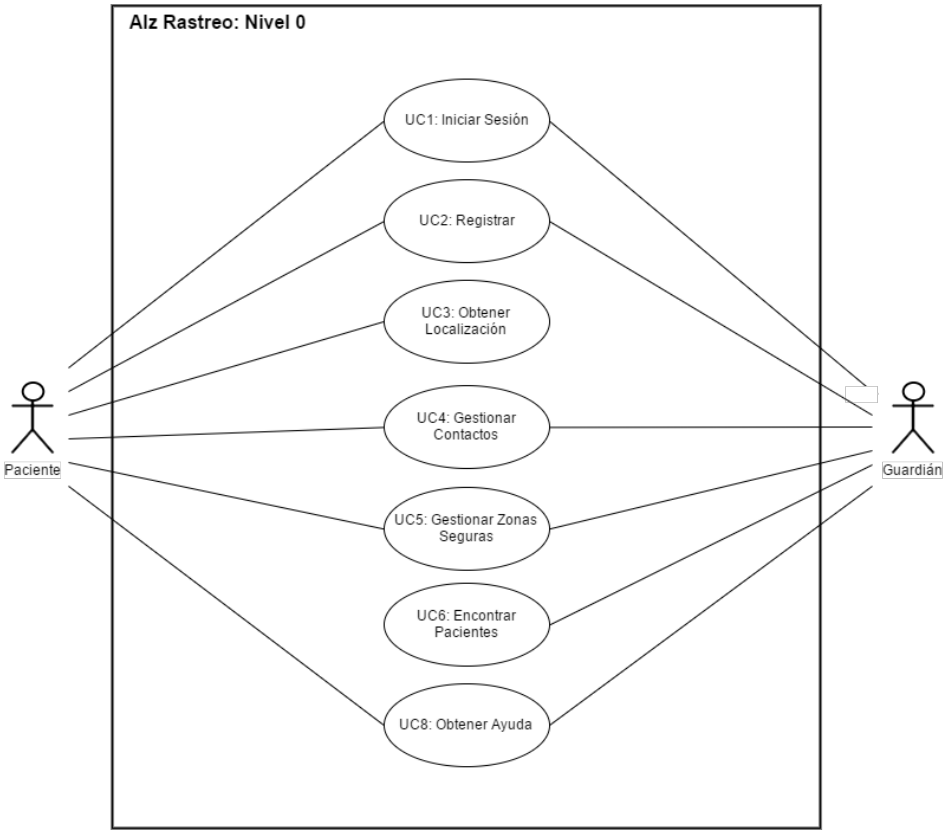

***Figura 19*. Diagrama de casos de uso nivel 0**

***Figura 20*. Diagrama de casos de uso nivel 1**

1.4.Requerimientos No Funcionales

1.4.1. NF1: Rendimiento

- El tiempo de respuesta del sistema después de una operación de entrada no debe superar los 15 segundos.
- La carga de los mapas en el sistema no debe superar los 15 segundos.
- La cantidad de memoria ocupada en el dispositivo móvil no debe ser mayor a los 20MB.

1.4.2. NF2: Usabilidad

- Un usuario debe ser capaz de determinar rápidamente que operaciones son capaces de realizar.
- El sistema debe ser capaz de discernir datos de entradas erróneos en todos los campos necesarios del sistema.

1.4.3. NF3: Disponibilidad

- La aplicación debe estar conectada al Internet.
- El dispositivo móvil debe tener activado los servicios de localización.

1.4.4. NF4: Seguridad

- Los mensajes entre el servidor y la aplicación deben estar encriptados para comunicaciones de inicio de sesión.
- No pueden existir dos cuentas de usuario con la misma cuenta de correo electrónico nombre de usuario.

1.4.5. NF5: Fiabilidad

- El sistema debe ser 90% fiable. Es decir, de cada 10 ejecuciones de la aplicación, 9 no tengan ningún error de compilación.

1.5.Definición del alcance del proyecto

Este proyecto tiene como meta desarrollar una aplicación móvil que asista a aquellos pacientes que padecen de la enfermedad de Alzheimer. Esto se logra obteniendo la información de ubicación del usuario del dispositivo móvil, y notificando a su guardián o familiares de su ubicación, cuando éste salga de una zona geográfica segura, previamente definida.

Los beneficios de esta aplicación móvil son los de proveer mayor independencia, autonomía y, por consiguiente, calidad de vida a aquellas personas que padecen la enfermedad de Alzheimer, al permitirles salir de casa sin preocupar a sus familiares o guardianes. Asimismo, la aplicación también pretende asistir a los familiares y guardianes de estos pacientes, en caso de un posible episodio de desorientación espacial, notificándoles en qué lugar se encuentra la persona enferma y de esta manera evitar potenciales emergencias. La aplicación permitirá asistencia a través del dispositivo móvil.

1.6.Establecer ambiente de desarrollo

La aplicación será desarrollada utilizando Ionic versión 1.2, AngularJS versión 1.4.6 y Apache Cordova versión 6.5.0. El IDE que será utilizado es Visual Studio Community 2015 versión 14.0.

El servidor será construido sobre Node.js versión 7.9.0, y el *framework* para Node.js será Express.js versión 4.15.2. El servidor de base de datos que se utilizará es CouchDB versión 1.6.1 y para la base de datos local de los dispositivos será PouchDB versión 5.4.5.

2. FASE DE INICIO

En la fase de desarrollo de este trabajo especial de grado, se preparó todo el ambiente de desarrollo para la correcta implementación de la aplicación móvil. Asimismo, y de forma paralela, se analizaron los requerimientos para lograr una refinación completa de la planificación.

2.1. Preparar ambiente de desarrollo

Para este proyecto, se decidió trabajar en un equipo Windows 10, primordialmente. Asimismo, para el desarrollo de la aplicación móvil multiplataforma, se utilizó Ionic y Apache Cordova; y como ambiente de desarrollo se utilizó Visual Studio 2015. Por esta razón, se utilizaron las tecnologías web HTML5, CSS3 y AngularJS, junto con el lenguaje de programación JavaScript. Como base de datos local a los dispositivos móviles se implementó PouchDB.

El sistema operativo del servidor es Ubuntu 16, en cual se creó un servidor web Node.js que utiliza el *framework* Express.js y se encarga de establecer la comunicación entre la aplicación y la base de datos, además de la ejecución de los eventos necesarios para el envío de correos electrónicos y notificaciones *push*. Además. Como servidor de base de datos se utilizó CouchDB, con el fin de facilitar los mecanismos de replicación y sincronización de las diferentes bases de datos.

En la sección de Anexos, en el Anexo A de este trabajo especial de grado, se pueden ver de forma más detallada y bajo un esquema de tutorial, los pasos necesarios para la instalación de Visual Studio 2015, Ionic y Apache Cordova. Como la plataforma móvil escogida para el desarrollo de la aplicación fue Android, también fue necesario instalar todos los paquetes SDK provistos por Android Studio para la correcta construcción del *.apk* de la aplicación. Por esta razón, en el Anexo B se detallan los pasos para la instalación de Android Studio y todos los paquetes necesarios para el desarrollo de aplicaciones móviles.

En el lado del servidor, fue necesario instalar Node.js para la implementación del servidor web, por lo que en el Anexo C se describe con mayor detalle los pasos a seguir; asimismo con la instalación del servidor de base de datos CouchDB en el Anexo D.

2.2. Guía de estilos y diseño de logos

Para una mejor experiencia de usuario y el desarrollo de una aplicación más intuitiva, se decidió marcar ciertos parámetros en lo referente al diseño de la aplicación y a los colores que se debían utilizar. Para la elección de colores se decidió tomar en cuenta ciertos aspectos, como los son el público objetivo y el tipo de aplicación. Como consecuencia de estas consideraciones, en la Figura 21 puede verse de forma gráfica la guía de estilos final de la aplicación.

***Figura 21*. Guía de estilos de la aplicación móvil**

Con el objetivo de proveer mejor experiencia de usuario, se decidió diseñar y crear los logos e imágenes propios a la aplicación; con el fin de evitar riesgos con la infracción de los derechos de autor. De esta manera, el logo final de la aplicación a mostrar en el dispositivo móvil, se puede observar en la figura 22.

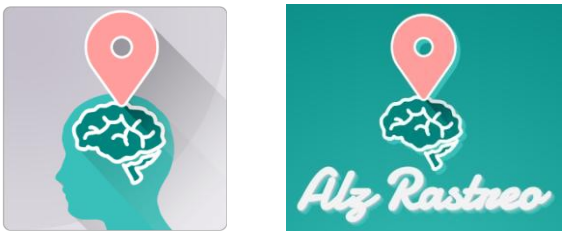

***Figura 22*. Logos de la aplicación móvilAlzRastreo**

En este logo se trató de mostrar las dos temáticas más relevantes tratadas en este trabajo especial de grado: la enfermedad de Alzheimer, representada por el cerebro, y la ubicación geoespacial, representada por el signo del marcador de mapa. Los colores que se escogieron tratan de reflejar seguridad y tranquilidad, así como dejar en evidencia que se trata de una aplicación móvil con fines de asistencia.

2.3.Especificación de módulos

Para este trabajo especial de grado que tiene como fin el desarrollo de una aplicación móvil que sirva como acompañante para aquellas personas que padezcan la enfermedad de Alzheimer, se definieron los siguientes módulos:

- Módulo de autenticación: este módulo engloba todo lo que involucra la autenticación de usuarios; desde el inicio de sesión y registro, hasta la recuperación de contraseña en caso de que ésta haya sido olvidada. Aquí es donde se definen los dos tipos de usuarios de la aplicación, y cómo el sistema los distingue para poder brindar las diferentes funcionalidades inherentes a cada tipo.
- Módulo de paciente: este módulo implementa todas las funciones y los servicios necesarios para la correcta entrega de las vistas relacionadas con los usuarios de tipo paciente. También involucra los servicios para las sincronizaciones de las bases de datos locales con las remotas, y las peticiones al servidor, así como la construcción de las herramientas necesarias para los requerimientos especificados en la fase anterior.

- Módulo de guardián: análogo al módulo paciente, pero para todos aquellos usuarios tipo guardián.
- Módulo de geolocalización: en este módulo es donde se manejan los objetos de la ubicación de los usuarios obtenidos del dispositivo móvil, así como la conexión con el API de Google Maps para su correcta visualización en los mapas. Aquí también se implementan las herramientas necesarias para la construcción de zonas seguras a través de un radio dado en metro, así como las fórmulas necesarias para determinar si dada la longitud y latitud de una ubicación de un usuario, éste se encuentra o no dentro de una zona segura.
- Módulo de *background*: este módulo es el que se encarga de implementar los servicios necesarios para que la aplicación continúe ejecutándose en el *background* del dispositivo móvil.
- Módulo de notificaciones *push*: este módulo involucra el manejo de las notificaciones *push* enviadas desde el servidor, y el registro de los dispositivos para el funcionamiento de estas notificaciones.
- Módulo del servidor web: en este módulo se implementa el servidor web que funciona como intermediario entre las peticiones de la aplicación móvil y el servidor de base de datos CouchDB. A su vez, también provee las funcionalidades necesarias para el desarrollo de los eventos que permiten al servidor monitorear las actualizaciones de la base de datos; y de esta manera realizar las notificaciones pertinentes.

3. FASE DE PRODUCCIÓN

En esta fase se implementó el desarrollo de las funcionalidades requeridas para la aplicación móvil “AlzRastreo”, dividiéndola en módulos y aplicando un ciclo de desarrollo iterativo e incremental. Esta fase tiene como meta la construcción de las funcionalidades núcleo, y así garantizar el éxito del proyecto. Las etapas que conforman la fase de producción involucran la planificación, el desarrollo y la liberación de los módulos que forman parte de la aplicación móvil. De esta manera, en la etapa de planificación se define el contenido base de la iteración, en la etapa de desarrollo se toman acciones de implementación del módulo de forma controlada y, por último, en la etapa de liberación se verifica y valida el módulo implementado.

3.1. Iteración 1

Se procede a detallar la iteración 1 del desarrollo de la aplicación móvil.

3.1.1. Planificación

El primer módulo que fue necesario implementar fue el módulo de autenticación. Debido a que el servidor CouchDB implementa sus propias herramientas de autenticación de usuarios, se decidió trabajar bajo este modelo, integrando a su vez el servidor web de Node.js para el manejo de los nombres de usuario, contraseñas y *tokens* de acceso. Para lograr esto se utilizó una

solución llamada SuperLogin del lado del servidor, y NG-SuperLogin para la integración con AngularJS en el lado del cliente.

3.1.2. Desarrollo

El paquete de autenticación SuperLogin debió ser configurado en el servidor web Node.js para su posterior uso, de la siguiente manera:

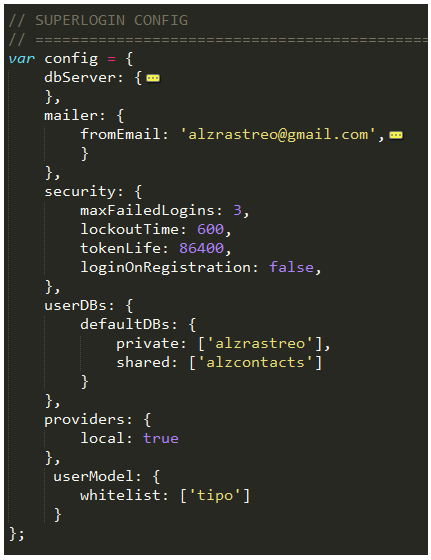

***Figura 23*. Configuración de SuperLogin en el servidor web Node.js**

En la configuración mostrada en la Figura 23, pueden verse detalles del servidor y la base de datos, entre otros, que implican campos de contraseñas, protocolos, usuarios, tiempo de vida de los *tokens* de autenticación y demás detalles de autenticación. SuperLogin proveyó de las herramientas para la construcción de *tokens* de accesos y contraseñas, permitiendo el uso de eventos de autenticación que fueron utilizados a través de la aplicación y el servidor web. Con esta configuración establecida, fue posible la inicialización de SuperLogin en el servidor web, como puede verse en la Figura 24.

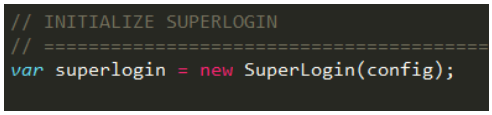

***Figura 24*. Inicialización de SuperLogin en el servidor web Node.js**

Una vez establecida esta parte del lado del servidor, es necesario integrarla en el lado del cliente. Esto se logró utilizando el paquete NG-SuperLogin, el cual se enlazó a SuperLogin con ayuda de AngularJS para su correcto funcionamiento, agregando la biblioteca en el código fuente de la aplicación, como puede observarse en la Figura 25.

```
<!--Dependencies-->
<script src="lib/ng-cordova-master/dist/ng-cordova.min.js"></script>
<script src="lib/ng-superlogin-master/src/ng-superlogin.js"></script>
<script src="lib/angular-messages/angular-messages.min.js"></script>
<script src="lib/http/http.min.js"></script>
```

***Figura 25*. Inclusión de NG-SuperLogin en el código fuente de la aplicación móvil**

Una vez configurados los paquetes que se utilizaron, se comenzó con la lógica de la aplicación para el módulo de autenticación. El módulo de autenticación está representado como un controlador basado en el patrón MVC, como puede verse en la figura 26.

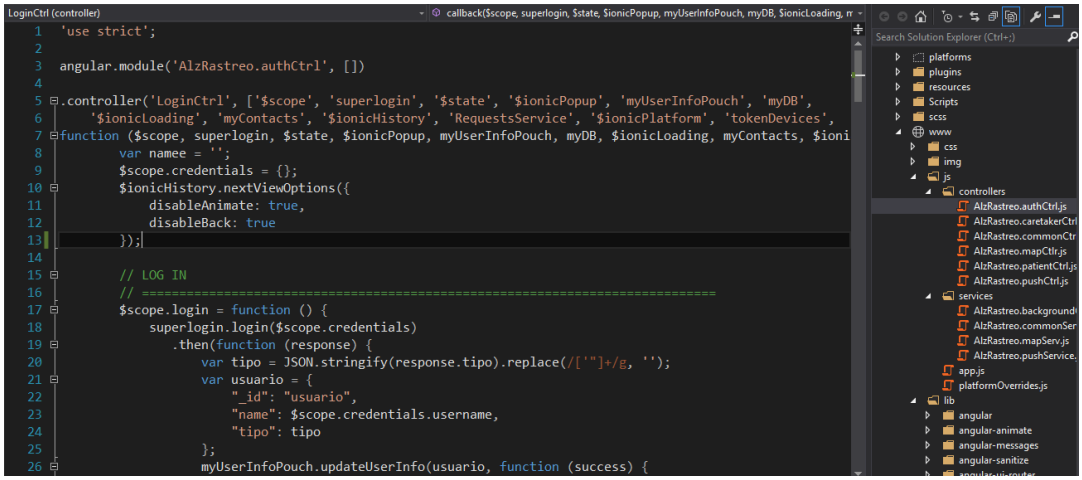

***Figura 26*. Controlador de autenticación en el código fuente de la aplicación móvil**

Las interfaces de usuario o vistas correspondiente a este módulo pueden verse en la Figura 27.

