PA133-05

SISTEMA MÓVIL DE SONDEO PREVENTIVO DE VEHÍCULOS CON SOPORTE OBDII PARA MEJORAR LA VIDA ÚTIL DEL AUTOMOTOR

ROLAND MAURICIO CRUZ VELANDIA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2014

PA133-05
SISTEMA MÓVIL DE SONDEO PREVENTIVO DE VEHÍCULOS CON SOPORTE OBDII PARA MEJORAR LA VIDA ÚTIL DEL AUTOMOTOR

Autor:

Roland Mauricio Cruz Velandia

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TITULO DE

MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director

Juan Pablo Garzón Ruiz

Comité de Evaluación del Trabajo de Grado

Ing. Rafael V. Páez Méndez PhD

Ing. Ricardo González PhD

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~PA133-05-PMovVidaAutomotor/

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERIA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.

Diciembre 2014

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS

Rector Magnífico

Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Luis David Prieto Martínez

Decano del Medio Universitario Facultad de Ingeniería

Padre Sergio Bernal Restrepo S.J.

Director Maestría en Ingeniería de Sistemas y Computación

Ingeniero Enrique González Guerrero

Director Departamento de Ingeniería de Sistemas

Ingeniero César Julio Bustacara Medina

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Le agradezco a Dios por estar siempre conmigo durante todas las etapas de mi vida y por bendecirme con la compañía de mi Abuela Rosaura Marín Peña

A mi abuela Rosaura Marín Peña le agradezco por haber sido madre, padre y amiga incondicional, por haberme albergado cuando más lo necesite y por siempre apoyarme sin pedir nada a cambio.

A mi tío Alirio Cruz por su apoyo durante este camino y por darme la fortaleza necesaria para seguir adelante.

A mi Prometida, Victoria Clavijo porque siempre me motivó, alentó y acompañó en este recorrido por la maestría.

A mis amigos gracias por su amistad verdadera, no son muchos, pero son muy valiosos y sus consejos, guías y motivaciones siempre son un aliciente para superarme día tras día.

Al Ingeniero Juan Pablo Garzón Ruiz, por la colaboración y guía brindada durante este proceso, por escucharme y aconsejarme.

Y a todos aquellos que de una u otra forma me han acompañado durante este viaje por la vida y en especial por la Maestría.

Gracias a todos.

TABLA DE CONTENIDO

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

ABSTRACT

Most vehicles manufactured after 1996 are governed by the OBDII regulation, which was created by the SAE to monitor various components of automobile and save the report of the troubles found, in order to maintain control of emissions between the per-mitted limits by CARB and EPA.

This allows that using hardware interfaces, you can capture the information of vehicle diagnostic system, however, although there are applications that connect to these systems, are limited to display information and does not offer an additional treatment thereof. Therefore, it was proposed to create a system that can send relevant data to the cloud so that skilled personnel perform the analysis thereof and, if necessary, generate a feedback to the user, aimed at extend the vehicle life.

RESUMEN

La mayoría de vehículos fabricados después del año 1996 se rigen por la regulación OBDII[1], la cual fue creada por la SAE[2] para monitorear diferentes componentes del automotor y guardar el informe de los fallos detectados en aras de mantener el control de emisiones en los límites permitidos por la CARB[3] y la EPA[4].

Esto permite que usando interfaces de hardware, se pueda capturar la información del sistema de diagnóstico del vehículo, sin embargo, aunque existen aplicaciones que se conectan con dichos sistemas, se limitan a mostrar la información y no ofrecen un tratamiento adicional de la misma. Por ello, se planteó crear un sistema que permita enviar los datos relevantes a la nube para que personal experto, realice el análisis de los mismos y de ser necesario, genere una retroalimentación al usuario, orientada a alargar la vida útil del vehículo automotor.

[1] OBDII: On Board Diagnostic – Version II (Diagnostico A Bordo - versión II)

[2] SAE: Society of Automotive Engineers (Sociedad de Ingenieros Automotrices)

[3] CARB: California Air Resources Board (Comisión de Recursos del Aire de California)

[4] EPA: Environmental Protection Agency (Agencia de Protección del Medio Ambiente)

RESUMEN EJECUTIVO

La regulación OBDII dio lugar a una potente herramienta enfocada principalmente a monitorizar los componentes del vehículo para que estos funcionen correctamente y así reducir las emisiones de CO2 [1]. La finalidad de la regulación es la de cumplir una función preventiva en lugar de correctiva, pero lamentablemente, su utilización suele estar limitada en gran medida a los talleres mecánicos y a personal calificado, debido a que la mayoría de información censada se mantiene aislada (salvo la mostrada por el vehículo mediante los indicadores LED del tablero de instrumentos) hasta que el vehículo sea monitorizado en un sitio especializado.

Esto implica que en la mayoría de los casos las acciones terminan siendo correctivas, por la ausencia de un sistema de alerta temprana sobre los daños reportados por el sistema, los cuales suelen ser detectados recién cuando el usuario acerca su vehículo a revisión en un taller. Esta situación conlleva implicaciones económicas y medioambientales por la circulación en mal estado del vehículo, todo ello a causa de defectos que se originaron con bastante tiempo de anterioridad, pero que no fueron reportados a tiempo

En este sentido se han creado aplicaciones, muchas de ellas orientadas a su uso en dispositivos móviles, las cuales capturan y muestran al usuario cierta información de forma amigable, ya sea de datos de telemetría (velocidad, revoluciones por minuto, nivel de combustible, temperaturas, entre otras) o asociada a fallas reportadas en el vehículo. Ejemplos de éstas aplicaciones son Torque (OBD2 & Car) [3] o DashCommand (OBD ELM Scanner) [4].

Sin embargo, aunque éstas y muchas otras aplicaciones significan un paso adelante orientado a acercar la información de los sistemas OBDII al usuario para mantenerlo al tanto del estado de su vehículo, se vuelve a la raíz del problema a causa de que en lo referente a los fallos detectados, un usuario común puede no estar en capacidad de interpretar la información del error adecuadamente; o de darle la importancia que puede llegar a tener para el buen funcionamiento del automotor, con lo cual se hace nuevamente necesaria la intervención y el concepto de un usuario experto.