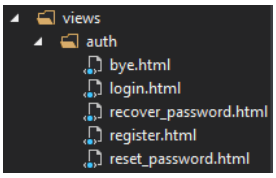

***Figura 27*. Vistas de autenticación en el código fuente de la aplicación móvil**

El módulo de autenticación no sólo se encarga de la autenticación de usuarios y sus registros en la aplicación móvil; también es el que establece la creación de las bases de datos locales en caso de inicio de sesión, la creación y enlace con la base de datos remota en el servidor, se encarga de definir las vistas a mostrar dependientes del tipo de usuario que inició sesión y se encarga de regresar el sistema a su estado inicial, al eliminar las bases de datos y demás datos locales generados por los usuarios cuando estos salen de su sesión en la aplicación. La razón de esto último es para evitar la contaminación de la replicación de los datos de otros usuarios a través de todas las bases de datos.

Para lograr todos los objetivos de este módulo, se crearon los siguientes archivos de código:

- app.js: en este archivo se definen todas las rutas de la aplicación. En el módulo de autenticación, tiene como fin la inicialización de NG-SuperLogin y los métodos y

funciones para la autenticación, registro y olvido de contraseña. Este archivo se utiliza a lo largo de todos los módulos.

- AlzRastreo.authCtrl.js: es donde se encuentra toda la lógica de autenticación, registro, olvido de contraseña y chequeo de nombre de usuario o correo electrónico ya utilizado. Este archivo es el que le indica a la aplicación cómo comportarse dependiendo del tipo de usuario que inicia la sesión.
- AlzRastreo.commonServices.js: en este archivo se definen los servicios que involucran la creación y sincronización de la base de datos local, y la interacción con el servidor web. Este archivo se utiliza a lo largo de toda la aplicación y todos sus módulos por cualquier tipo de usuario.

3.1.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. En la Figura 28 puede verse de forma gráfica las interfaces de usuarios finales que involucran el módulo de autenticación de la aplicación.

***Figura 28*a. Interfaz de usuario de inicio de sesión**

***Figura 28 b29.* Interfaz de usuario de registro**

***Figura 28 c30*. Interfaz de usuario de recuperación de contraseña**

En la interfaz de usuario de inicio de sesión pueden observarse las tres diferentes acciones que un usuario puede escoger, al abrir la aplicación desde su dispositivo móvil. La primera acción es la de autenticar un usuario ya registrado con su nombre de usuario y la contraseña, presionando el botón de “Iniciar Sesión”; esta acción lleva al usuario hacia su página de inicio correspondiente, dependiendo de su tipo de usuario. La segunda acción que puede tomar es la de registrarse al presionar el enlace “¿No estás registrado? Regístrate.”, en cuyo caso la aplicación permite al usuario navegar a la interfaz de registro de usuario; una vez en ella, su muestran los campos requeridos que deben ser llenados para un registro exitoso. Finalmente, la última opción es la de recuperar contraseña, en cuyo caso el usuario debe ingresar el correo electrónico con el que está registrado a la aplicación, y esperar un correo con más instrucciones.

3.2.Iteración 2

Se procede a detallar la iteración 2 del desarrollo de la aplicación móvil.

3.2.1. Planificación

El segundo módulo núcleo de la aplicación, es el módulo paciente. Este módulo es el que se encarga de la navegación de los usuarios a todas las interfaces a la que los pacientes tienen acceso. Asimismo, y a través de ciertos servicios definidos en AngularJS, se encarga de la correcta sincronización de bases de datos PouchDB con la base de datos CouchDB y de otras funcionalidades en lo referente al servidor web para la obtención de datos, así como demás lógica de la aplicación.

3.2.2. Desarrollo

Para el desarrollo de este módulo se definieron los correspondientes controladores y servicios de AngularJS para el acceso a la base de datos PouchDB locales y las peticiones HTTP al servidor web para la lectura, actualización y eliminación de datos almacenados en el servidor de base de datos CouchDB a través de las rutas definidas en éste. Las bibliotecas que se utilizaron se pueden ver en la Figura 29, junto con el controlador definido.

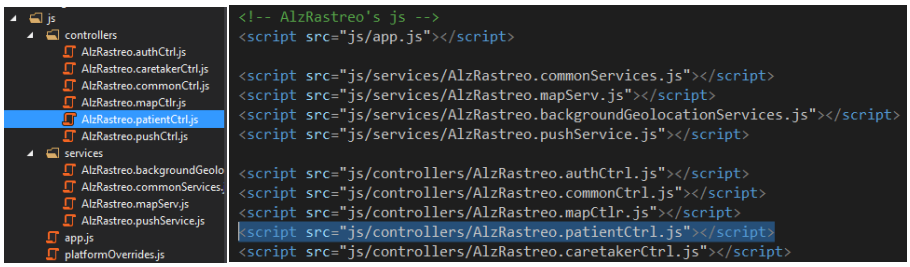

***Figura29*. Controlador de usuarios pacientes e inclusión de bibliotecas en el código fuente de la aplicación móvil**

Una vez configurados los paquetes y bibliotecas a utilizar, se comenzó con la lógica de la aplicación para el módulo paciente. El módulo paciente está representado como un controlador basado en el patrón MVC, como puede verse en la figura 30.

***Figura 30*. Controlador de los usuarios pacientes en el código fuente de la aplicación móvil**

En este módulo es donde se implementa toda la lógica de la obtención de los datos y su enlace a las vistas del sistema para la correcta visualización de estos datos por parte del usuario. Además, también provee todas las funcionalidades que involucran el enrutamiento a las diferentes interfaces y las acciones que deben ser tomadas en ellas.

Las vistas que corresponden a este módulo se pueden ver en la Figura 31.

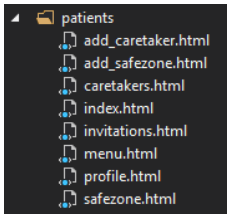

***Figura 31* Vistas del módulo paciente en el código fuente de la página**

El módulo paciente se encarga de realizar las peticiones HTTP al servidor web para la lectura, actualización y eliminación de datos a la base de datos de CouchDB. Asimismo, también se encarga de las múltiples verificaciones necesarias para el correcto funcionamiento de la aplicación móvil, correspondiente a los usuarios tipo paciente.

De esta forma, y para lograr todos los objetivos de este módulo, se crearon los siguientes archivos de código:

- app.js: en este archivo se definen todas las rutas de la aplicación. En el módulo paciente, es donde se realizan las autenticaciones adecuadas antes del enrutamiento de cada vista de este módulo.
- AlzRastreo.patientCtrl.js: es donde se encuentra toda la lógica de del módulo paciente, y las funciones que se ejecutan dependiendo de las diferentes acciones que el usuario paciente puede realizar sobre el sistema.

3.2.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. De esta manera, las interfaces de inicio definidas para los pacientes se pueden ver en la Figura 32.

***Figura 32 a*. Interfaces de inicio para los usuarios paciente de la aplicación móvil cuando está en un lugar seguro.**

***Figura 32 b*. Interfaces de inicio para los usuarios paciente de la aplicación móvil cuando no está en un lugar seguro**

***Figura 32 c*. Interfaces de inicio para los usuarios paciente de la aplicación móvil cuando no tiene zonas seguras definidas**

Cada interfaz mostrada en la Figura 32 se construye a través de ciertas verificaciones que la aplicación realiza para mostrar la vista correspondiente al usuario paciente. De esta manera, cuando un paciente ingresa a la aplicación, lo primero que se muestra es su ubicación geoespacial a través del API de Google Maps. Si el usuario que entra a la aplicación es un paciente con zonas seguras registradas y en un lugar seguro, se muestra la interfaz de inicio con la notificación: “Estás en un lugar seguro”, si, por el contrario, el paciente se encuentra fuera de alguna zona segura asociada a su cuenta, la notificación que se muestra es: “No estás en un lugar seguro. En estos momentos tus guardianes están siendo notificados de tu ubicación.” En dado caso de que el paciente no tenga zonas seguras asociadas a su cuenta aún, la notificación muestra: “No tienes zonas seguras definidas. Contacta a tu guardián para definir tu primera zona.”

Desde la vista de inicio, el guardián puede navegar a tres interfaces diferentes, o quedarse en el inicio. Si desea navegar a la ventana de guardianes, la aplicación entrega las vistas representada por la Figura 33.

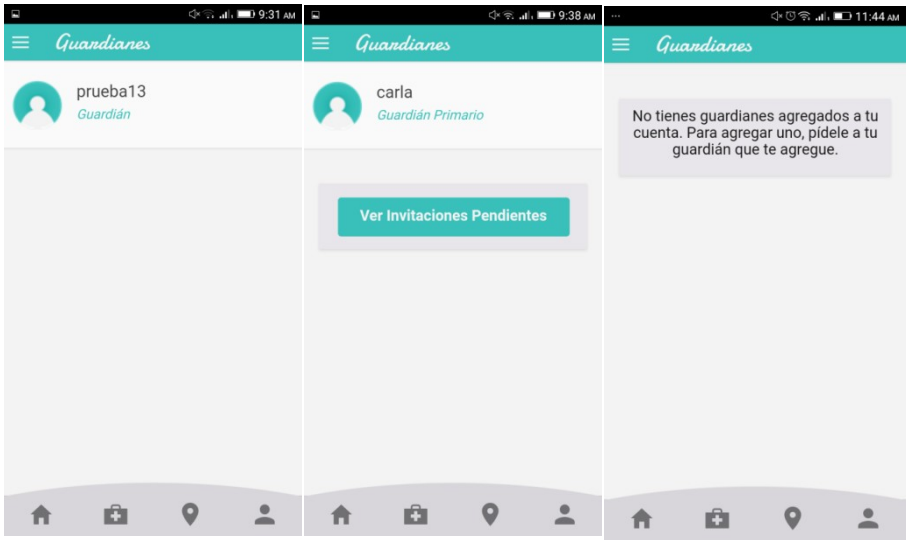

***Figura 33*. Interfaces de guardianes para los usuarios paciente de la aplicación móvil**

Si el usuario paciente tiene guardianes agregados a su cuenta y aceptados con anterioridad, se muestran en forma de lista con su título de "Guardián" o "Guardián Primario", dependiendo del caso. Si el paciente tiene además invitaciones pendientes de otros guardianes que aún no han sido aceptadas, se muestra un botón que permite la navegación a la vista que lista estas invitaciones para su eliminación o aceptación. Asimismo, en caso de que el paciente no tenga guardianes registrados, se mostrará una notificación de "No tienes guardianes agregados a tu cuenta. Para agregar uno, pídele a tu guardián que te agregue como paciente."

Un paciente puede navegar también a sus zonas seguras. Las vistas que representan las zonas seguras pueden verse en la Figura 34.

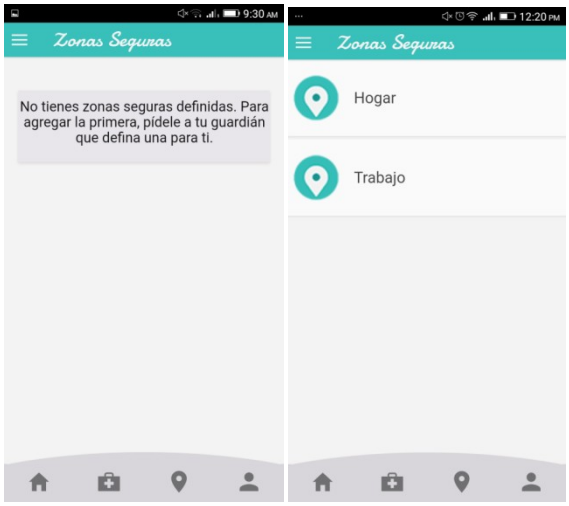

***Figura 34*. Interfaces de zonas seguras para los usuarios pacientes de la aplicación móvil**

Si el paciente no tiene zonas seguras definidas, una notificación se muestra con las siguientes palabras: "No tienes zonas seguras definidas. Para agregar la primera, pídele a tu guardián que

defina una para ti.” Si, por el contrario, tiene zonas seguras definidas, se listan una debajo de la otra con el nombre de la zona segura correspondiente.

3.3.Iteración 3

Se procede a detallar la iteración 3 del desarrollo de la aplicación móvil.

3.3.1. Planificación

El tercer módulo que se implementará en la aplicación es el módulo guardián. Análogo al módulo paciente, éste se encarga de navegación de los usuarios a todas las interfaces a la que los guardianes tienen acceso. Utilizando servicios de AngularJS parecidos a los del módulo paciente, tiene como objetivo la sincronización de bases de datos PouchDB con la base de datos CouchDB y la implementación de otras funcionalidades en lo referente al servidor web para la obtención de datos, así como la lógica de la aplicación en lo referente a los guardianes.

3.3.2. Desarrollo

Para el desarrollo de este módulo se definieron los correspondientes controladores y servicios de AngularJS para el acceso a la base de datos PouchDB locales y las peticiones HTTP al servidor web para la lectura, actualización y eliminación de datos almacenados en el servidor de base de datos CouchDB a través de las rutas definidas en éste. Las bibliotecas que se utilizaron son las mismas que para el módulo de los pacientes, mostrados en la Figura 29.

Una vez configurados los paquetes y bibliotecas a utilizar, se comenzó con la lógica de la aplicación para el módulo de guardián. El módulo guardián está representado como un controlador que puede verse con mejor detalle en la figura 35.

***Figura 35*. Controlador de los usuarios guardianes en el código fuente de la aplicación móvil**

Como el módulo paciente, este módulo también implementa toda la lógica de la obtención de los datos y su enlace a las vistas del sistema para la correcta visualización de estos datos por parte del guardián. Además, también se encarga de proveer todas las funcionalidades que involucran el enrutamiento a las diferentes interfaces y las acciones que deben ser tomadas en ellas.

Las vistas que corresponden a este módulo se pueden ver en la Figura 36.

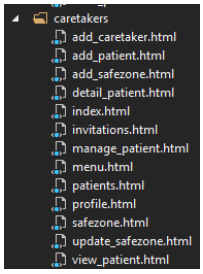

Figura 36. **Vistas del módulo guardián en el código fuente de la aplicación**

Para lograr todos los objetivos de este módulo, se crearon los siguientes archivos de código:

- app.js: en este archivo se definen todas las rutas de la aplicación. En el módulo guardián, es donde se realizan las autenticaciones adecuadas antes del enrutamiento.
- AlzRastreo.caretakerCtrl.js: en este archivo se encuentra toda la lógica del módulo guardián, así como todas las funciones que se ejecutan dependiendo de las diferentes acciones que el usuario guardián puede realizar sobre el sistema.

3.3.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. De esta manera, las interfaces de inicio definidas para los guardianes pueden verse en la Figura 37.

Figura 37 a. **Interfaces de inicio de los usuarios pacientes de la aplicación móvil cuando no tiene guardianes agregados**

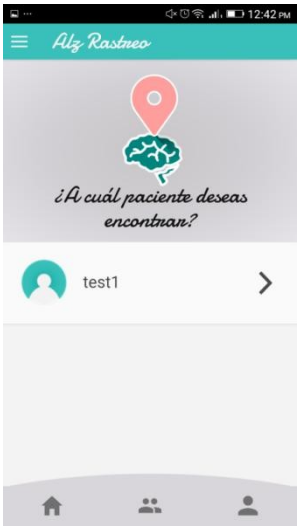

Figura 37 b. **Interfaces de inicio de los usuarios pacientes de la aplicación móvil cuando tiene guardianes agregados**

Figura 37 c. **Interfaces de inicio de los usuarios pacientes de la aplicación móvil cuando tiene invitaciones pendientes.**

Si el guardián que ingresa a la interfaz de inicio de la aplicación no tiene aún ningún paciente agregado, le aparece una notificación con la frase: “No tienes pacientes agregados a tu cuenta.” Si, por el contrario, un paciente ingresa y ya tiene pacientes agregados a su cuenta, los cuales ya

aceptaron su invitación y una relación se formó entre ambos, se listan en la aplicación con su nombre de usuario; de forma que el usuario guardián puede ingresar a la ubicación de cada uno de ellos. Consecuentemente, en caso de que el guardián haya agregado a un paciente y éste aún no lo haya aceptado, se muestra una notificación de: “Ver invitaciones enviadas.”

Cuando un guardián ingresa a uno de los pacientes que son listados en la vista de inicio, las interfaces de usuario mostradas corresponden a la Figura 38.

***Figura 38 a*. Interfaces de ver paciente de los usuarios guardianes de la aplicación móvil cuando el guardián es primario**

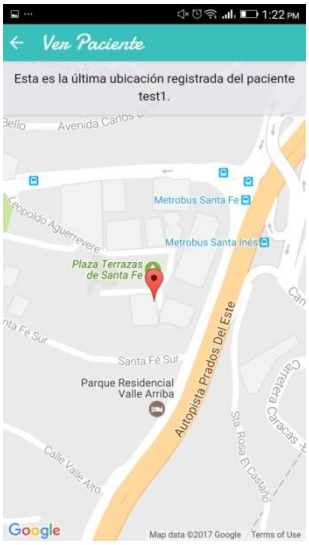

***Figura 38 b*. Interfaces de ver paciente de los usuarios guardianes de la aplicación móvil cuando el guardián no es primario**

Cuando un guardián tiene agregado un paciente del cual es el Guardián Primario, la vista de la última ubicación del paciente contiene un menú especial que le permite al guardián administrar la cuenta del paciente de la siguiente manera: administrando los guardianes del paciente, administrando las zonas seguras del paciente y visualizando los detalles de su cuenta. Además, se muestra una notificación, la cual dice: “Al ser Guardián Primario de test1, tienes privilegios especiales sobre su cuenta.”. Por el contrario, si el guardián que tiene agregado al paciente no es su guardián primario, sólo puede acceder a su última ubicación y visualizarla en un mapa.

Cuando un guardián presiona administrar guardianes, el módulo guardián muestra las interfaces que pueden verse en la Figura 39.

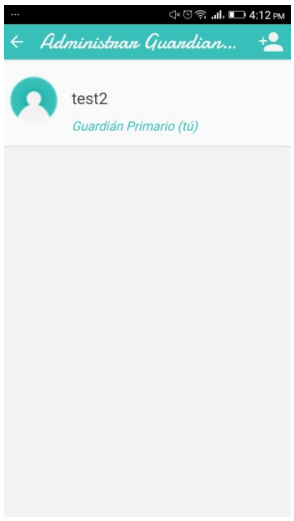

***Figura 39 a*. Interfaz de administrar guardianes**

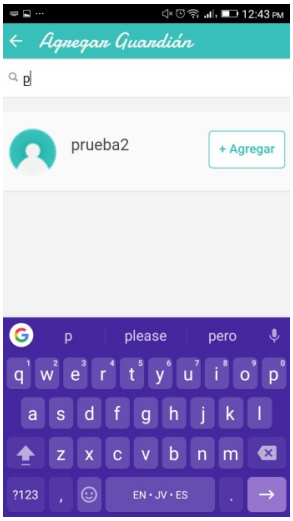

***Figura 39 b*. Interfaz de agregar guardián a un paciente**

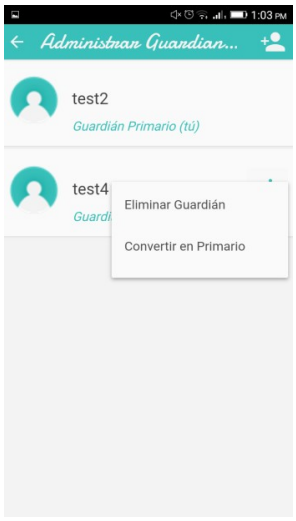

***Figura 39 c*. Interfaz de eliminar o convertir guardián en primario**

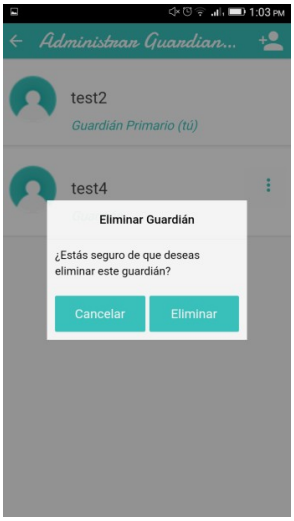

***Figura 39 d*. Interfaz de alerta de eliminar guardián**

Unos guardianes con privilegios especiales de guardián primario tienen acceso a una lista de todos los guardianes que un paciente en particular posee, y puede agregar otros guardianes a ese paciente, buscándolos en una barra especial que filtra usuarios de tipo guardián y enviando una invitación al paciente para su aceptación. Además, puede ceder su privilegio de guardián primario a cualquier otro guardián aceptado por el paciente, o eliminar algún guardián.

Por otro lado, cuando un guardián presiona administrar zonas seguras, las interfaces de usuario que involucran esta acción pueden observarse en la Figura 40.

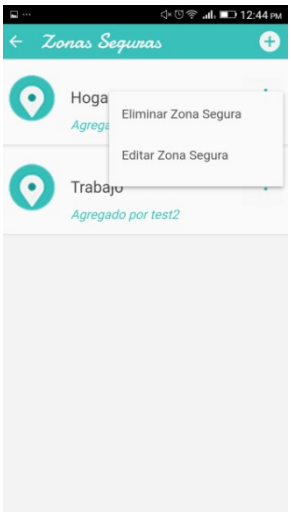

***Figura 40 a*. Interfaz de ver zonas seguras de un paciente**

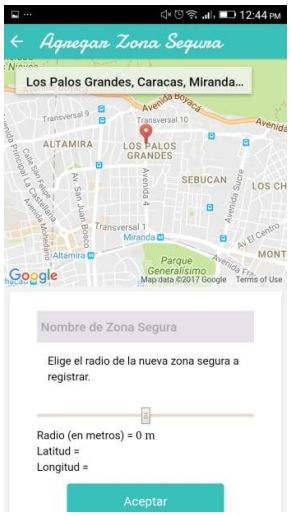

***Figura 40 b*. Interfaz de agregar una zona segura a un paciente**

***Figura 40 c*. Interfaz de editar una zona segura a un paciente**

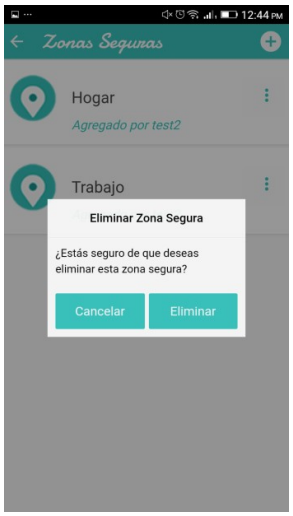

***Figura 40 d*. Interfaz de eliminar zona segura de un paciente**

Como usuario guardián primario, un guardián puede agregar, eliminar y editar las zonas seguras de un paciente en especial. Para agregar una zona segura, se utiliza el API de Google Maps para

la visualización de los mapas; sobre estos mapas el usuario escoge la ubicación específica donde quiere localizar la zona y un radio. El radio escogido representa el radio del círculo que define los límites geográficos de la zona segura que el paciente puede recorrer sin estar en peligro. Con la barra de búsqueda, es posible buscar ubicaciones geográficas por todo el mundo para la agregación de zonas seguras. Un guardián primario también puede editar una zona segura ya definida, cambiando su nombre, radio y hasta su ubicación geográfica.

Un usuario guardián puede navegar también a sus pacientes. Las vistas que representan estas interfaces de usuario pueden verse en la Figura 41.

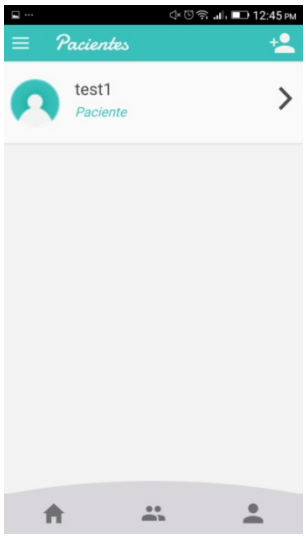

Figura 41 a. **Interfaz de ver pacientes para usuarios guardianes**

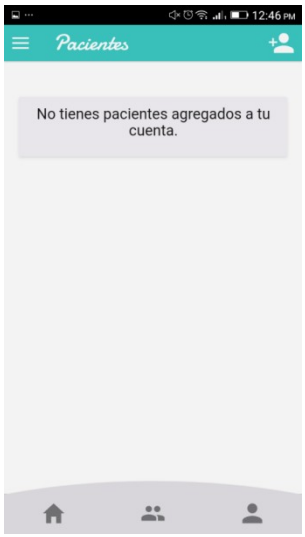

Figura 41 b. **Interfaz de ver pacientes para usuarios guardianes cuando no tienen pacientes agregados**

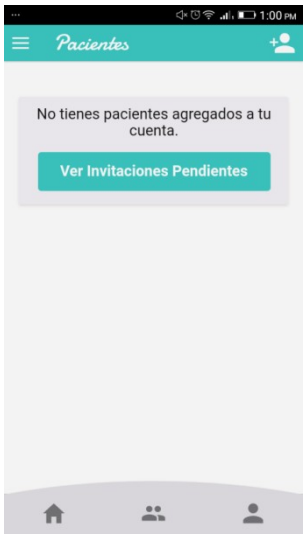

Figura 41 c. **Interfaz de ver pacientes para usuarios guardianes cuando han enviado una invitación a un paciente**

De forma análoga a los usuarios de tipo paciente, los guardianes pueden ver en forma de lista los pacientes que tienen agregados a su cuenta. En caso de que un guardián no haya enviado alguna invitación a un paciente o aún no tenga pacientes agregados a su cuenta, se muestra la notificación que indica “No tienes pacientes agregados a tu cuenta.” Por otro lado, si un paciente ya envió una invitación a un paciente, pero éste aún no la ha aceptado, se muestra un botón que permite la visualización de la lista de las invitaciones pendientes por el guardián.

3.4.Iteración 4

Se procede a detallar la iteración 4 del desarrollo de la aplicación móvil.

3.4.1. Planificación

El cuarto módulo núcleo que se implementó en la aplicación, es el módulo de geolocalización. Este es el módulo que define los servicios y la configuración de los paquetes, y controladores necesarios para la correcta obtención de los datos obtenidos por los servicios de ubicación del

dispositivo móvil, la conexión con el API de Google Maps y su almacenamiento en la base de datos.

3.4.2. Desarrollo

Para el desarrollo de esta aplicación fue necesaria la inclusión al proyecto de Cordova el *plugin* núcleo de geolocalización, como puede verse en la figura 42.

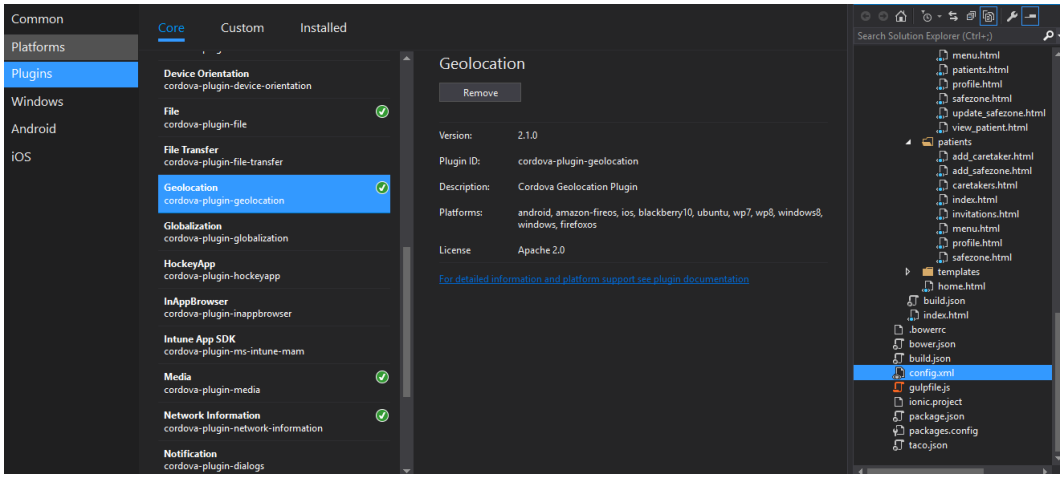

***Figura 42.*Inclusión del plugin Geolocation en el código fuente de la aplicación móvil**

Este *plugin* provee información concerniente a la ubicación del dispositivo móvil, entre las que se encuentran la latitud y longitud. Con esta información, se crearon los servicios necesarios para la generación de los mapas y los marcadores de ubicación, como puede verse en forma parcial en la Figura 43.

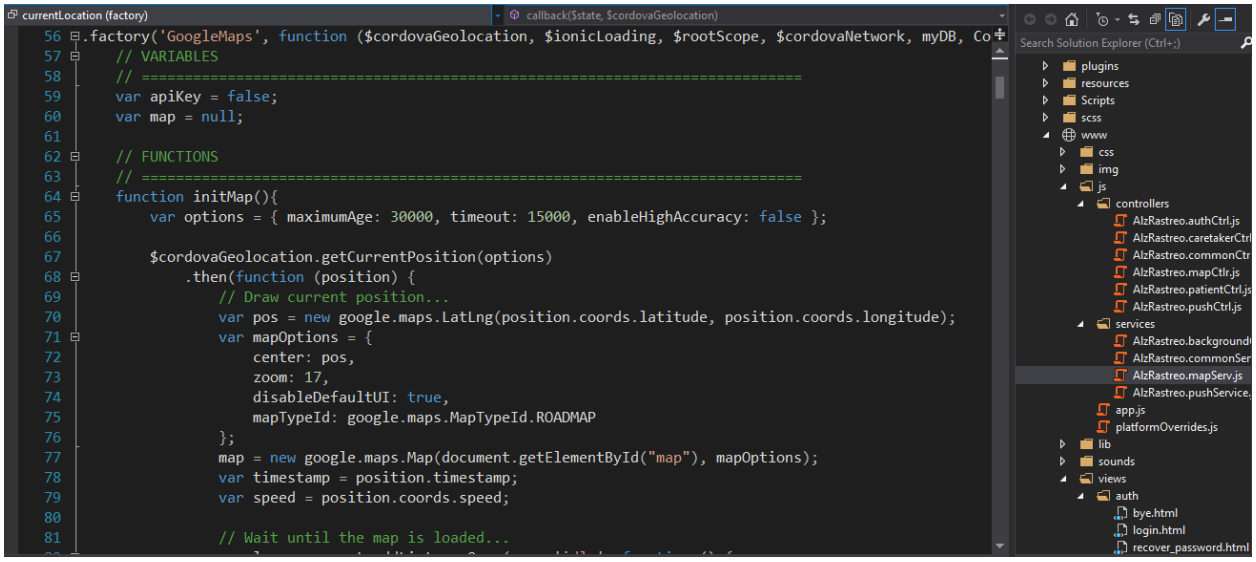

***Figura 43.* Servicio de geolocalización en el código fuente de la aplicación móvil**

Una vez obtenido los datos de longitud y latitud, fue posible instanciar un objeto google.maps.LatLng para la visualización de los mapas, marcadores y zonas seguras, además de

las funciones necesarias para establecer si dada una ubicación y una lista de zonas seguras, un paciente se encuentra en un lugar seguro; es decir, dentro de una zona segura.

Para lograr todos los objetivos de este módulo, se crearon los siguientes archivos de código:

- AlzRastreo.mapCtrl.js: en este archivo se define el controlador del módulo de geolocalización. Es el encargado de la lógica a nivel de mapas y la conexión con el API de Google Maps.
- AlzRastreo.mapServ.js: este archivo contiene todas las interacciones con el *plugin* de geolocalización de Cordova, así como los servicios de la base de datos y el servidor web.

3.4.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. Los servicios que se pudieron establecer en este módulo fueron utilizados a lo largo del módulo paciente y guardián para el diseño, desarrollo e implementación de los mapas y notificaciones necesarias. Asimismo, este módulo permitió informar al servidor web cuándo un paciente se encuentra fuera de una zona segura, y de esta forma realizar las notificaciones pertinentes a través de correos electrónicos y notificaciones *push*.

3.5. Iteración 5

Se procede a detallar la iteración 5 del desarrollo de la aplicación móvil.

3.5.1. Planificación

El quinto módulo de la aplicación es el módulo de *background*. Este es el módulo que provee a la aplicación con los servicios, la configuración de los paquetes y los controladores necesarios para que pueda ser ejecutada en forma de *background* en el dispositivo móvil. Para el desarrollo de este módulo se utilizó un *plugin* de Cordova llamado CDVBackgroundGeolocation que permite conocer la ubicación geográfica en forma de latitud y longitud del dispositivo móvil.

3.5.2. Desarrollo

Para el desarrollo de este módulo fue necesaria la inclusión del *plugin*CDVBackgroundGeolocation, como puede verse en la figura 44.