Con ello en mente, éste proyecto se enfocó en brindar una solución informática que le permitiera a los usuarios de vehículos con sistemas OBDII, contar con una herramienta que adicional a las características de captura y visualización de los datos de telemetría básicos y de códigos de Falla arrojados por el sistema de diagnóstico a bordo de los vehículos, hiciera uso de la facilidad de transmisión de datos de los dispositivos móviles de hoy en día para enviar a un Portal Web en la nube, todos los códigos de falla detectados, para que por medio del Portal un usuario experto en la interpretación de los mismos, pudiera emitir un concepto o de ser necesario, solicitar la ejecución de comandos adicionales orientados a determinar el problema que originó el código de falla detectado.

La solución construida nombrada **doc2car** se compone de dos partes fundamentales: por un lado se desarrolló un prototipo de aplicación móvil (creada para su uso en dispositivos Android)

orientada a los conductores, la cual tiene la capacidad de conectarse con el sistema OBDII mediante el uso de una interfaz de hardware Bluetooth. Esta aplicación permite la captura y visualización de ciertos datos básicos tales como: velocidad actual, revoluciones por minuto, nivel de combustible, temperatura del líquido refrigerante, entre otros. Así mismo, la aplicación móvil tiene la funcionalidad que le permite consultar los códigos de fallas residentes en el vehículo y en caso de existir, enviarlos a un repositorio de Base de Datos alojado en la nube.

Por otro lado, se creó el prototipo del Portal Web usando el mismo proveedor de servicios en la nube que se utilizó para almacenar los datos que maneja el sistema móvil. Dicho portal cuenta con los módulos básicos autenticación, autorización, visualización de fallas reportadas, emisión de conceptos y envío de notificaciones a los conductores. La finalidad del portal es servir de puente de comunicación entre el conductor y el usuario experto, ya que por medio de los módulos disponibles, el usuario experto podrá revisar los códigos de falla y con base en sus conocimientos proceder a solicitar información adicional al conductor y/o pedir autorización para la ejecución de comandos en el vehículo de forma remota.

Un componente vital en el sistema es el mecanismo de comunicación entre el usuario experto y el conductor y para ello se utilizó el servicio de notificaciones que hace uso de la tecnología Push creada por Apple [5] y que es soportada por el proveedor de servicios en la nube. Este método permite que las notificaciones puedan ser creadas y enviadas desde el portal Web y recibidas por la aplicación móvil del conductor, donde ésta última tiene la capacidad de ejecutarse en segundo plano o de iniciarse al recibir dichas notificaciones para así mostrar la información enviada desde el portal. Adicionalmente, la aplicación móvil puede almacenar los comandos solicitados por el experto para ejecutarlos la próxima vez que el conductor enlace la aplicación móvil con la interfaz de hardware OBDII de su vehículo.

Como resultado se pueden concluir que gracias a las ventajas tecnológicas de hoy en día, el prototipo cuenta con gran potencial de crecimiento. Todo ello a raíz de la reducción de la brecha existente entre los conductores y los usuarios expertos, ya que con el sistema desarrollado es posible llevar una monitorización continua de los vehículos reduciendo así, la posibilidad de movilizar un automotor defectuoso sin que el conductor se percate del fallo.

Adicionalmente se resalta que aunque el prototipo se enfocó en la detección, captura y análisis de códigos de falla, el sistema puede ser fácilmente modificado para enfocarse en otros nichos que requieran información diferente de los sistemas OBDII, tales como los datos de telemetría, los cuales le serían de gran ayuda a entidades como aseguradoras, o empresas de alquiler de vehículos a la hora de llevar un seguimiento de desgaste de su planta automotriz. Además, el sistema se puede orientar incluso para ser usado como sistema de seguimiento y control para entidades gubernamentales o medioambientales.

Como conclusión se puede decir que gracias a los avances tecnológicos en materia de comunicaciones, es posible integrar diversos sistemas para enriquecer sus funcionalidades, permitiendo así brindar mejores servicios a los usuarios.

INTRODUCCIÓN

Este proyecto lleva como título "Sistema Móvil de Sondeo Preventivo de Vehículos con Soporte OBD-II para Mejorar la Vida Útil del Automotor", y se desarrolló dentro del grupo de investigación ISTAR del Departamento de Ingeniería de Sistemas de la facultad de Ingeniería de la Pontificia Universidad Javeriana.

La idea que originó el proyecto surgió a raíz de la observación y el análisis de algunas aplicaciones que muestran información capturada por los sistemas OBDII (Ej.: Torque OBD2 & Car [3] o DashCommand OBD ELM Scanner [4]), dado que aunque dichas aplicaciones son bastante completas y muestran diversa información al usuario, en lo referente a los códigos de falla, la interpretación de los mismo están restringida al conocimiento que el conductor tenga sobre el tema. Así que por lo general el conductor debe llevar el vehículo al taller para que un usuario experto emita su concepto.

Por consiguiente, lo que se quiere lograr es reducir la brecha que existe entre la detección de una falla en un vehículo y el tiempo que pasa hasta que dicha falla es analizada por un usuario experto. Para contrarrestar esta falencia se desarrolló un Sistema compuesto por una aplicación móvil y un portal Web residente en la nube, el cual toma los datos relevantes asociados a códigos de falla capturados por el Sistema OBDII, para luego enviarlos a la nube en tiempo real usando la red de datos de un Smartphone, facilitando así que un usuario experto pueda ingresar al portal Web y analizar los códigos capturados y emitir un concepto al respecto o solicitar incluso la ejecución de más comandos al sistema OBDII. Todo ello orientado a llevar un seguimiento y análisis en tiempo real y así evitar problemas más graves a futuro en los automóviles

Para explicar el proceso realizado a lo largo del proyecto, este documento se ha desglosado en los siguientes capítulos:

- En el capítulo uno se realiza una descripción del problema, y se presenta la relevancia que tiene el desarrollo del proyecto. En este capítulo se da a conocer el objetivo general del proyecto, los objetivos específicos, la metodología utilizada, y las fases metodológicas del mismo.
- El capítulo dos contempla el marco teórico de este proyecto y se presentan los conceptos más relevantes utilizados a lo largo del mismo.
- El capítulo tres describe todo el proceso de desarrollo, incluyendo las etapas de definición de clientes, selección de las herramientas y el diseño de la solución. Así mismo se presentan los detalles de las etapas de codificación y se muestra el resultado del sistema desarrollado.
- El capítulo cuatro abarca la etapa de pruebas y validación tanto de la aplicación móvil como del portal Web.
- El capítulo cinco se enfoca en las conclusiones del trabajo realizado y en las posibles extensiones a realizarse en trabajos futuros.
- Posteriormente, en el capítulo seis se mostrará la bibliografía referenciada durante el desarrollo del proyecto.
- Por último, se enumeran los Anexos de este documento.