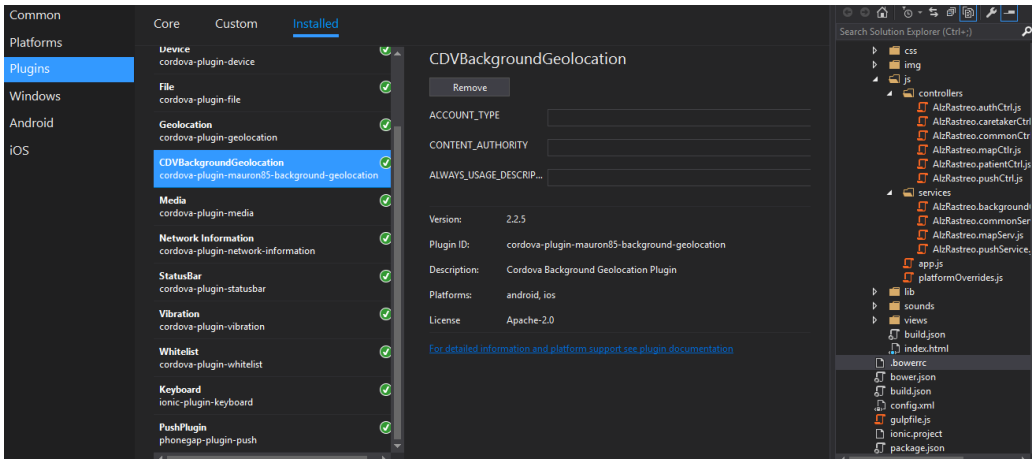

Figura 44. **Inclusión del *plugin* CDVBackgroundGeolocation en el código fuente de la aplicación móvil**

Con el *plugin* correctamente instalado, se pudo implementar el código que permite saber a la aplicación en qué modo se encuentra; es decir, si está ejecutándose en el *background* o en el *foreground* del dispositivo móvil. Esta última parte puede observarse de mejor manera en la Figura 45 y en la Figura 46.

```
// BACKGROUND GEOLOCATION
// ================================================================================
$ionicPlatform.on("pause", function () {
    if (superlogin.authenticated()) {
        myUserInfoPouch.isPatient(function (successs) {
            if (successs === true) {
                BackgroundGeolocationService.init();
            }
        });
    } else {
        console.log("NOT AUTHENTICATED.... DO NOTHING");
    };
});
$ionicPlatform.on("resume", function () {
    BackgroundGeolocationService.stop();
});
```

Figura 45. **Código de *background geolocation* en el código fuente de la aplicación móvil**

Figura 46. Servicio de *background geolocation*en el código fuente de la aplicación móvil

Una vez obtenidos los datos de latitud y longitud, fue posible instanciar el objeto google.maps.LatLng() para la construcción de las zonas seguras y poder determinar si el paciente se encuentra o no dentro de una zona segura.

Para lograr con los objetivos propuestos de este módulo, se desarrollaron los siguientes archivos de código:

- AlzRastreo.backgroundGeolocationService.js: define todas las interacciones con el *plugin*CDVBackgroundGeolocation de Cordova y la API de Google Maps, así como las conexiones con la base de datos y el servidor web.

3.5.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. Al ejecutarse la aplicación en forma de *background* en el dispositivo móvil, se muestra una notificación que le permita saber de esto al usuario paciente. La notificación que se muestra puede verse en la Figura 47.

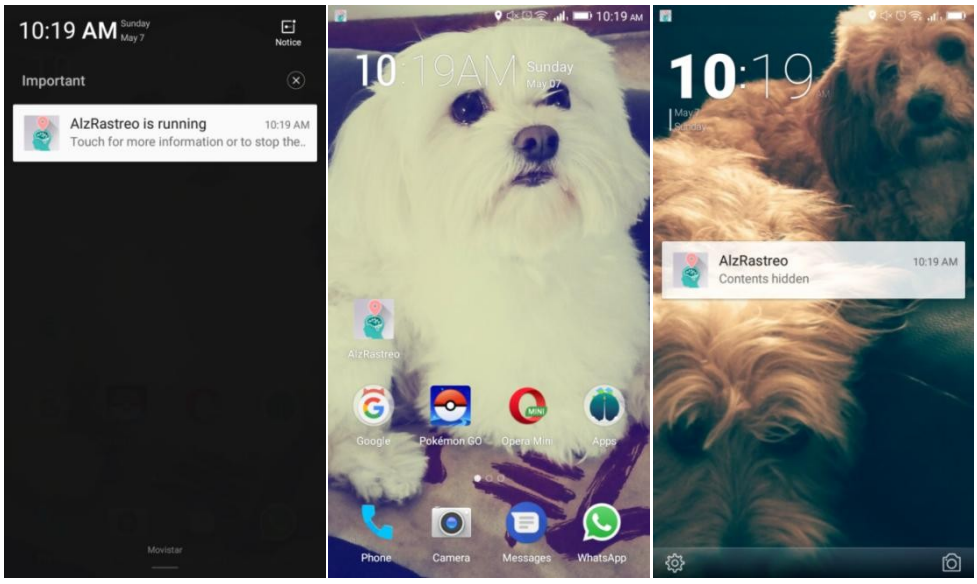

Figura 47. **Vista de la notificación de la aplicación móvil ejecutándose en el *background* del dispositivo móvil**

3.6.Iteración 6

Se procede a detallar la iteración 6 del desarrollo de la aplicación móvil.

3.6.1. Planificación

El quinto módulo de la aplicación es el módulo de notificaciones *push*. Este es el módulo que provee a la aplicación con los servicios, la configuración de los paquetes y los controladores necesarios para la implementación de las notificaciones *push*. Para esto, fue necesaria la agregación del *plugin* llamado PushPlugin en el lado del cliente y el paquete node-gcm para el lado del servidor web.

3.6.2. Desarrollo

Para el desarrollo de este módulo fue necesaria la inclusión del *plugin*PushPlugin, como puede verse en la Figura 48, en el lado del cliente para la recepción de las notificaciones *push*.

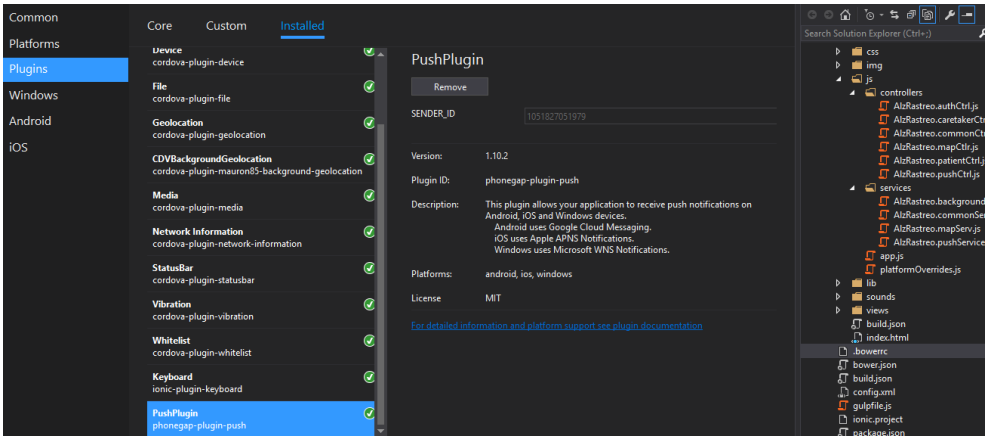

Figura 48. **Inclusión del *plugin*PushPlugin en el código fuente de la aplicación móvil**

Mientras que en el lado del servidor web fue necesario agregar el paquete node-gcm para el envío de las notificaciones *push*, como se muestra en la Figura 49.

```
'use strict';
var gcm = require('node-gcm');


// CREATE SENDER & MESSAGE
// ================================================================================
var apiKey = "AIzaSyBTJH236lqoIJvZrJ6xhFweHs6X6IVr7hk";
var service = new gcm.Sender(apiKey);
var message = new gcm.Message();


// SETUP NOTIFICATION TO SEND
// ================================================================================
module.exports = {
    sendNotification: function(receivers, patient, lat, lng){
        console.log( "SENDING NOTIFICATIONS... ");
        var title = patient + " esta fuera de una zona segura";
        message.addData('title', patient + " esta fuera de una zona segura");
        message.addData('message', 'En estos momentos estamos rastreando su ubicacion');

        service.send(message, { registrationTokens: receivers }, function (err, response) {
            if(err) console.error(err);
            else    console.log(response);
        });
    }
};
```

Figura 49. **Implementación de las *pushnotifications*en el código fuente del servidor web Node.js**

Para la recepción de las notificaciones *push*, fue necesaria la inicialización del *plugin* en el lado del cliente, la cual puede verse en la Figura 50. De esta forma, el dispositivo está listo para la recepción de las notificaciones enviadas por el servidor web.

Figura 50. **Implementación del código de las *push notifications* en el código fuente de la aplicación móvil**

En general, se desarrollaron los siguientes archivos de código para el envío y recepción de las notificaciones *push*:

- AlzRastreo.pushService: este archivo contiene todos los servicios necesarios para el registro de los dispositivos móviles a través del servidor web.

3.6.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. Cuando el servidor web envía una notificación al dispositivo móvil del guardián, la notificación obtenida puede observarse en la Figura 51.

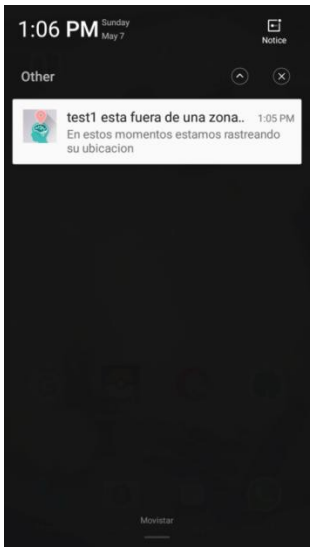

Figura 51. **Vista de las *pushnotifications* en el dispositivo móvil**

3.7. Iteración 7

Se procede a detallar la iteración 7 del desarrollo de la aplicación móvil.

3.7.1. Planificación

El último módulo de este proyecto es el módulo servidor web. El servidor web tiene como meta trabajar como un intermediario entre la aplicación móvil “AlzRastreo” y el servidor de base de datos CouchDB, además de implementar las herramientas de notificación a los guardianes cuando sus pacientes se encuentran fuera de un lugar seguro. De esta manera, se decidió utilizar Node.js y el *framework* Express.js para la creación del servidor web.

3.7.2. Desarrollo

Para el desarrollo del servidor web se utilizó Node.js y Express.js. El package.json generado para la correcta instalación del servidor puede verse en la Figura 52. Las instrucciones para instalar este servidor web se encuentran en el Anexo C.

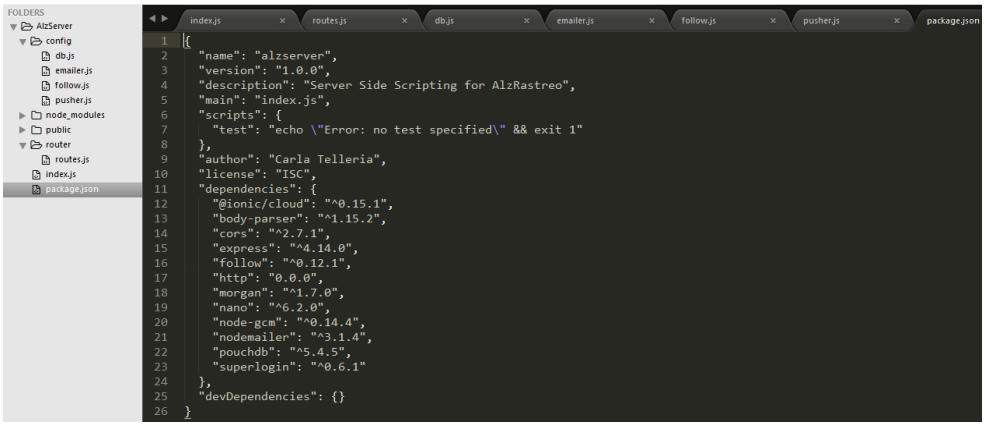

***Figura 52*. Definición del package.json del servidor web Node.js**

El servidor web de este proyecto se generó utilizando Express.js, el cual puede observarse con mejor detalle en la Figura 53.

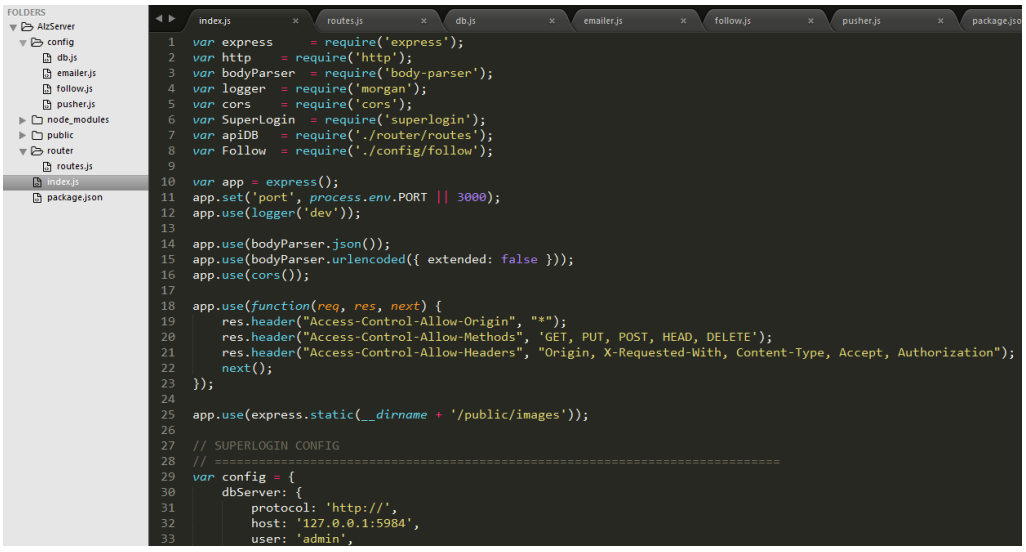

***Figura 53*. Código fuente del servidor web Node.js**

Para este servidor web, cada petición HTTP debe poseer las cabeceras correctas permitidas, y en él se permiten los siguientes métodos HTTP: GET, POST, HEAD y DELETE. Una vez las configuraciones del servidor se realizaron, fue necesaria la creación de un módulo definido como modules.export que trabaja como un API entre la aplicación y la base de datos. En otras palabras, fue necesaria la definición de las rutas a las que la aplicación debe realizar sus peticiones, como puede observarse en la Figura 54.

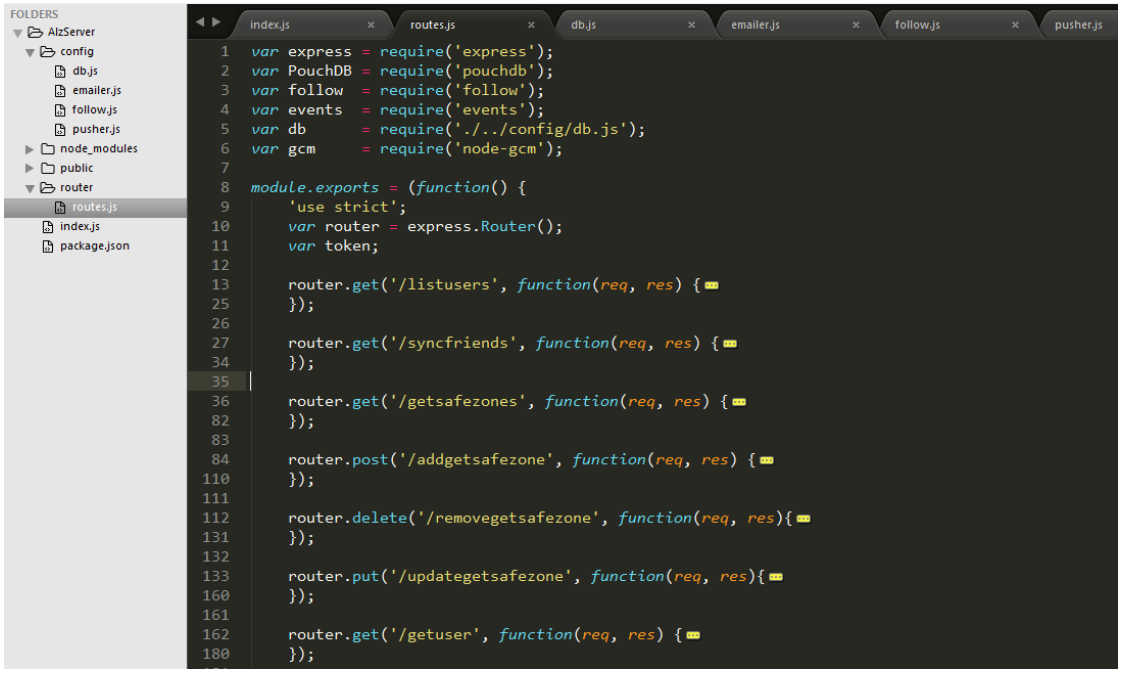

Figura 54. Definición de las rutas en el servidor web Node.js

Estas rutas permiten a la aplicación interactuar con los datos de la base de datos CouchDB. Una vez estas rutas fueron implementadas, se decidió que la forma en la que el sistema realizaba las notificaciones necesarias a los guardianes de que sus pacientes se encontraban fuera de un lugar seguro, era a través del servidor web.