I – DESCRIPCIÓN DEL PROYECTO REALIZADO

Para comenzar se realizará una presentación del problema abarcado mostrando la relevancia que tiene el desarrollo del proyecto. Posteriormente se darán a conocer el objetivo general, los objetivos específicos, la metodología utilizada y las fases metodológicas del mismo.

1. Presentación del problema

La creciente emisión de Gases de Efecto Invernadero (GEI) y en especial de Dióxido de Carbono (CO2), han acelerado el cambio climático en nuestros días. En éste sentido, existen evidencias que indican que la mayor parte del calentamiento global ha sido causado por las acciones humanas, dado que gran parte de las actividades que realizamos día a día; tales como movilidad, alimentación y ocio, entre otras, implican una utilización de energía, lo que a su vez significa una mayor emisión de estos gases en la atmósfera [6].

Con éste problema en mente, se creó la iniciativa conocida como HUELLA DE CARBONO, la cual tiene como finalidad cuantificar la cantidad de emisiones de GEI, medidas en emisiones de CO2 equivalente, que son liberadas a la atmósfera debido a las actividades humanas cotidianas o a la comercialización de un producto o servicio. [7]

Precisamente, un sector clave que contribuye a incrementar el valor de la HUELLA DE CARBONO, es el creciente sector automotriz, el cual en 2012 vendió 315.968 automóviles, solo en el territorio Colombiano [8]. Esto, debido a que los automóviles se convirtieron en indispensables para el desarrollo de nuestra sociedad y actualmente forman parte directa o indirecta de nuestras actividades habituales.

En 1988, la "California Air Resources Board" (CARB) determinó que todos los automóviles a gasolina comercializados en Estados Unidos deberían contar con un sistema de diagnóstico que fuera capaz de monitorizar algunos de los componentes controladores de emisiones en los autos, para mantener las emisiones dentro de los límites permitidos por la ley. Ésta regulación, conocida como Sistema de Control de a Bordo I (OBDI: On Board Diagnostics) fue obligatoria para todos los vehículos en Estados Unidos a partir de 1991. Sin embargo, los sistemas OBDI no eran tan efectivos porque solamente monitoreaban algunos de los componentes relacionados con las emisiones. Así que en 1996, se creó la segunda versión conocida como OBDII, la cual estaba encaminada a cumplir con medidas un poco más estrictas [9].

Debido a esto, la mayoría de vehículos recientes cuentan con sistemas bajo la regulación OBDI u OBDII enfocados principalmente a monitorizar los componentes del vehículo para que funcionen correctamente y así reducir las emisiones de CO2. En resumen se puede decir que OBDI y OBDII son regulaciones cuya finalidad primordial es cumplir una función preventiva en lugar de correctiva, pero lamentablemente, su utilización solía ser aprovechada exclusivamente por los talleres mecánicos especializados y por personal calificado con el hardware necesario para acceder a dichos sistemas. Por ende, la mayoría de información censada se mantenía aislada

(salvo la mostrada por el vehículo mediante los indicadores LED del tablero de instrumentos) hasta que el vehículo fuera monitorizado en un sitio especializado.

Esta limitación suele afectar económicamente a los conductores, dado que los costos en los que puede llegar a incurrir un conductor cuando una falla no es detectada y avisada a tiempo, pueden ser mayores a si dicha falla es detectada y resuelta oportunamente.

Sin embargo, hoy en día es posible acceder a la información de los Sistemas OBDII de una forma más práctica y al alcance de los usuarios, ya que por medio de una interfaz de hardware relativamente económica (50-100USD) [10] [11] la cual se conecta directamente al vehículo, se pueden enviar los datos OBDII usando diferentes medios de comunicación (USB, WI-FI, Bluetooth). Haciendo uso de estas interfaces se han creado algunas aplicaciones, muchas de ellas orientadas a su uso en dispositivos móviles, caracterizándose por capturar y mostrar al usuario la información obtenida de una forma amigable. Ejemplos de éstas aplicaciones son Torque (OBD2 & Car) [3] o DashCommand (OBD ELM Scanner) [4], entre muchas otras.

No obstante, aunque éstas aplicaciones significan un paso adelante orientado a acercar la información de los sistemas OBDII al usuario para mantenerlo al tanto del estado del vehículo, en lo referente a los fallos detectados, un usuario común puede no estar en capacidad de interpretar la información del error adecuadamente; o de darle la importancia que puede llegar a tener para el buen funcionamiento del automotor, con lo cual se hace nuevamente necesaria la intervención y el concepto de un usuario experto.

Así pues, el problema de estudio que se plantea resolver en éste proyecto, es el de incrementar la vida útil de los automotores, y de forma indirecta, aportar a la reducción de las emisiones contaminantes, mediante la reducción de la brecha entre el tiempo que le toma a un sistema OBDII detectar una falla y el tiempo que un usuario experto puede analizarla y emitir un concepto.

2. Presentación de la solución

La solución planteada consistió en la creación de un sistema compuesto por dos aplicaciones. Por un lado, se construyó una aplicación móvil con capacidad de recibir notificaciones desde Internet y que sea capaz usar una interfaz de hardware OBDII para conectarse, capturar y realizar seguimiento periódico a la información asociada a los códigos de falla detectados por el sistema OBDII. Así mismo, cada vez que se detecten fallos, estos son enviados en tiempo real a un repositorio de datos suministrado por un proveedor de servicios en la nube.

Por otro lado, se desarrolló un portal Web hospedado por el mismo proveedor de servicios en la nube el cual está pensado para que un usuario experto pueda acceder a la información de códigos de falla reportados por la aplicación móvil. De esta forma, el usuario experto tiene la posibilidad de analizar dichos fallos y de ser necesario emitir un concepto sin la necesidad de acercar el vehículo al taller. En éste sentido, fue vital el soporte para enviar notificaciones desde la nube y recibirlas por parte de la aplicación móvil, debido a que éste fue el medio de transmisión usado por el usuario experto para comunicar su concepto al conductor

En la siguiente figura se puede apreciar el esquema de la solución planteada:

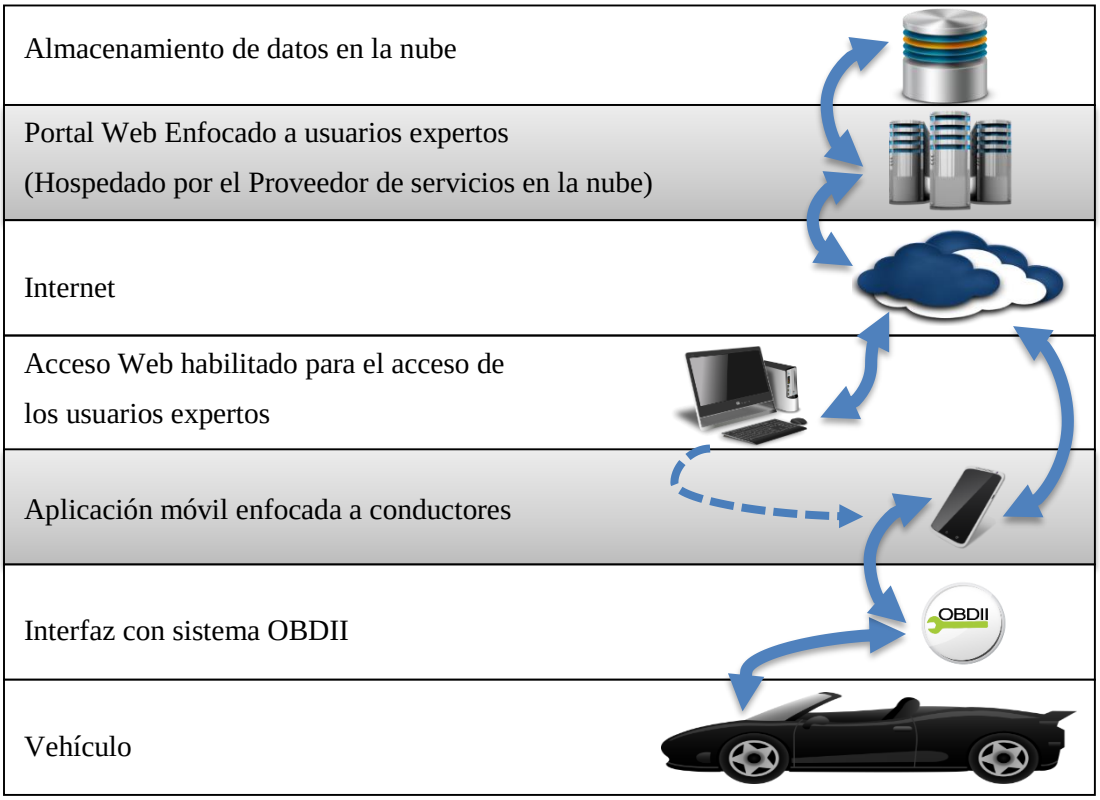

Figura 1: Esquema de la solución planteada

3. Objetivos

De acuerdo a la problemática identificada y a la solución planteada se han definido para este proyecto un objetivo general y cinco objetivos específicos, los cuales se enuncian a continuación.

3.1 Objetivo General

- ✓ Generar una solución informática enfocada a automóviles con soporte OBDII, que aproveche las ventajas tecnológicas y de comunicación de los Smartphone actuales, para publicar información en Cloud Computing que pueda ser monitorizada, permitiendo una detección temprana de fallos mecánicos.

3.2 Objetivos específicos

1. Identificar las funcionalidades enfocadas a detección de fallos de la regulación OBDII que serán publicadas y monitorizadas en Cloud Computing

2. Determinar un protocolo de comunicación entre el Smartphone y el sistema en Cloud Computing que brinde rapidez y seguridad
3. Diseñar la arquitectura informática del sistema a implementar.
4. Implementar la solución informática que cumpla con la funcionalidad definida.
5. Validar la solución informática, mediante la verificación de los requerimientos definidos por parte de un experto.

4. Metodología

Las metodologías ágiles poseen ciertas propiedades que las hacen totalmente aplicables al desarrollo de proyectos de software móvil. En [12] se identifican los métodos ágiles como la solución potencial para el desarrollo de software en dispositivos móviles.

Por esta razón, se ha decidido hacer uso de la metodología Mobile-D [13], la cual fue creada apoyándose en muchas otras soluciones bien conocidas y consolidadas: eXtreme Programming (XP) [14], Crystal methodologies [15] y Rational Unified Process(RUP) [16]. En resumen, los principios de programación extrema se han reutilizado en lo referente a las prácticas de desarrollo, las metodologías Crystal proporcionan un input muy valioso en términos de la escalabilidad de los métodos y RUP es la base para el diseño completo del ciclo de vida.

Sin embargo, se harán un par de cambios debido a las características propias del proyecto, los cuales implican agregar una etapa inicial, previa a las definidas en Mobile-D, en la cual se hará el levantamiento de la información necesaria para dar inicio al proyecto. Adicionalmente, la etapa de "Estabilización" se usará con un propósito diferente al concebido en la metodología, ya que en ésta se usa para integrar y estabilizar desarrollos de equipos de trabajo separados geográficamente.

Las etapas contempladas para el desarrollo del proyecto son: Levantamiento de información, Exploración, Inicialización, Producción, Estabilización y Pruebas. La Figura 2 ilustra las fases y etapas que contempla la metodología a usar.

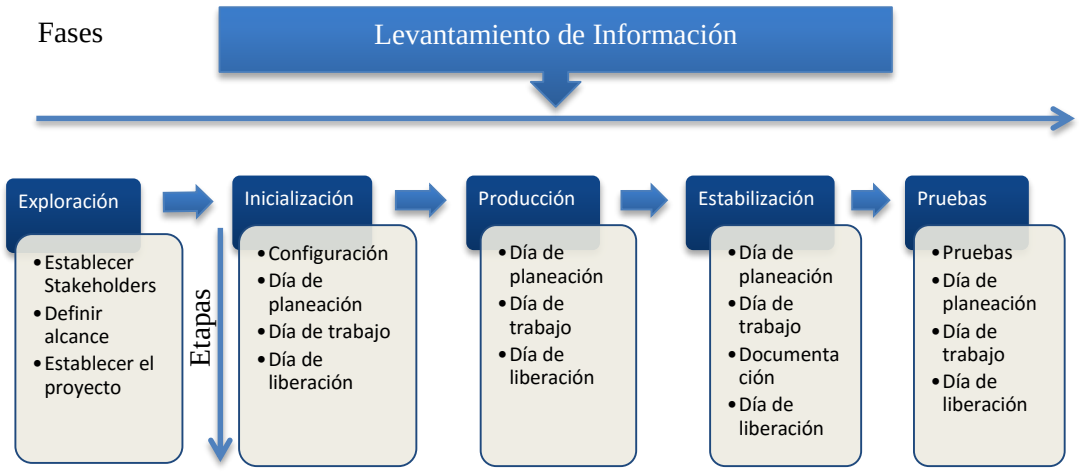

Figura 2: Fases y etapas de la metodología a usar

A continuación se detallan cada una de estas fases.