Para lograr esto, fue necesario el desarrollo de manejadores de eventos, que tienen como objetivo disparar un evento cada vez que algún cambio se hace en las bases de datos de los pacientes, con respecto a datos de ubicación y si se encuentran fuera de sus zonas seguras establecidas. Para lograr esto, se utilizó una biblioteca de CouchDBllamada*follow*, la cual permite el monitoreo de las bases de datos CouchDB. En la Figura 55 puede verse con mejor detalle el uso de esta biblioteca.

```
// EVENT AND LISTENER SETUP
// =====================================================================
// Listener for database created everytime a user is registered.
superlogin.once('signup', function(userDoc, provider){
    console.log('DB: ' + userDoc._id);
    var db = userDoc._id;
    Follow.followDB(db);
});
```

Figura 55. **Definición de la biblioteca *follow* para el monitoreo de las bases de datos CouchDB en el servidor web Node.js**

De esta manera, y cada vez que un usuario se registra y su base de datos es creada con SuperLogin, la función followDB se encarga de monitorear si un documento con una bandera de "Safe" levantada, es agregado a la base de datos, como muestra la Figura 56.

```
// EXPORT FOLLOW DB OBJECTS
// ================================================================================
module.exports = {
    followDB: function (dbName) {
        console.log("Following: " + dbName);
        var database = "http://admin:adminpassword@localhost:5984/alzrastreo$" + dbName;
        follow({db: database, include_docs: true, since: "now"}, function(error, change) {
            if(!error) {
                if(change.doc.location && change.doc.safe === 'N'){
                    console.log("Notifying caretakers for... " + dbName);
                    listCaretakers(dbName, change.doc.location.lat, change.doc.location.lng);
                };
            } else {
                console.log(error);
            };
        });
    }
};
```

***Figura 56*. Definición de funciones para el monitoreo de las bases de datos CouchDB en el servidor web Node.js**

Cuando esto ocurre, el servidor se encarga de enviar las notificaciones *push* y los correos electrónicos que pueden verse con mejor detalle en la Figura 57.

```
// SETUP EMAIL TO SEND
// ================================================================================
module.exports = {
    sendEmail: function(receivers, patient, lat, lng){
        receivers = receivers.toString();
        var date = new Date();
        var year = date.getFullYear();
        var subject = patient + ' se encuentra fuera de una zona segura';
        var html = '<!DOCTYPE html><html lang="es"><head><meta name="format-detection'
            from: '<alzrastreo@gmail.com>',
            to: receivers,
            subject: subject,
            html: html
        };
        transporter.sendMail(mailOptions, (error, info) => {
            if (error) {
                return console.log(error);
            }
                console.log('Message %s sent: %s', info.messageId, info.response);
        });
    }
};
```

Figura 57. Código para el envío de los correos electrónicos desde el servidor web Node.js

Para lograr los objetivos de este módulo, el servidor web está formado por los siguientes archivos de código que representan la lógica del servidor:

- index.js: contiene todas las configuraciones para la inicialización del servidor web y los eventos para el monitoreo de las bases de datos.
- route.js: es donde se establecen todas las rutas del servidor web, a través de las cuales la aplicación móvil tiene interacción con las bases de datos CouchDB.
- db.js: contiene las variables y los objetos para las configuraciones de la base de datos, permitiendo modularidad al servidor web.

- emailer.js: es el código que se encarga de enviar los correos electrónicos a los guardianes cada vez que un paciente se encuentra fuera de un lugar seguro.
- follow.js: en este archivo se encuentran todas las funciones para el monitoreo de las bases de datos CouchDB.
- pusher.js: contiene el código para el envío de las notificaciones push a los guardianes cada vez que un paciente se encuentra fuera de un lugar seguro.

3.7.3. Liberación

Los resultados obtenidos en la presente iteración fueron exitosos. Cuando el servidor web está activo, en el terminal se muestra cada registro de las peticiones de los usuarios, así como los envíos de notificaciones y su éxito o fracaso, como puede verse en la Figura 58.

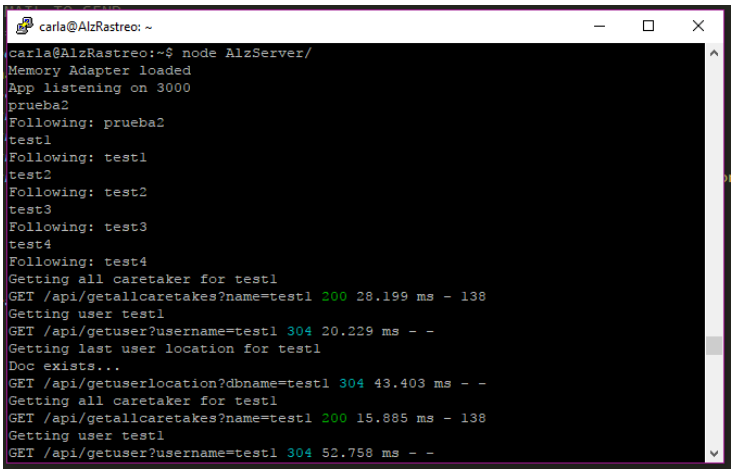

***Figura 58*. Ejecución del servidor web Node.js**

4. FASE DE ESTABILIZACIÓN

4.1. Iteración 1

Se procede a detallar la iteración 1 de la fase de estabilización de la aplicación móvil.

4.1.1. Planificación

En esta fase se integraron todos los módulos que fueron detallados en la fase anterior, junto con el servidor web y la base de datos para la construcción entera del sistema. Vale destacar que, aunque se trató de desarrollar este proyecto de forma modular, muchos módulos eran necesarios para el correcto funcionamiento de otros. Como consecuencia de esto, algunos módulos se desarrollaron en forma paralela. Por ejemplo, el módulo servidor web se fue construyendo a medida que los demás módulos fueron implementados.

4.1.2. Desarrollo y liberación

Para la unión de todos los controladores y servicios de la aplicación, se agregan los archivos de código como dependencias, como puede verse en mejor detalle en la Figura 60.

```
var app = angular.module('AlzRastreo',
    [
        'ionic',
        'superlogin',
        'ngMessages',
        'ngCordova',
        'AlzRastreo.authCtrl',
        'AlzRastreo.commonCtrl',
        'AlzRastreo.mapCtlr',
        'AlzRastreo.patientCtrl',
        'AlzRastreo.caretakerCtrl',
        'AlzRastreo.mapServ',
        'AlzRastreo.commonServices',
        'AlzRastreo.backgroundGeolocationService',
        'AlzRastreo.pushService'
    ]);
```

***Figura 59*. Integración de los módulos en el código fuente de la aplicación móvil**

Una vez determinadas estas dependencias, se procedió a integrar los módulos para la ejecución fluida de la aplicación. De esta forma, se comprobó el funcionamiento después de esta integración y se pudo concluir que todos los módulos, operando como uno solo o en una misma aplicación, funcionaron correctamente.

5. FASE DE PRUEBAS

En esta sección se detallarán las actividades que involucran las pruebas del sistema según la metodología Mobile-D implementada para el desarrollo de la aplicación móvil descrita en este trabajo especial de grado. Las pruebas que se realizaron, específicamente, fueron las pruebas de funcionalidad y las de aceptación, y usabilidad.

5.1.Pruebas de funcionalidad

Las pruebas de funcionalidad del sistema se realizaron con el propósito de verificar la funcionalidad del sistema, tanto de la aplicación móvil como la del servidor web, la entrada y salida de datos, su procesamiento y recuperación.

5.1.1. Registro de usuario

En esta prueba se realizó el registro de un usuario y se procedió a verificar si esta acción tuvo éxito o no. Para el registro de usuario se utilizó el nombre de usuario “test5”, el correo electrónico prueba “test5@test.com” y como contraseña “123456”, mostrado en la Figura 60. Luego se comprobó en la base de datos de usuarios CouchDB la correcta inserción del documento.

Figura 60. **Registro de usuario test5 a la aplicación móvil**

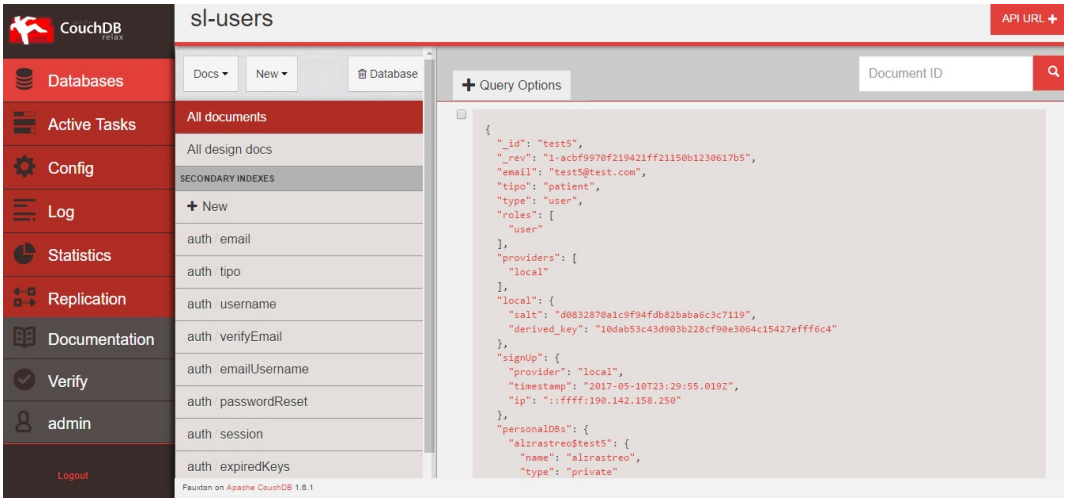

Figura 61. **Base de datos CouchDB de usuarios de la aplicación móvil**

Como muestra la Figura 61, el documento fue agregado con éxito a la base de datos.

5.1.2. Inicio de sesión

En esta prueba se realizó el inicio de sesión del usuario *test5* creado en la prueba anterior. Para llevarla a cabo, se ingresó el nombre de usuario y la contraseña, y se presionó el botón iniciar sesión.

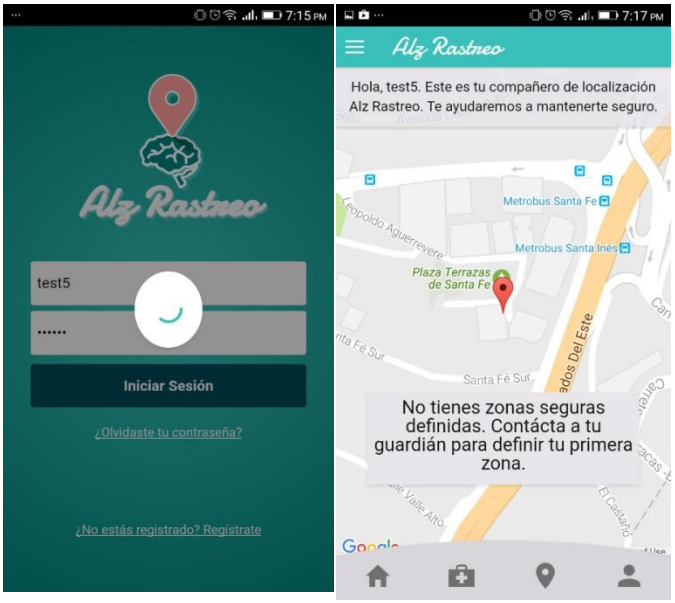

***Figura 62*. Inicio de sesión del usuario "test5" en la aplicación móvil**

Como puede verse en la Figura 62, el inicio de sesión fue exitoso para el usuario de prueba test5. Esto se puso comprobar con el documento de sesión que se creó en la base de datos de sesiones CouchDB.


```
{
  "_id": "org.couchdb.user:DY16v6lyRsOwbVmyW2fIXA",
  "_rev": "1-4f9885ce7cd6088eba258cfb4574725a",
  "password_scheme": "pbkdf2",
  "iterations": 10,
  "type": "user",
  "name": "DY16v6lyRsOwbVmyW2fIXA",
  "user_id": "test5",
  "expires": 1494546312705,
  "roles": [
    "user:test5",
    "user"
  ],
  "derived_key": "1fcf0697d6a10293f40a9402646ac82bb2b40cc4",
  "salt": "72cb404f6a3ed84f19ad7af85ecf66e3"
}
```


***Figura 63*. Documento de sesión en la base de datos de sesiones CouchDB**

5.1.3. Aceptar invitación de guardián, agregar guardián y ver guardianes (usuarios pacientes):

Para esta prueba, primero se procedió a crear un usuario “test4” de tipo guardián y, mediante manipulaciones directas a la base de datos, se creó un documento con el fin de modelar una relación entre “test4” y “test5”, con la cual se garantiza que “test4” haya realizado la petición a “test5”, solicitando ser guardián de éste. Esta relación puede verse en la base de datos representada por la Figura 61.

```
{
  "_id": "test4:test5",
  "_rev": "1-1a0d9923be0f9979a427768b58eed67f",
  "status": "pending",
  "from": "test4",
  "to": "test5",
  "tipo": "friendship"
}
```


***Figura 64*. Documento de invitación de guardián, de "test4" a "test5" en la base de datos CouchDB**

De esta forma, con “test5” se dirigió entonces a la interfaz ver guardianes, en la que se comprobó la aparición de dicha invitación, como se muestra en la Figura 65. Luego se procedió a aceptar la invitación, y como consecuencia de esto el usuario “test5” pasa a tener un guardián relacionado a él, y por defecto, éste se convierte automáticamente en el guardián primario de “test5”. El nuevo documento que representa esta relación entre “test4” y “test5” puede verse en el documento antes presentado, pero con ciertos cambios, como muestra la Figura 66.

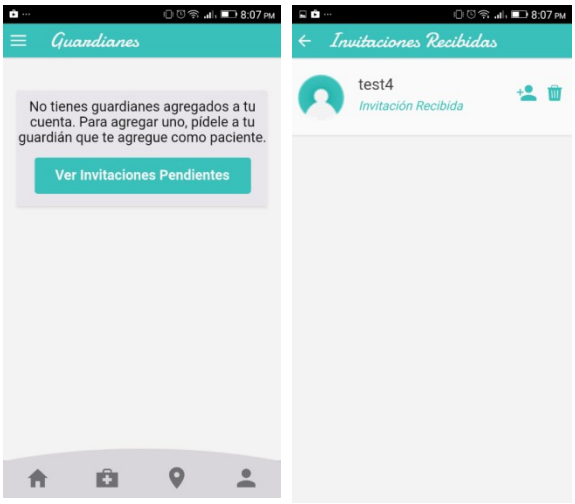

***Figura 65*. Invitación recibida de "test4" para el usuario "test5" en la aplicación móvil**


```
{
  "_id": "test4:test5",
  "_rev": "2-04ece81423e04d3f0e8b814176a4cc5f",
  "status": "accepted",
  "from": "test4",
  "to": "test5",
  "tipo": "friendship",
  "primario": true
}
```


***Figura 66*. Documento de aceptación de usuario guardián "test4" a usuario paciente "test5" en la base de datos CouchDB**

Se procede entonces a ingresar a la interfaz de ver guardianes de “test5”, donde se certifica que en efecto “test4” fue añadido como guardián primario de “test5”, como puede verse en la Figura 67.

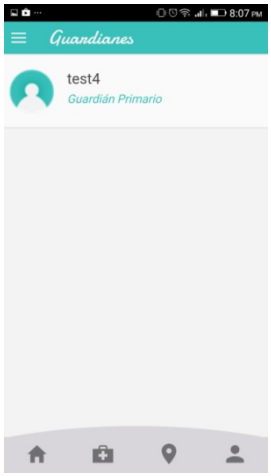

***Figura 67*. Verificación de usuario "test4" agregado exitosamente a usuario "test5" en la aplicación móvil**

5.1.4. Enviar invitación a paciente, buscar paciente, agregar paciente y ver pacientes (usuarios guardianes):

Para esta prueba, se utilizó el usuario guardián “test6”. Con el fin de realizar una invitación, se dirigió a la interfaz de usuario ver pacientes dónde se procedió a presionar el botón agregar paciente. Una vez presionado el botón, se desplegó la interfaz de buscar paciente. En la barra de búsqueda mostrada, se escribió “test5”. Como consecuencia de esto, en el resultado de la búsqueda pudo observarse en efecto el usuario paciente “test5” como puede verse en la Figura 68, posteriormente se procedió a presionar el botón “agregar”. Para comprobar que en efecto se envió la invitación adecuadamente, se navegó hasta la interfaz ver paciente donde se mostró el botón “ver invitaciones pendientes” con éxito, como también muestra la Figura 68.

***Figura 68 a*. Verificar acción de buscar paciente**

***Figura 68 b*. Interfaz de agregar paciente**

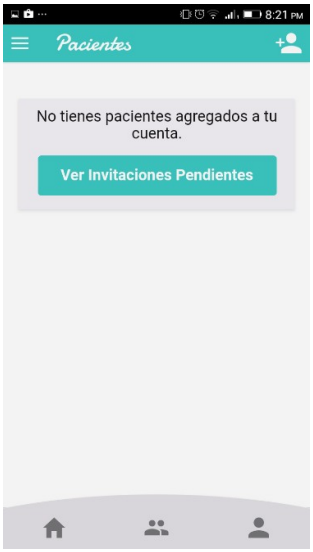

***Figura 68 c*. Interfaz de ver pacientes con el despliegue del botón "ver invitacionespendientes"**

Asimismo, en la Figura 69 se puede ver en la base de datos la inclusión del documento que representa esta relación.


```
{
  "_id": "test6:test5",
  "_rev": "3-8f9cae4b14fadb80f584deb806ef8340",
  "status": "accepted",
  "from": "test6",
  "to": "test5",
  "tipo": "friendship",
  "primario": false
}
```


***Figura 69*. Inclusión del documento relación entre "test5" y "test6" en la base de datos**

Mediante manipulaciones en la base de datos, se procedió entonces a aceptar esta invitación de tal forma que, cuando se ingresa a la interfaz ver paciente, se verifica que éste ya pertenece a la lista de pacientes relacionados con el usuario guardián “test6”, como puede verse en la Figura 70.

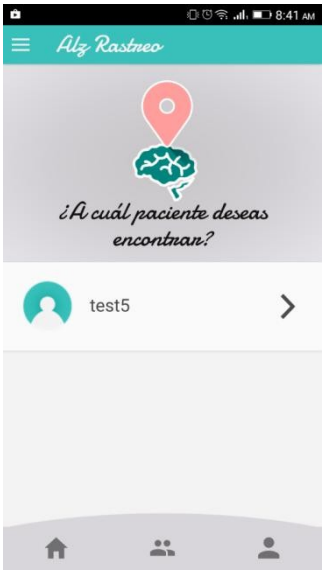

***Figura 70 a*. Verificación de agregación de pacientes "test5" a guardián "test6" en la interfaz de inicio**

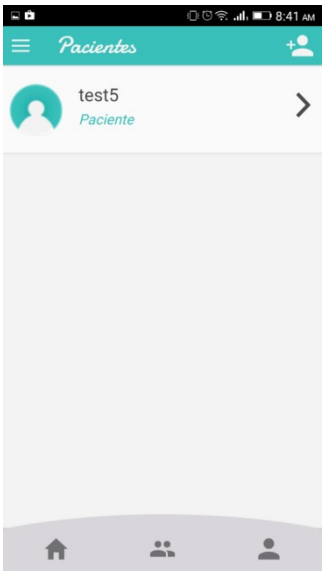

***Figura70 b*. Verificación de agregación de paciente "test5" a guardián "test6" en la interfaz de ver pacientes**

5.1.5. Agregar, eliminar y editar zonas seguras:

Para esta prueba se accedió con el usuario "test4" a la interfaz de rastrear paciente. En ella se accedió a la opción de administrar zonas seguras del paciente "test5". Asimismo, para agregar una zona segura se procedió a presionar el botón "agregar zona segura", el cual redirigió al usuario a la interfaz que puede verse en la Figura 71.

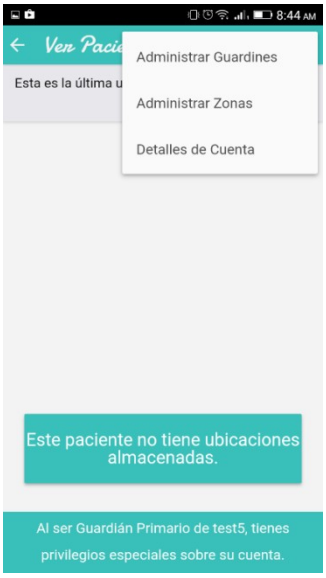

***Figura 71 a*. Verificación de acción agregar nueva zona segura a paciente "test5" en ver pacientes**

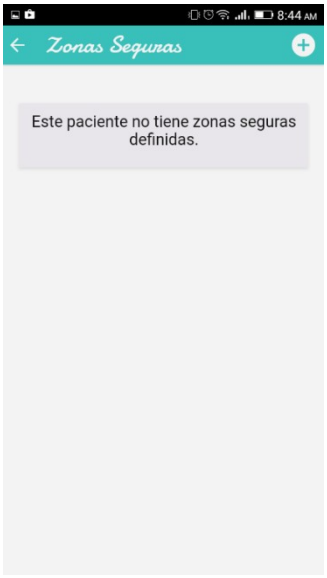

***Figura 71 b*. Verificación de acción agregar nueva zona segura a paciente "test5" en ver zonas seguras**

***Figura 71 c*. Verificación de acción agregar nueva zona segura a paciente "test5" en agregar zonas seguras**

En la interfaz que se desplegó como resultado, se llenaron los campos necesarios que define la zona segura, tras lo cual se presionó el botón “aceptar”, como puede verse en la Figura 71. Para confirmar el éxito de esta operación, se confirmó mediante la interfaz administrar zonas seguras, que esta nueva zona se ha añadió a la lista. En la base de datos puede verse esta zona segura como un documento, la cual está representada la Figura 72.

***Figura 72*. Documento de zona segura de paciente "test5" en la base de datos**

Se presionó luego el botón “administrar zona” y luego “editar zona segura”. Con lo cual se desplegó nuevamente los campos de la zona creada. Se modificó el nombre de la misma, así como el radio de ésta y se presionó aceptar. Para verificar que la zona segura se editó correctamente, se verificó que, en la lista de zonas seguras, la zona creada con el nuevo nombre. En la Figura 73 se pueden ver los cambios en el documento de la base de datos.

Luego se procedió a eliminar la zona segura creada presionando el botón “eliminar”. A continuación, puede verse como la zona segura desaparece de la lista de zonas seguras del paciente, ver Figura 74.

```
{
  "_id": "b8acd664c3a9ace491706690e5382ce4",
  "_rev": "2-ad539374ab95baa5c1bfe294b4bd4093",
  "name": "Casa",
  "lat": "10.48761143966057",
  "lng": "-66.82445883750916",
  "radius": "103",
  "type": "safezone",
  "added_by": "test4"
}
```


***Figura 73*. Documento de zona segura de "test5" editado en la base de datos**

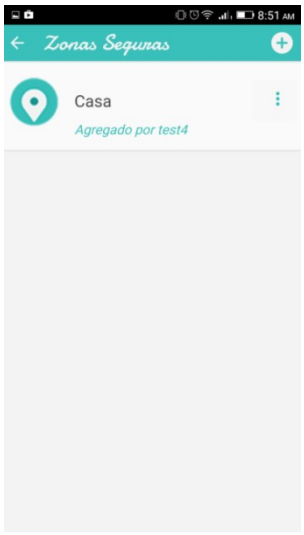

***Figura 74*. Verificación de zona segura editada para paciente "test5" en la aplicación móvil**

5.1.6. Agregar guardianes a un paciente y ceder guardián primario a otro guardián (usuarios guardianes):

Para esta prueba se utilizó el usuario guardián “test4”. Con este fin se procedió a navegar hasta la interfaz administrar guardianes del paciente relacionado “test5” y se aseguró de que se listaran todos los guardianes de este paciente. Para ceder el título de guardián primario se presionó el botón “convertir en guardián primario” en la interfaz de administrar guardianes del paciente “test5”, como puede verse en la figura 75.

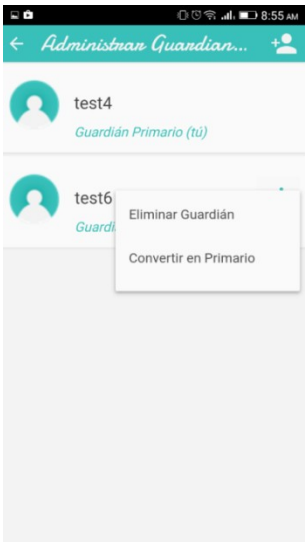

***Figura 75*. Verificar acción convertir en guardián primario a otro guardián en la aplicación móvil**

Efectivamente, una vez presionado el botón, la aplicación redirigió al usuario al inicio, despojando a “test4” de su título de guardián primario. De esta manera, se procedió a presionar rastrear paciente, y se puede ver que ya no existe la posibilidad de administrar el mismo. Estos cambios pueden verse en la base de datos CouchDB representado por la figura 76 y 77.

Figura 76. Documento editado de la relación entre "test4" y "test5" en la base de datos

Figura 77. Documento editado de la relación entre "test5" y "test6" en la base de datos

5.2.Pruebas de usabilidad y aceptación

Con el fin de evaluar la calidad de la aplicación móvil desarrollada en este trabajo especial de grado, se realizó una encuesta de usabilidad y aceptación como instrumento de evaluación. De esta manera, y con los resultados obtenidos, fue posible establecer un análisis cualitativo de los usuarios que fueron entrevistados para demostrar que los requerimientos iniciales de usabilidad

de la aplicación fueron cumplidos. La escala utilizada para medir y conocer el grado de conformidad de los usuarios con la aplicación, fue la escala de Likert, la cual evalúa cada valor de respuesta de la siguiente manera: (1) Totalmente en desacuerdo, (2) En desacuerdo, (3) Ni de acuerdo ni en desacuerdo, (4) De acuerdo y (5) Totalmente de acuerdo.

El artefacto que representa la entrevista puede verse con más detalle en la Tabla 1.

Preguntas	**Respuestas**				
En general, piensa que se presentan de forma intuitiva las principales funcionalidades de la aplicación. Ejemplo: iniciar sesión, registrarse, agregar paciente, etc.	1	2	3	4	5
En general, le parece que son adecuadas para el tema las interfaces de usuario con la que interactúa con la aplicación.	1	2	3	4	5
En general, está satisfecho con la facilidad de realizar las funcionalidades que provee la aplicación.	1	2	3	4	5
En general, está satisfecho con el tiempo empleado en realizar las funcionalidades que provee la aplicación.	1	2	3	4	5
En general, considero que la aplicación utiliza un lenguaje común a través de frases, terminología y conceptos.	1	2	3	4	5
En general, considero que los iconos y términos utilizados en las diferentes interfaces y menús se mantienen homogéneos a través de la aplicación.	1	2	3	4	5
En general, considero que el diseño de la aplicación es una ayuda y permite realizar funciones de forma más rápida o eficiente a través del uso de accesos directos.	1	2	3	4	5
En general, considero que el sistema presenta ayuda y documentación necesaria para la correcta utilización de las herramientas.	1	2	3	4	5

Tabla 1 **Encuesta de usabilidad y aceptación para la aplicación móvil "AlzRastreo"**

La encuesta fue aplicada a 10 personas de diferentes profesiones y con diferentes conocimientos en el área de la computación, y en especial el desarrollo móvil. Entre los profesionales evaluados se encuentran dos Ing. Mecánicos, dos Lic. De Computación inmersos en el área de desarrollo web, una Ing. Químico, una Ing. Electrónico actualmente desarrollando en el área móvil. Las cuatro personas restantes están conformadas por dos amas de casa de edad mayor, y dos estudiantes de la Universidad Central de Venezuela.

5.2.1. Resultados de las pruebas de aceptación y usabilidad

Dadas las preguntas que pueden verse en la Tabla 1, se obtuvieron los siguientes resultados por parte de los usuarios.

Para la pregunta número uno “en general, piensa que se presentan de forma intuitiva las principales funcionalidades de la aplicación”, se obtuvo:

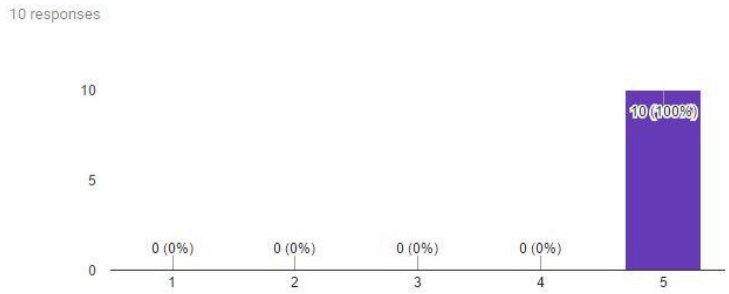

Figura 78. Resultados de la primera pregunta de la encuesta para la prueba de usabilidad y aceptación

En la Figura 78 se puede observar de manera clara que el 100% de los entrevistados consideran que las funcionalidades de la aplicación se presentan de manera intuitiva. Por lo que se consiguió lo que se propuso de forma exitosa.

Para la pregunta número dos “en general, le parece que son adecuadas para el tema las interfaces de usuario con la que interactúa con la aplicación”, se obtuvo:

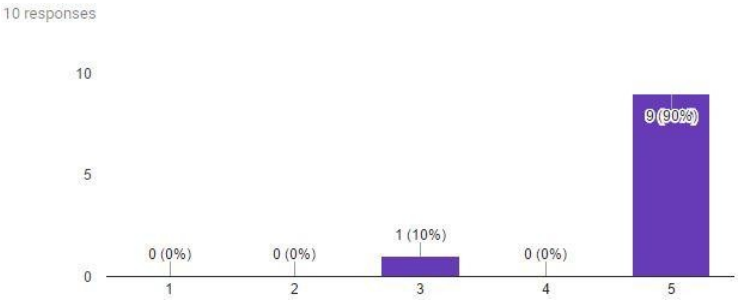

***Figura 79*.Resultados de la segunda pregunta de la encuesta para la prueba de usabilidad y aceptación**

En la Figura 79 puede verse cómo el 90% de los usuarios entrevistados piensan que las interfaces son adecuadas para el tema, sin embargo, un 10% no está ni de acuerdo ni en desacuerdo. Aunque este requerimiento puede considerarse un éxito, se podrá ahondar más en el tema para el futuro para resolver esta incertidumbre.

Para la pregunta número tres “en general, está satisfecho con la facilidad de realizar las funcionalidades que provee la aplicación”, se obtuvo lo siguiente:

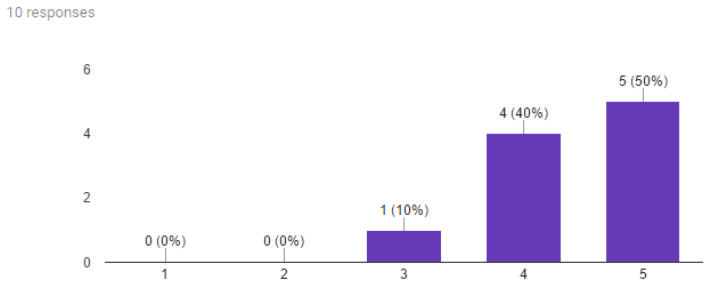

***Figura 80*. Resultados de la tercera pregunta de la encuesta para la prueba de usabilidad y aceptación**

En la Figura 80 pude verse que el 50% de los usuarios entrevistados están totalmente de acuerdo con respecto a la facilidad que provee la aplicación para realizar sus funcionalidades, mientras que el 40% sólo se muestran de acuerdo, y un 10% de los usuarios no se encuentra ni de acuerdo ni en desacuerdo; por lo tanto, aunque se concluye de forma positiva, se perciben indicios claros de que se podría mejorar de forma significativa este tema.

Para la pregunta número cuatro “en general, está satisfecho con el tiempo empleado en realizar las funcionalidades que provee la aplicación.”, se obtuvo:

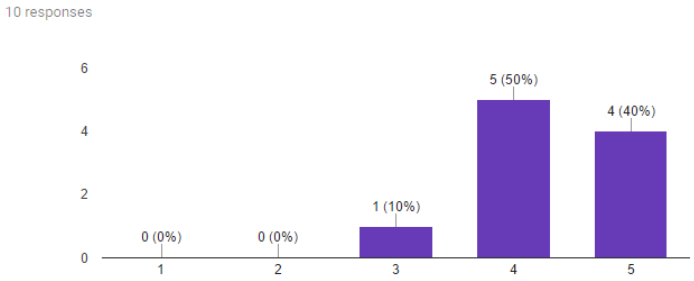

***Figura 81*. Resultados de la cuarta pregunta de la encuesta para la prueba de usabilidad y aceptación**

Como puede observarse en la Figura 81, el 40% de los usuarios se sienten totalmente de acuerdo con el tiempo que les toma realizar las funcionalidades de la aplicación móvil y un 50% se encuentra de acuerdo, por lo que también se puede concluir positivamente. Sin embargo, el 10% no se considera ni en acuerdo ni en desacuerdo, por lo que existe un margen de mejora bastante significativo.

Para la pregunta número cinco “en general, considero que la aplicación utiliza un lenguaje común a través de frases, terminología y conceptos”, se obtuvo:

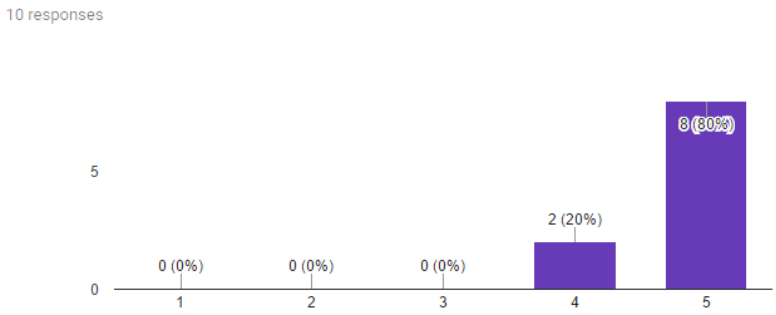

Figura 82. **Resultados de la quinta pregunta de la encuesta para la prueba de usabilidad y aceptación**

De esta manera, la Figura 82 muestra que el 80% de los usuarios entrevistados se encuentran totalmente de acuerdo y consideran que la aplicación móvil muestra de forma correcta sus conceptos y terminologías, lo que se considera un éxito y un cumplimiento de requerimientos. Sin embargo, un 20% sólo se encuentra de acuerdo, por lo que, aunque la prioridad es más baja, también es un tema que se podría ahondar más en futuros trabajos.