4.1 Fase I: Levantamiento de información

En ésta primera fase, se espera adquirir la información necesaria para orientar adecuadamente el proyecto y para lograr esto, será necesario hacer una revisión de la bibliografía relacionada con la solución informática que se busca implementar, para así proceder posteriormente a realizar las actividades sugeridas por la metodología Mobile-D.

Así que en esta fase, se han planeado las siguientes actividades

- Recolección de información sobre la regulación OBDII.
- Recolección de información sobre las interfaces Automóvil - Smartphone
- Recolección de información sobre protocolos de comunicación Smartphone – Web - Automóvil.
- Recolección de información sobre las diversas plataformas de desarrollo
- Generar documento del estado del arte del proyecto.

Dichas actividades a su vez están encaminadas a cumplir con los dos primeros objetivos específicos que se han propuesto para el desarrollo del presente proyecto

4.2 Fase II: Exploración

En ésta fase apoyándose en las recomendaciones de la metodología Mobile-D, se pretende llevar a cabo el proceso inicial de planificación, así como también, establecer los conceptos básicos del proyecto.

Las actividades a realizar en esta fase son:

- Establecer los clientes potenciales.
- Definir el grupo de Stakeholders.
- Realizar una definición inicial de requerimientos en forma de "historias de usuario".
- Hacer la planeación inicial del proyecto.
- Definir el entorno del proyecto.
- Definir la arquitectura a utilizar.

Es conveniente indicar que las funcionalidades a implementar se definirán teniendo en cuenta criterios como: la opinión de un experto en cuanto a la lógica del monitoreo, el sistema operativo base, la complejidad, el impacto y el alcance del proyecto.

4.3 Fase III: Inicialización

Ésta fase está pensada para posibilitar el éxito de las siguientes, por tal motivo, la meta será preparar el proyecto para evitar todos los posibles problemas que puedan surgir durante el desarrollo de la solución informática. Además, se prepararán todos los recursos físicos, tecnológicos y de comunicaciones para las actividades de producción.

Para llevar a buen término ésta fase, se proponen las siguientes actividades:

- Configurar el entorno de desarrollo del proyecto (IDE´s, dispositivos, periféricos, SVN[5]).
- Entrenarse sobre los conceptos y el uso de las herramientas que se utilizarán.
- Definir los medios de comunicación con los clientes y stakeholders.
- Crear el documento de requerimientos iniciales con base en las historias de usuario
- Construir el plan inicial del proyecto

Como resultado ésta fase la anterior (Exploración) se pretende haber cumplido el tercer objetivo específico definido en el proyecto.

4.4 Fase IV: Producción

En ésta fase se llevará a cabo toda la implementación de la solución informática usando un ciclo de desarrollo iterativo e incremental, en el cual se usará el desarrollo dirigido por pruebas (TDD) [17] y para esto, se tendrá como regla que antes de iniciar el desarrollo de una funcionalidad deberá existir una prueba que verifique su funcionamiento.

Para cada nueva iteración que se obtenga, se llevarán a cabo las siguientes actividades:

- Analizar los requerimientos de la iteración.
- Realizar la planeación de la iteración con base en las historias de usuario.

[5] Sistema de Control de versiones: http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html

- Definir, crear y revisar las pruebas de aceptación
- Realizar las tareas de desarrollo orientadas a pruebas (TDD)
- Realizar las pruebas de aceptación
- Integrar la iteración con la rama principal
- Generar retroalimentación con base en la iteración anterior.

4.5 Fase V: Estabilización

La metodología Mobile-D contempla ésta etapa para hacer una integración completa del sistema en los casos en que el desarrollo del proyecto involucra grupos trabajando en diferentes módulos o sub-sistemas de un mismo proyecto desde puntos geográficos diferentes.

El proyecto actual no sigue este comportamiento. Sin embargo, ésta fase se usará para hacer la integración y estabilización de los sub-sistemas que involucra la solución informática (componente en Cloud Computing y componente móvil).

Esto se hará debido a que es necesario contar con una etapa donde se pueda asegurar la calidad de integración y de implementación la solución generada.

Esta fase se llevará a cabo cada vez que se complete una nueva iteración en la fase anterior y para ello se llevarán a cabo las siguientes actividades

- Planear el proceso de integración
- Definir, crear y revisar las pruebas de aceptación de integración
- Generar documentación sobre la iteración.
- Llevar a cabo tareas de refactorización (de ser necesario)
- Integrar los sub-sistemas.
- Realizar las pruebas de integración
- Integrar la iteración con la rama principal

Generar retroalimentación con base en la iteración anterior.

4.6 Fase VI: Pruebas y preparación del sistema

Una vez terminado totalmente el desarrollo se pasará a la fase de pruebas, donde se iterará hasta llegar a una versión estable según lo establecido en los requerimientos definidos en las primeras fases.

Para ello, se llevarán a cabo las siguientes actividades:

- Detectar defectos de la solución implementada
- Analizar los defectos y crear o modificar las pruebas de aceptación
- Solucionar los defectos y validarlos contra las pruebas de aceptación
- Integrar la iteración con la rama principal

Generar retroalimentación con base en la iteración anterior

Al finalizar el proceso iterativo sobre los módulos a desarrollar se espera haber cumplido con los dos objetivos específicos restantes, los cuales están enfocados a desarrollo y validación del prototipo a crear.

5. Impacto esperado

Como fruto de éste trabajo se espera que el sistema desarrollado pueda ser extendido para ser aprovechado por usuarios de otros países. Así mismo se espera que el sistema generado en el proyecto pueda ser usado como base para generar sistemas enfocados en nichos específicos que requieran de un tratamiento diferente o adicional de la información, pero que usen el mismo esquema para acercarse a los conductores. Tal puede ser el caso de las entidades privadas y gubernamentales (aseguradoras, alquiladores de vehículos, ministerios de transporte y medio ambiente, entre otros), a las cuales les puede interesar hacer un control más estricto sobre los vehículos de un nicho de mercado particular, para así verificar el correcto trato y mantenimiento por parte de los conductores de dichos automotores

Si los entes de control de Estado hacen uso del sistema generado en éste proyecto, la comunidad en general podría verse beneficiada de éste proyecto, al contar con una herramienta de seguimiento y control vehicular efectiva que contribuiría a mantener un medio ambiente en condiciones más saludables.