Para la pregunta número seis de la entrevista: “en general, considero que los iconos y términos utilizados en las diferentes interfaces y menús se mantienen homogéneos a través de la aplicación”, se obtuvo:

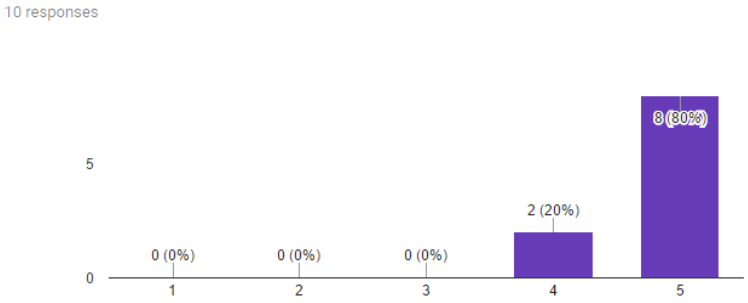

Figura 83. **Resultados de la sexta pregunta de la encuesta para la prueba de usabilidad y aceptación**

En la Figura 83 puede observarse de forma clara cómo el 80% de los usuarios se sienten totalmente de acuerdo al considerar que las interfaces de usuario aplicadas a lo largo de la aplicación se mantienen homogéneas y un 20% se encuentra sólo de acuerdo. Con estos resultados se puede concluir de forma positiva.

Para la pregunta número siete de la entrevista: “en general, considero que el diseño de la aplicación es una ayuda y permite realizar funciones de forma más rápida o eficiente a través del uso de accesos directos”, se obtuvo:

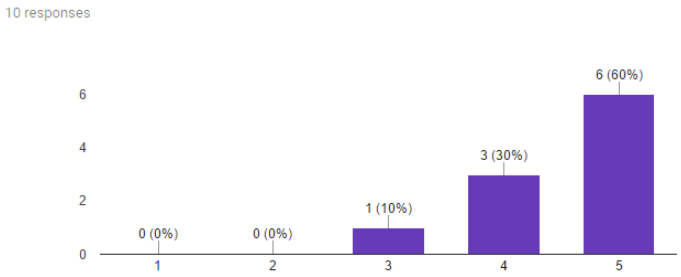

***Figura 84*.Resultados de la séptima pregunta de la encuesta para la prueba de usabilidad y aceptación**

Como se muestra en la Figura 84, el 60% de los usuarios están totalmente de acuerdo con el diseño de la aplicación y sus accesos directos, mientras que un 30% se considera solamente de acuerdo, por lo que también se concluye de forma positiva. Sin embargo, el 10% se consideran ni de acuerdo ni en desacuerdo, por lo que también existe un margen de mejora significativo, dado estos resultados.

Para la última pregunta de la entrevista: “en general, considero que el sistema presenta ayuda y documentación necesaria para la correcta utilización de las herramientas”, se obtuvo:

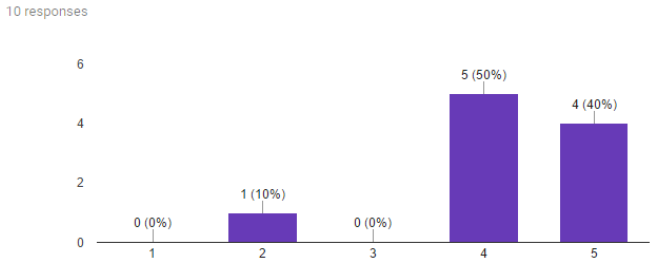

***Figura 85*. Resultados de la octava pregunta de la encuesta para la prueba de usabilidad y aceptación**

En la Figura 85 puede verse que el 10% de los usuarios no están de acuerdo en que la aplicación móvil presenta la documentación necesaria para la correcta utilización de las herramientas, mientras que un 50% se considera de acuerdo y un 40% totalmente de acuerdo. Por esta razón, vale la pena ahondar en estos resultados y mejorar las capacidades de documentación y ayuda presente en la aplicación, en trabajos futuros.

CONCLUSIONES Y TRABAJOS FUTUROS

Mediante el proyecto desarrollado en este trabajo especial de grado, puede concluirse que se cumplió el objetivo principal, y se logró implementar de manera efectiva el desarrollo de una aplicación móvil basada en geolocalización que funcione como acompañante para aquellas personas que padecen de la enfermedad de Alzheimer.

Asimismo, funciona como una solución alternativa para la localización de personas, proveyendo a sus usuarios de más seguridad al evitar los riesgos que involucran la desorientación espacial en pacientes con Alzheimer, notificándole a sus guardianes o familiares cuándo estos se encuentran lejos de un lugar seguro y proporcionando de herramientas para localizarlos y, de esta manera, encontrarlos.

Mediante el desarrollo de esta aplicación, también se logró crear un diseño exitoso, lo que puso demostrarse en las pruebas de usabilidad y aceptación, al hacer uso de una interfaz de usuario usable, intuitiva y sencilla, que está enfocada específicamente hacia personas de edades mayores, brindando así facilidad a la hora de utilizar todas las funcionalidades que ésta provee. De la misma manera, se logró configurar el ambiente de desarrollo de la aplicación, y los componentes necesarios tanto para la aplicación móvil como el servidor web.

También se logró implementar todo el desarrollo de notificaciones a los usuarios guardianes por correo electrónico y notificaciones *push*, para alertarlos de cuándo sus seres queridos o pacientes se encuentren fuera de un lugar seguro. Asimismo, se logró el desarrollo de un servidor web, el cual trabaja como un intermediario entre la base de datos y la aplicación móvil, con el fin de ejecutar las notificaciones antes mencionadas cuando sea pertinente.

Aunque hubo ciertas dificultades en cuanto a conseguir los equipos necesarios para el desarrollo de la aplicación y el servidor web, estas fueron superadas en el proceso del desarrollo. De esta manera se puede concluir de forma positiva que la aplicación móvil desarrollada cumple con los objetivos planteados y satisface los requerimientos funcionales que fueron descritos.

Aun así, existen muchas nuevas funcionalidades que pueden ser implementadas y mejoras en las que se pueden trabajar en futuros trabajos. Entre las nuevas funcionalidades a trabajar en un futuro se contempla:

- La creación de un portal web que permita la visualización de pacientes en un mapa a través de un navegador web.
- Funcionalidades con respecto a utilizar la información capturada que involucra la velocidad de los usuarios y la dirección en la que se dirigen en forma de norte, sur, este y oeste.
- La agregación de la aplicación móvil a Google Play.
- Funcionalidades de un calendario en el módulo de paciente, para la agregación de horas de toma de medicamentos, y la implementación de alertas y recordatorios.

- La agregación de opción de llamadas de emergencia o comunicación por voz a sus guardianes o cuidadores.
- La implementación de funcionalidades de detección de caídas utilizando el acelerómetro del dispositivo móvil.

ANEXOS

1. ANEXO A: INSTALACIÓN DE MICROSOFT VISUAL STUDIO COMMUNITY 2015

Para la instalación de Microsoft Visual Studio Community 2015 es necesario descargar el ejecutable del programa desde la página de Windows [69]. Una vez descargado, se ejecuta el instalador y se procede a seguir los pasos indicados en el *wizard*. Es importante asegurarse de incluir los componentes adicionales del IDE, llamados HTML/JavaScript (Apache Cordova), los cuales se encuentran en la parte de Desarrollo Móvil Multiplataforma, como puede verse en la Figura 86. Esto instalará Apache Cordova más todas las herramientas necesarias para el desarrollo de aplicaciones multiplataformas.

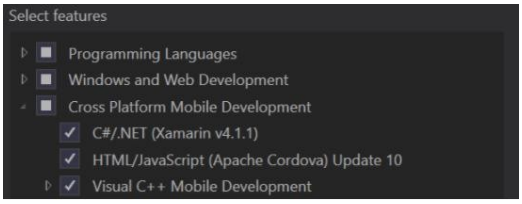

***Figura 86*. Componentes adicionales de Cordova en Visual Studio**

Luego, se escoge “accept” para la correcta instalación de Visual Studio y, una vez hecho esto, se debe crear un nuevo proyecto presionando el botón de “New Project” en el navegador de la aplicación. Entre las opciones que aparecen en el menú izquierdo de la pantalla, se debe ingresar a Online y buscar Ionic en la barra de búsqueda del lado derecho, como puede verse en la Figura 87.

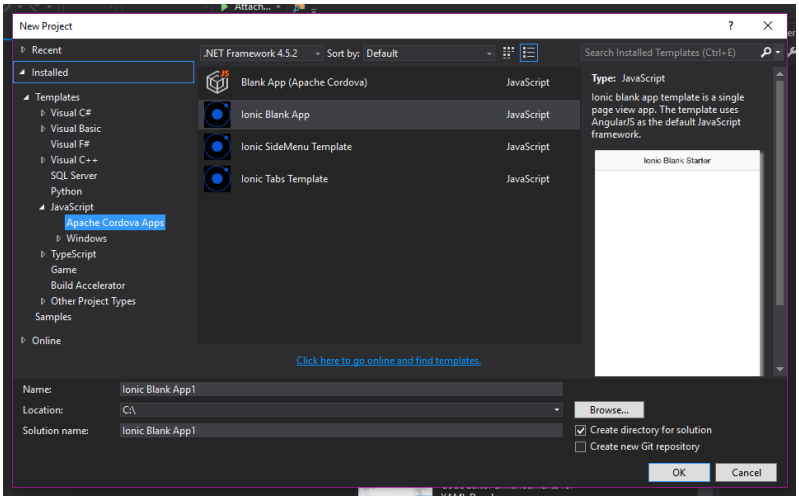

***Figura 87*. Crear nuevo proyecto Ionic en Visual Studio**

Para la aplicación móvil “AlzRastreo”, se utilizó la plantilla llamada Ionic Blank App, ya que posee sólo las funcionalidades mínimas para el desarrollo de aplicaciones en Ionic.

En caso de que se desee abrir una solución Visual Studio previamente creada, se debe ubicar en el navegador ubicado en la parte de arriba de la aplicación y escoger la opción “File”, luego se debe elegir la opción de “Open y allí presionar el botón de “Project”, como puede verse en la Figura 88.

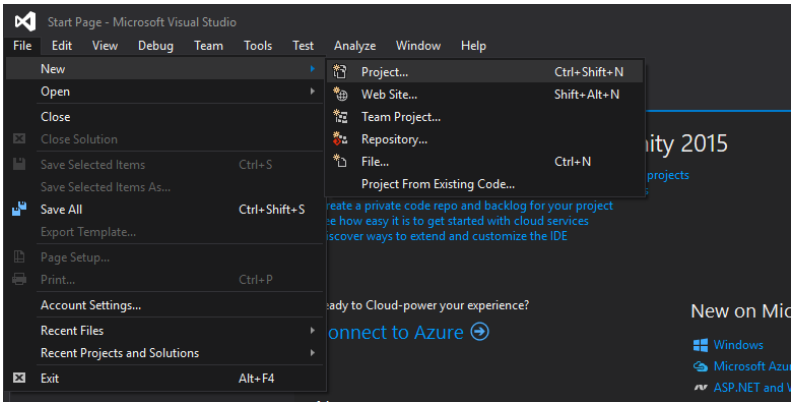

***Figura 88*. Abrir proyecto en Visual Studio**

2. ANEXO B: INSTALACIÓN DE ANDROID STUDIO EN WINDOWS 10

Para instalar Android Studio en Windows, lo primero que debe hacerse ir a la página de Android para descargar el ejecutable del programa para la versión correcta del sistema operativo [70]. Una vez descargado, se procede a seguir los pasos indicados por el Android Studio Setup Wizard, como puede verse en la Figura 89.

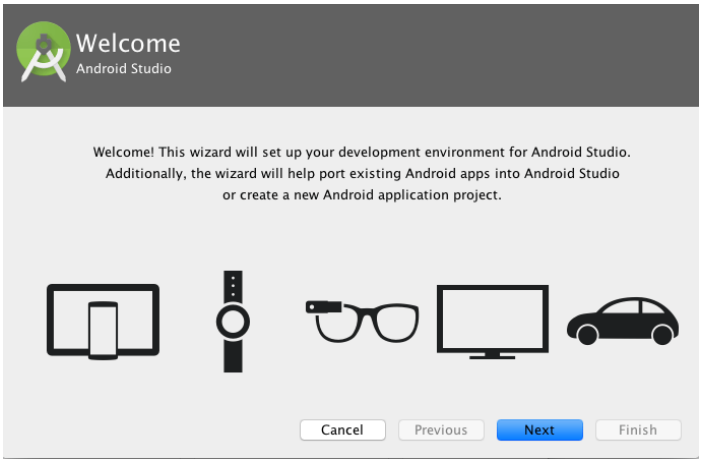

***Figura 89*. Android Studio setuo wizard**

Una vez seguido todos los pasos y presionando el botón "aceptar", y la instalación de Android Studio, es necesario instalar los SDK necesarios en Cordova para la compilación de la aplicación móvil y la construcción del *.apk*. Para esto abrimos el SDK Manager que se instaló junto con Android Studio como puede verse en la Figura 90, y se agregan todos los repositorios y bibliotecas que aparecen en la Figura 91, los cuales son utilizados por la aplicación móvil "AlzRastreo".

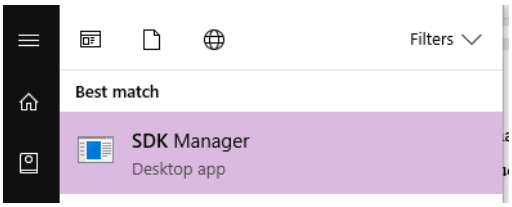

***Figura 90*. Búsqueda del SDK Manager en Windows**

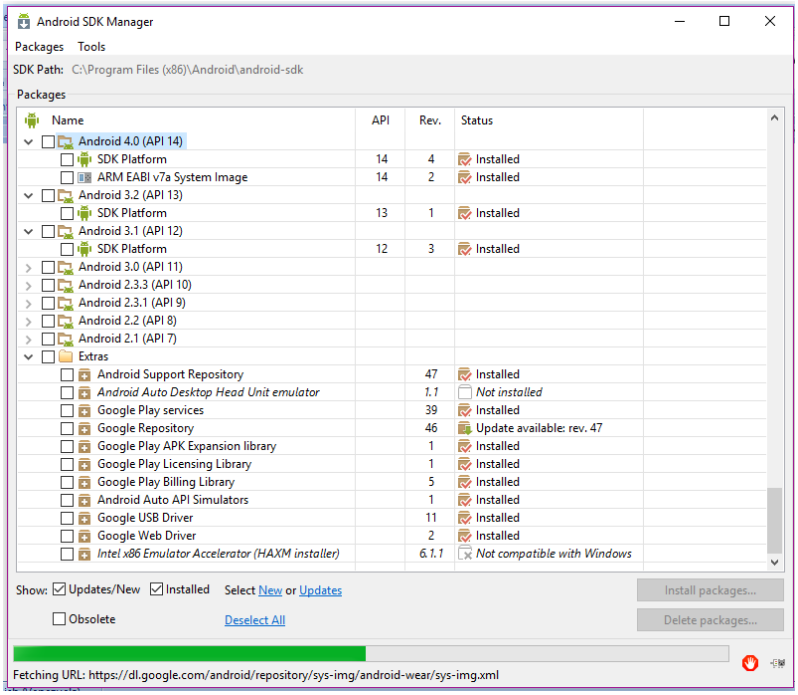

***Figura 91*. Repositorios y bibliotecas utilizados por la aplicación móvil "AlzRastreo"**

3. ANEXO C: INSTALACIÓN DE NODE.JS

Para instalar Node.js en el servidor, es necesario utilizar los comandos de Linux requeridos para la instalación de cualquier programa. Antes de instalarlo debemos asegurarnos que los paquetes necesarios para la compilación de Node.js están instalados, con el comando que puede observarse en la Figura 92.

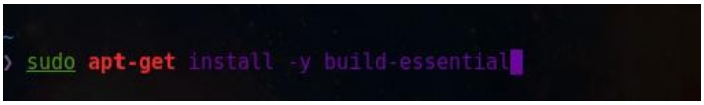

***Figura 92*. Instalar dependencias necesarias para Node.js**

Una vez se instalan los paquetes para su compilación, es necesario agregar Node.js en el repositorio de distribuciones binarias de Debian y Ubuntu, para luego ejecutar el comando de instalación, como se muestra en laFigura93.