II – MARCO TEÓRICO

Luego de describir el proyecto y la metodología a seguir, se procederá a hablar sobre las bases teóricas en las que se fundamenta la solución propuesta, para lo cual se presentan los conceptos más relevantes que serán utilizados a lo largo del mismo.

Así pues esta sección cubre lo descrito en la etapa de levantamiento de información (definida en la metodología) y está encaminada a cumplir los dos primeros objetivos específicos.

1. Sistemas OBD para Regulación de Control de Emisiones [1] [20]

La preocupación por regular el control de las emisiones causadas por los automóviles nació en 1966 en la cuenca de Los Ángeles – California, debido al crecimiento de la polución en la zona.

A raíz de ésta preocupación, se crearon diversos controles, los cuales fueron extendidos en 1968 a nivel nacional por parte del Gobierno Federal de los Estados Unidos. Posteriormente, en 1970 el Congreso Norteamericano aprobó la ley de aire limpio, al mismo tiempo que se creó la Agencia de Protección del Medio Ambiente (Environmental Protection Agency - EPA).

1.1 OBD-I [18]

Con estas acciones, se dio inicio a la publicación de una serie de normas y regulaciones para el control de las emisiones y mantenimiento de los vehículos. Así fue como en el año 1988, la Comisión de Recursos del Aire de California (California Air Resources Board - CARB) creó la regulación de los Sistemas de Diagnóstico de a Bordo (On Board Diagnostic - OBD), la cual estaba enfocada a los vehículos de modelo de fabricación mayor o igual a 1988 que fueran vendidos en California.

En principio cada fabricante creo su propia solución para cumplir con estas normas y debido a la diversidad de sistemas y señales, en 1998 la Sociedad de Ingenieros Automotrices (Society of Automotive Engineers - SAE) definió lo que debería ser el conector estándar y el conjunto de señales de prueba de diagnóstico, las cuales fueron adoptadas en su mayoría por la EPA.

Ésta regulación se conoce hoy en día como OBD-I y estaba enfocada a monitorizar ciertos parámetros del sistema como:

- La sonda lambda
- El sistema de regulación de gases de escape (Exhaust Gas Recirculation - EGR)
- Módulo de control electrónico (Electronic Control Module - EMC).

Para alertar al conductor sobre el mal funcionamiento del vehículo y de la necesidad de una revisión de los sistemas de control de emisiones, se definió el uso de una bombilla indicadora de mal funcionamiento (Malfunction Indicator Lamp - MIL), denominada Check Engine o Service Engine Soon. Así mismo, se requería indicar el código de falla (Diagnostic Trouble Code - DTC) para facilitar la identificación del sistema o componente defectuoso.

1.2 OBDII [18] [19] [20]

El primero de enero de 1996 entró en vigencia la segunda versión de ésta regulación (OBDII), la cual fue desarrollada por la SAE, debido al endurecimiento en los límites de emisiones permitidos por parte de la CARB y la EPA. Dicha regulación para los sistemas de diagnóstico contemplaba todos los vehículos de gasolina fabricados e importados por los Estados Unidos a partir de ese año y para los vehículos de pasajeros y camiones diésel a partir de 1997.

Hoy en día un sistema bajo la regulación OBDII está en capacidad de controlar prácticamente todos los sistemas de control de emisiones y los componentes que puedan afectar los gases de escape producidos por el vehículo.

Similar a la regulación OBD-I, la regulación OBDII dictamina que se debe guardar el código de la falla (DTC) cuando se detecte que un componente está operando fuera de las condiciones definidas por el fabricante o cuando se supere el umbral máximo de emisiones permitido. También se debe contar con una lámpara MIL que deberá encenderse para avisar al conductor de la falla y ésta no deberá apagarse hasta que se realicen las correspondientes reparaciones o hasta que hayan pasado 3 ciclos de conducción consecutivos, sin que el problema reincida.

Los códigos de falla DTC generados deben ser almacenados en la Memoria de Almacenamiento Activa (KAM - Keep Alive Memory) para que puedan ser consultados posteriormente por un experto que esté en capacidad de detectar y corregir el origen del fallo. Según la regulación, la información residente en la KAM se borrará después de 40 arranques en frío.

2. Monitores de emisiones OBDII

Una componente esencial de los sistemas OBDII presentes en los vehículos, son los Monitores de Emisiones, los cuales se encargan de determinar si todos los componentes de emisiones, han sido evaluados por el sistema OBDII. Estos monitores realizan pruebas periódicas a ciertos sistemas y componentes, para garantizar que se están operando dentro de los límites permitidos.

La EPA ha definido 11 Monitores de Emisiones (o Monitores I/M). Sin embargo, es importante mencionar que no todos los monitores están actualmente soportados por todos los vehículos. La cantidad de monitores presentes en cada vehículo varía según la estrategia de control de emisiones del fabricante.

Estos monitores se pueden dividir en 2 categorías. Por un lado se tienen los monitores que realizan comprobaciones continuamente y por el otro, se tienen los monitores que realizan las comprobaciones solo bajo condiciones específicas de operación del vehículo.

2.1 Monitores Continuos

Los siguientes componentes continuamente monitorizados están siempre listos:

- Fallos del Encendido
- Sistemas del Combustible

- Monitor General de Componentes (CCM)

Cuando el vehículo es puesto en marcha, el sistema OBDII comienza a monitorizar continuamente los sensores claves del motor, del sistema de encendido y la demanda de combustible.

2.2 Monitores no Continuos

Estos se diferencian de los monitores continuos, dado que existen emisiones y componentes del sistema del motor, que requieren unas condiciones específicas de funcionamiento del vehículo antes de censar, tales como conducción por carretera, paradas y arrancadas, conducción por ciudad y conducción nocturna, entre otras. Estos monitores se enumeran a continuación:

- Sistema EGR
- Sensores O2
- Catalizador
- Sistema Evaporativo
- Calentador Sensor O2
- Aire Secundario
- Catalizador calentamiento
- Sistema A/C
- Estado Monitores de Emisiones OBDII

3. Modos de Prueba

La regulación OBDII contempla varios modos de prueba, los cuales han sido creados pensando en que sean comunes para todos los vehículos, independientemente del fabricante. Estos modos de prueba se usan para acceder a diferente información del sistema de diagnóstico.

Los modos de trabajo más extendidos son los siguientes:

- **$01**: Utilizado para conocer la información del módulo electrónico (ECU) que se encuentra a disposición de la herramienta de escaneo.
- **$02**: Muestra los datos de cuadro congelado (“Freeze Frame Data”) que ha ido almacenado la ECU.
- **$03**: Muestra la lista de los códigos de fallos asociados a emisiones que han sido guardados.
- **$04**: Se utiliza para limpiar la información de códigos de diagnóstico asociados a emisiones que han sido guardados, así como también los datos de cuadro congelado.
- **$05**: Muestra los valores tomados a los sensores de oxígeno y los resultados de los test que les ha realizado.
- **$06**: Se usa para hacer una petición al sistema de monitoreo y así obtener los resultados de los test realizados por la ECU al sistema de monitoreo continuo y no continuo. Existe normalmente un valor mínimo, máximo y actual para cada monitoreo no continuo.

- **$07**: Se usa para solicitar los códigos de problema de diagnóstico detectados durante el último ciclo de conducción. Este modo lo suelen utilizar los técnicos después de una reparación del vehículo, y después de borrar la información de diagnóstico para ver los resultados de las pruebas luego de un ciclo de conducción, determinando si la reparación ha solucionado el problema.
- **$08**: Permite hacer pruebas externas para controlar el funcionamiento del sistema de diagnóstico interno del vehículo.
- **$09**: Usada para obtener información del vehículo, tal como el número de identificación del vehículo, la identificación del software instalado en la ECU. Entre otros.

Adicional a los modos mencionados, los fabricantes suelen incluir más funciones para poder controlar y gestionar muchos más aspectos del vehículo.

4. Códigos de Falla (Diagnostic Trouble Code - DTC)

Los códigos de falla están regulados por la norma SAE J1979 y este debe ser seguido por los fabricantes de vehículos. Los códigos de falla constan de 5 caracteres, donde el primer carácter es una letra y los 4 restantes son números.

A continuación se hace una descripción de cada carácter, sus posibles valores y su significado.

CAR	DESCRIPCIÓN	POSIBLES VALORES
1	Indica la función del vehículo	P - Tren motriz o motor y trasmisión (Powertrain)
		B - Carrocería (Body)
		C - Chasis (Chassis)
		U - No definido (Undefined)
2	Indica si el código es genérico o específico	0 - Genérico para todas las marcas y definido por SAE.
		1 - Específico y definido por el fabricante del vehículo. El código generalmente es diferente para cada fabricante.
3	Indica el subsistema del vehículo	0 - El sistema electrónico completo
		1 y 2 - Control combustión
		3 - Sistema de encendido
		4 - Control de emisión auxiliar
		5 - Control de velocidad y ralentí
		6 - ECU y entradas y salidas
		7 - Transmisión
4 y 5	Indican la falla	

Tabla 1: Descripción de caracteres DTC, sus posibles valores y significado

En la siguiente figura se observa la función que cumple cada carácter que compone el código de una falla.

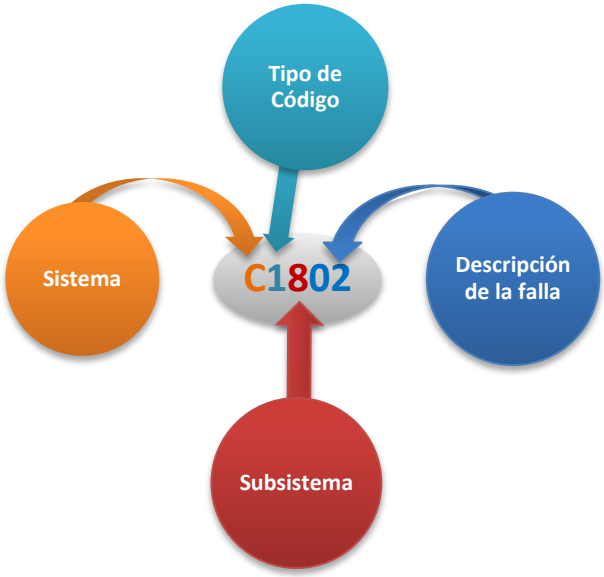

Figura 3: Función de cada carácter de un código de falla

5. Conector de Diagnóstico

La norma ISO 15031-3:2004 reglamenta las especificaciones del conector que debe ser usado para acceder al sistema OBDII. Dicho conector debe ser hembra (J1962) de 16 pines (2*8), aunque no todos los pines son utilizados y según la norma, debe estar ubicado en el compartimiento de pasajeros, cerca al puesto del conductor

En la siguiente figura se puede observar la distribución de los pines en el conector.

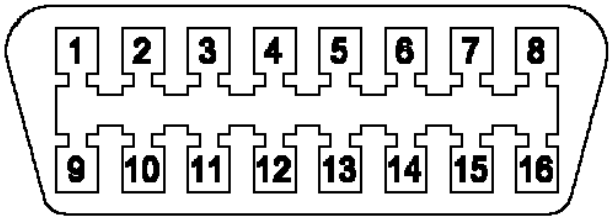

Figura 4: Pines del Conector OBDII[6]

[6]Imagen tomada de: http://www.furgovw.org/index.php?topic=231559.0

En la siguiente tabla se describe el uso de cada pin presente en el conector

PIN	USO
1	Reservado para uso específico del fabricante del vehículo
2	J1850 Bus+
3	Reservado para uso específico del fabricante del vehículo
4	Masa de Chasis
5	Masa de señal
6	CAN High (J-2284)
7	ISO 9141-2 Línea K e ISO/DIS 14230-4
8	Reservado para uso específico del fabricante del vehículo
9	Reservado para uso específico del fabricante del vehículo
10	J1850 Bus
11	Reservado para uso específico del fabricante del vehículo
12	Reservado para uso específico del fabricante del vehículo
13	Reservado para uso específico del fabricante del vehículo
14	CAN Low (J-2284)
15	ISO 9141-2 línea L e ISO/DIS 14230-4
16	Alimentación de batería

Tabla 2: Pines del conector de diagnóstico OBDII

6. Protocolos de señal

Actualmente la regulación OBDII permite 5 protocolos de señal para uso en los conectores de diagnóstico. La mayoría de vehículos implementa solo 1 de los protocolos. La forma más fácil de conocer el protocolo usado en un vehículo es revisando los pines presentes en el conector J1962, aunque por otro lado, los fabricantes han escogido que protocolo utilizar y todos los vehículos que salen de su fábrica salen con el mismo protocolo, por tanto ésta es otra opción para identificar el tipo de protocolo usado en el conector OBDII.