***Figura 93*. Comando de instalación de Node.js**

4. ANEXO D: INSTALACIÓN DE COUCHDB

Para la instalación de CouchDB en el servidor, debemos asegurarnos de instalar todas las dependencias necesarias junto con couchDB, como puede verse en la Figura 94.

Figura 94. **Instalación de dependencias necesarias para CouchDB**

Una vez instalados estos paquetes, es necesario ejecutar el comando que muestra la Figura 95 para la instalación de CouchDB y luego el comando que se muestra en la Figura 96, para la construcción de CouchDB.

Figura 95**. Instalación de CouchDB**

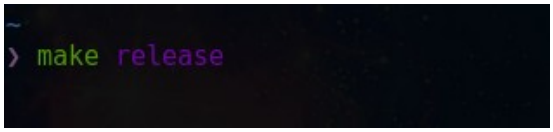

Figura 96**. Construcción de CouchDB**

Una vez instalado CouchDB, es necesario cambiar los permisos de propiedad (ownership) para la correcta configuración de CouchDB con Node.js. Esto se logra con el comando que aparece en la Figura 97.

Figura 97. **Cambios de permisos de ownerships de CouchDB**

5. ANEXO E: Instalar, configurar y administrar el servidor Web

Para instalar el servidor web “AlzRastreo”, es necesario conectarse al servidor a través del protocolo SSH o SFTP en caso de que se encuentre en un lugar remoto, o abriendo el sistema de archivos si se tiene acceso físico. Una vez hecho esto, hay que ubicarse en el home del servidor, y copiar los archivos del servidor web “AlzRastreo” en un directorio llamado AlzServer.

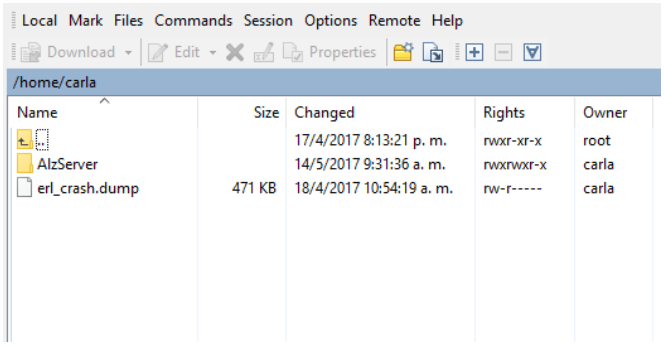

Figura 98. **Creación del directorio AlzServer**

Para la instalación del servidor web, es necesario ubicarse dentro de la carpeta AlzServer y ejecutar el comando que puede verse en la Figura 99. Una vez se instalan todas las bibliotecas, es necesario chequear que CouchDB está siendo ejecutado con Admin Party activado y un usuario administrador llamado “admin” está registrado con la contraseña “adminpassword”. En caso de que Admin Party no esté activado, se debe activar y utilizar las credenciales anteriores para el correcto funcionamiento del servidor.

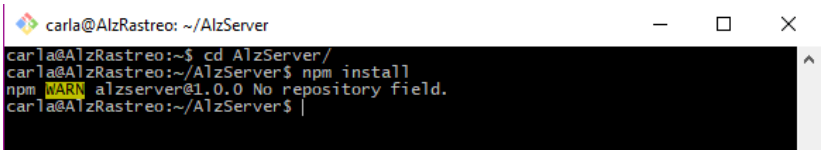

Figura 99. **Instalación de bibliotecas del servidor**

Una vez hecho esto, se debe levantar el servidor web utilizando el comando que muestra la Figura 100.

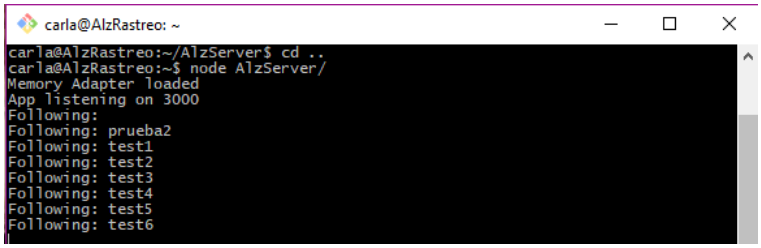

Figura 100. **Comando para levantar el servidor**

El servidor web “AlzRastreo” tiene 3 carpetas principales y un archivo de configuración llamado index.js, como se ve en la Figura 101. Este archivo de configuración es el que define el puerto en el cual el servidor web trabaja, las peticiones HTTP que acepta, configuraciones para la

biblioteca de autenticación llamada SuperLogin, definición de rutas, y eventos necesarios para el monitoreo de las bases de datos de los usuarios de la aplicación móvil.

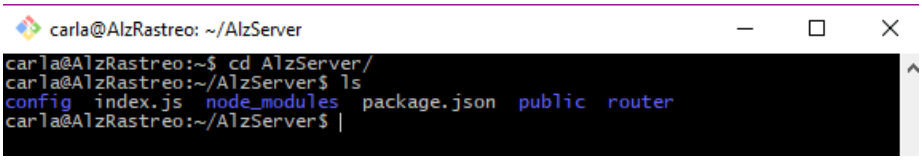

Figura 101. **Lista de directorios y archivos de AlzServer**

En el directorio llamado config está la configuración de la base de datos y donde se define el usuario administrador para tener permisos totales sobre ésta. En ella también se ubican los archivos que involucran el envío de mensajes de correo electrónico y notificaciones push, y los manejadores de eventos necesarios para el envío de estos.

En el directorio llamado router es donde se definen las rutas del servidor web y donde se ubican todas las conexiones, vistas y consultas a la base de datos con el fin de realizar las operaciones CRUD necesarias.

El directorio public es donde se ubican las imágenes y los archivos que son públicos hacia el internet.

REFERENCIAS BIBLIOGRÁFICAS

[1] Alzheimer's Disease International (2016). *World Alzheimer Report 2016: Improving healthcare for people living with dementia*. Recuperado en mayo de 2017, de https://goo.gl/MoxPCW .

[2] Alzheimer's Disease International (2015). *World Alzheimer Report 2015: The Global Impact of Dementia*. Recuperado en abril de 2016, de http://goo.gl/HXf5Pu.

[3] Querfurth H.W, L. F. (2010). Alzheimer's Disease. *The New England Journal of Medicine*, pág. 329–344.

[4] NIA (2011). *Alzheimer's disease genetics fact sheet*. NIH Publication. Recuperado en enero de 2016, de

https://goo.gl/tKs9OE.

[5] Maurer, K., Volk, S., Gerbaldo, H. (1997). Auguste D and Alzheimer's disease. *The Lancet*, volumen 349, número 9064, pág. 1546–1549. DOI: http://dx.doi.org/10.1016/S0140-6736(96)10203-8.

[6] Alzheimer's Association (2016). *ComfortZone – Technology 101*. Recuperado en abril de 2016, de

http://goo.gl/5TximK.

[7] Poushter, J. (2016). *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies*.

Pew Research Center. Recuperado en febrero de 2016, de http://goo.gl/RlXhCI.

[8] Alzheimer Society of Canada (2013*). Locating Devices*. Recuperado en enero de 2016, de http://goo.gl/d8TRsQ.

[9]Abrahamsson, P., Hanhineva, A., Hulkko, H., et al. (2004). Mobile-D: An Agile Approach for Mobile Application Development. *Proceeding OOPSLA '04 Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pág. 174-175. ISBN: 1-58113-833-4. DOI: 10.1145/1028664.1028736.

[10] Gasca, M., Camargo, L., Medina, B. (2013). Metodología para el desarrollo de aplicaciones móviles. *Tecnura*, volumen 18, número 40, pág. 20-35.

[11] Spataru, A. (2010). Agile Development Methods for Mobile Applications. *University of Edinburgh*. Recuperado en abril de 2016, de https://goo.gl/IYosyl.

[12] Mahmood, S., Lu, J. (2013). An Investigation into Mobile Based Approach for Healthcare Activities, Occupational Therapy System. *Proceedings of The International Conference on Software Engineering Research and Practice, SERP13*, pág. 95-101.

[13] Agile Software Technologies Research Programme, VTT. *Mobile-D*. Recuperado en mayo de 2016, de http://goo.gl/4fFFis.

[14] Tweri. *¿WhatisTweti?* Recuperado en mayo de 2016, de http://goo.gl/a7oguU.

[15] Google Play.Ttweri Alzheimer Caregiver Tool (2014). Recuperado en mayo de 2016, de https://goo.gl/IaiFev.

[16] Prince of Songkla University, Information and Communication Technology. Recuperado en mayo de 2016, de http://ict.sci.psu.ac.th/th/.

[17] Google Play. Alzheimer Caretaker (2015). Recuperado en mayo de 2016, de https://goo.gl/Um85tB.

[18] Google Play. Alzheimer Patient (2015). Recuperado en mayo de 2016, de https://goo.gl/7rAj9x.

[19] Devendra Singh Kushwaha, Vikash Kumar Singh. (2015). Comparative Study of Various Cellular Phone Os In Current World. *International Journal of Computer Informatics & Technological Engineering*, vol. 2, núm. 5. ISSN 2348-8557.

[20] Silberschatz, A., Galvin, P., Gagne, G. (2012). Operating System Concepts, 9na edición, Wiley Publishing. ISBN 978-1-118-06333-0.

[21] JayavardhanaGubbi, RajkumarBuyya, et al. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, vol. 29, núm. 7, pág. 1645-1660. DOI 10.1016/j.future.2013.01.010.

[22] Android Source. *The Android Source Code*. Recuperado en mayo del 2017, de https://goo.gl/D6Ah1F.

[23] Android Developers. *Android, the world's most popular mobile platform*. Recuperado en mayo de 2017, de https://goo.gl/IIXu0a.

[24] GSM Association (2014*). Understanding the Internet of Things (IoT).*Recuperadoenfebrero de 2016, de http://goo.gl/tSa2EF.

[25] Android Source. *Android Interfaces and Architecture*. Recuperado en mayo de 2017, de https://goo.gl/7BpO2D.

[26] Yin Yan, Chunyu Chen, KarthikDantu, et al. (2016). Using a Multi-Tasking VM for Mobile Applications. *HotMobile '16 Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pág. 93-98. ISBN 978-1-4503-4145-5. DOI 10.1145/2873587.2873596.

[27] Ahmad, N., Boota, M.W. and Masoom, A.H. (2015). Comparative Analysis of Operating System of Different Smart Phones. *Journal of Software Engineering and Applications*, vol. 8, núm. 3, pág. 114-126. DOI 10.4236/jsea.2015.83012.

[28] Ramírez Vique, R. (2012). Métodos para el desarrollo de aplicaciones móviles, Cataluña.

[29] Jeremy Wilken. (2015). Ionic in Action, Manning Publications.

[30] IBM Software. *Native, web or hybrid mobile-app development*. White Paper. Recuperado en marzo de 2016, de http://goo.gl/aW2FDD.

[31] Lionbridge (2012). Mobile Web Apps vs. Mobile Native Apps: How to Make the Right Choice, White Paper.

[32] Martin G., Suman J. and Vitaly S. (2014). Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks. *NDDS Symp*, pag. 1–15.

[33] Apache Cordova. Overview. Recuperado en mayo de 2017, de https://goo.gl/IuKvSe.

[34] Muñoz G. *Developing multidevice-apps using Apache Cordova and HTML5*. Recuperado en mayo de 2017, de https://goo.gl/BlU68F.

[35] Heitkotter H., Hanschke S. and Majchrzak T. (2013) . Evaluating Cross-Platform Development

Approaches for Mobile Applications. *Springer Berlin Heidelberg,* Author’s version.

[36] Vandecandelaere, B. (2014-2015). Developing TheUdubs-It Platform As A Hybrid App With The Ionic Framework. *New Media And CommunicationTechnology.* Recuperado en mayo de 2017, dehttps://goo.gl/ymQeyL.

[37] Ionic Framework. *Core Concepts*. Recuperado en mayo de 2017, de https://goo.gl/nqTYr9.

[38] Lumsden, A. (2012). *A Brief History of the World Wide Web*.Recuperadoen mayo de 2017, de https://goo.gl/as591y.

[39] Mozilla Developer Network. *HTML basics*. Recuperado en mayo de 2017, de https://goo.gl/wVVTOl. Versión actualizada: Nov 28, 2016.

[40] TutorialPoints, (2012). *HTML: Hyper Text MarkupLanguage*. Recuperado en mayo de 2017, de https://goo.gl/OS5eip.

[41] Eland, T. *An Educator's Introduction to HTML*. Recuperado en mayo de 2017, de https://goo.gl/2CfetN.

[42]Mozilla DeveloperNetwork.*DocumentObjectModel (DOM).*Recuperado en mayo de 2017, de https://goo.gl/mZvncq. Versión actualizada: May 1, 2017.

[43] Mozilla Developer Network. *Glossary: DOM*. Recuperado en mayo de 2017, de https://goo.gl/la7dMu. Versión actualizadaSep 9, 2015.

[44] W Lie, H. (1994). *Cascading HTML stylesheets -- a proposal*. Recuperado en mayo de 2017, de https://goo.gl/aQqjRd.

[45] W3C Recommendation. *Cascading Style Sheet, level 1 (1999).*Recuperado en mayo de 2017, de https://goo.gl/2WWBhu.

[46] Mozilla Developer Network. *How CSS works*. Recuperado en mayo de 2017, dehttps://goo.gl/PQ93M6. Versión actualizadaSep 15, 2016.

[47] W3C Recommendation. *Cascading Style Sheet, level 1 (2008)*. Recuperado en mayo de 2017, de https://goo.gl/WRdHGx.

[48] Mozilla Developer Network. *About JavaScript*. Recuperado en mayo de 2017, dehttps://goo.gl/9v2OXl. Versión actualizadaOct 30, 2015.

[49] Mozilla Developer Network. *A re-introductionto JavaScript (JS tutorial).*Recuperado en mayo de 2017, dehttps://goo.gl/NFZvke. Versión actualizadaMay 1, 2017.

[50] W3C. *Javascript Web Apis*. Recuperado en mayo de 2017, de https://goo.gl/fd0HlR.

[51] AngularJS. *WhatIsAngularJS?* Recuperado en mayo de 2017, de https://goo.gl/1CIEcJ.

[52] Branas, R. (2014). AngularJS Essentials. Packt Publishing, Birmingham, UK..ISBN 978-1-78398-008-6.

[53] Hota, A.K., Madan Prabhu, D. NODE.JS: Lightweight, Event driven I/O web development. *Informatics*, pag. 30-31, 2014.

[54] Training. *About Node.js, and why you should add Node.js to your skill set?*Recuperadoen mayo de 2017, de https://goo.gl/ZsamwK.

[55] Node.js. *Node.js FoundationSurveysResults*. Recuperado en mayo de 2017, de https://goo.gl/stJx1D.

[56] Moniruzzaman, A. B. M., and Hossain S. A. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application* vol. 6, núm. 4. 2013.

[57] PokornýJ. (2013). NoSQL Databases: a step to database scalability in Web environment. *International Journal of Web Information Systems*, vol. 9, núm. 1, 2013 pág. 69-82. DOI 10.1108/17440081311316398.

[58] Microsoft Azure. *NoSQL Vs. SQL Comparison*. Recuperado en mayo de 2017, de https://goo.gl/Tmv2Yz.

[59] Techopedia. *Why NoSQL Trumps Relational Databases for Mobile Applications*. Recuperado en mayo de 2017, de https://goo.gl/TceZmb. Versión actualizada: May 4, 2016.

[60] Apache CouchDB. *CouchDB*. Recuperado en mayo de 2017, de https://goo.gl/zIHp99.

[61] Jackson, J. (2010). *CouchDB NoSQL Database Ready for Production Use*. Recuperado en mayo de 2017, de https://goo.gl/xx4tNz.

[62] Apache CouchDB. *WhyCouchDB?.* Recuperado en mayo de 2017, de https://goo.gl/AzN4OB.

[63] Apache CouchDB Documentation. *Technical Overview*. Recuperado en mayo de 2017, de https://goo.gl/dfZp3s.

[64] Anderson, C., Lehnardt, J. andSlater, N. *CouchDB - The Definitive Guide.* DesignDocuments. Recuperado en mayo de 2017, de https://goo.gl/76E9Qz.

[65] Anderson, C., Lehnardt, J. andSlater, N. CouchDB - The Definitive Guide. WhyCouchDB?.Recuperado en mayo de 2017, de https://goo.gl/27aNzm.

[66] Apache CouchDB. *Eventual Consistency*. Recuperado en mayo de 2017, de https://goo.gl/a9NlTo.

[67] Apache CouchDB. *GettingStarted*. Recuperado en mayo de 2017, de https://goo.gl/3yiyxj.

[68] PouchDB. *IntroductiontoPouchDB.* Recuperado en mayo de 2017, de https://goo.gl/3Jqi6T.

[69] Microsoft. Descargas de Visual Studio. Recuperado en mayo de 2017, de https://goo.gl/Ymv6c4.

[70] Android Studio. Android Studio IDE oficial para Android. Recuperado en mayo de 2017, de https://goo.gl/kMkh5r.