La siguiente es la lista de protocolos que la regulación OBDII soporta en los conectores J1962:

1. **SAE J1850 PWM**: Protocolo de Modulación de Ancho de Pulso (Pulse-Width Modulation)
   - pin 2: Bus+
   - pin 10: Bus–
   - High voltage is +5 V
   - Longitud del mensaje restringida a 12 bytes, incluyendo el control de redundancia cíclica (CRC - Cyclic Redundancy Check)
   - Usado en vehículos de Ford USA

2. **SAE J1850**: VPW Protocolo de Modulación de Ancho de Pulso Variable (Variable Pulse Width)
   - pin 2: Bus+
   - Bus idles low
   - High voltage is +7 V
   - Decision point is +3.5 V
   - Longitud del mensaje restringida a 12 bytes, incluyendo CRC
   - Utiliza CSMA/NDA
   - Usado en vehículos de GM USA

3. **ISO 9141-2**: Éste protocolo tiene una tasa de datos seriales asíncrona de 10.4 kBaud. Es similar al protocolo RS-232; sin embargo, los niveles de señal son diferentes.
   - pin 7: K-line
   - pin 15: L-line (optional)
   - UART signaling
   - Longitud del mensaje máxima de 260Bytes. El campo de datos puede ser máximo de 255Bytes
   - Usado en vehículos Europeos, Asiáticos y Chrysler

   .
4. **ISO 14230 KWP2000**: (Keyword Protocol 2000)
   - pin 7: K-line
   - pin 15: L-line (optional)
   - Capa física idéntica a la norma ISO 9141-2
   - Tasa de transferencia de 1.2 a 10.4kBaud
   - El mensaje puede tener hasta 255Bytes en el campo de datos
   - Utilizado por el grupo VAG

5. **ISO 15765 CAN**: Protocolo creado por Bosch y usado en entornos diferentes al de la industria automotriz. Sin embargo, todos los vehículos vendidos en estados Unidos a partir de 2008 deben implementar el protocolo CAN como uno de sus protocolos de señal.
   - pin 6: CAN High
   - pin 14: CAN Low

7. Interfaces de Hardware OBDII

Una vez conocido el conector y los protocolos de comunicación, se necesita del hardware adecuado para poder acceder a los datos de la unidad de diagnóstico de cada vehículo. Actualmente existen dispositivos que brindan soporte a los protocolos definidos por la regulación OBDII. Sin embargo, no todos dan soporte a la totalidad de protocolos. Es por ello que entre los más

usados se encuentra el chip creado por ELM Electronics[7] conocido como ELM327 [21], el cual según el fabricante actualmente da soporte a todos los protocolos definidos por la regulación OBDII. Gracias a esto, cuando un sistema externo quiera establecer comunicación con el sistema OBDII no se tendrá que preocupar por el protocolo de comunicación a usar, ya que el chip ELM327 se encargará de hacer dicho proceso y realizar la comunicación adecuadamente.

Adicionalmente, éste chip es ofrecido con diferentes interfaces de comunicación que permiten acceder a los datos capturados por el sistema OBDII de diversas maneras. En la siguiente tabla se enumeran las diferentes interfaces (RS232, USB, Bluetooth y WIFI) disponibles y su posibilidades de acceso desde equipos de escritorio y terminales móviles iOS, Android y Windows Phone

Tipo de Interfaz	PC	Android	iOS	Windows Phone
ELM327 RS232	✓			
ELM327 USB	✓			
ELM327 Bluetooth	✓	✓		✓
ELM327 WiFi	✓	✓Con restricciones	✓	

Tabla 3: Tipo de interfaz y compatibilidad- Modificada de la original [22]

Como se puede observar, según el fabricante, la interfaz WIFI solo puede ser usada por computadoras o dispositivos iOS. Sin embargo, luego de indagar un poco más, se pudo determinar que la interfaz WIFI se puede usar con dispositivos Android que permitan definir una red WIFI de forma manual, lo cual es posible, si el dispositivo Android tiene permisos de súper usuario (Ser usuario root puede anular la garantía del dispositivo móvil). Así mismo según [23], ya existen aplicaciones que usan la interfaz ELM327 Bluetooth.

Pero independiente del tipo de interfaz de comunicación, el chip ELM327 ofrece las siguientes funciones:

- Lee códigos de error tanto genéricos como de fabricante y muestra su interpretación.
- Más de 3000 definiciones de códigos en la base de datos.
- Muestra datos en vivo de sensores como:
  - RPM o Valor de carga.
  - Temperatura de refrigerante.
  - Estado del sistema de combustible.
  - Velocidad o Ajustes de combustible a corto y largo plazo.
  - Presión del colector
  - Tiempo de avance
  - Temperatura del aire

[7] http://elmelectronics.com/obdic.html

- Caudal de aire
- Posición del acelerador.
- Sensores de oxígeno y voltajes
- Presión de combustible.

En cuanto a las especificaciones técnicas del ELM327 se tienen:

- Protocolos OBD2 soportados:
  - ISO15765-4 (CAN);
  - ISO14230-4 (KWP2000);
  - ISO9141-2;
  - J1850 VPW;
  - J1850 PWM
- Velocidad de transmisión: 9600 ó 38.400
- Indicadores: LED: OBD, Tx/Rx, RS232 Tx/Rx, Power Voltaje: 12V.
- Con protección interna para corto circuitos y sobre voltaje.
- Soporte para todos los vehículos OBD2 certificados.

En la siguiente figura se puede observar una imagen de un ELM327 Bluetooth (Izquierda) y un ELM327 WIFI (Derecha).

Figura 5: Interfaces de Hardaware ELM327 Bluetooth y WIFI

8. Plataformas Móviles

Luego de tener claras las bases teóricas sobre las cuales se fundamenta el proyecto, se pasará a realizar el estudio de las principales plataformas móviles para saber hacia cuál de ellas enfocar el desarrollo del proyecto.

Dado que se planteó la creación de una Aplicación Móvil, se han realizados diversas consultas estadísticas en el sitio Web StatCounter[8] sobre los Sistemas Operativos orientados a plataformas móviles (omitiendo Tablets) y su penetración en el mercado a nivel nacional y mundial.

[8] http://gs.statcounter.com/

Estas estadísticas contemplan el lapso comprendido entre enero de 2010 y Noviembre de 2013 y los resultados generados se pueden ver en las figuras mostradas a continuación.

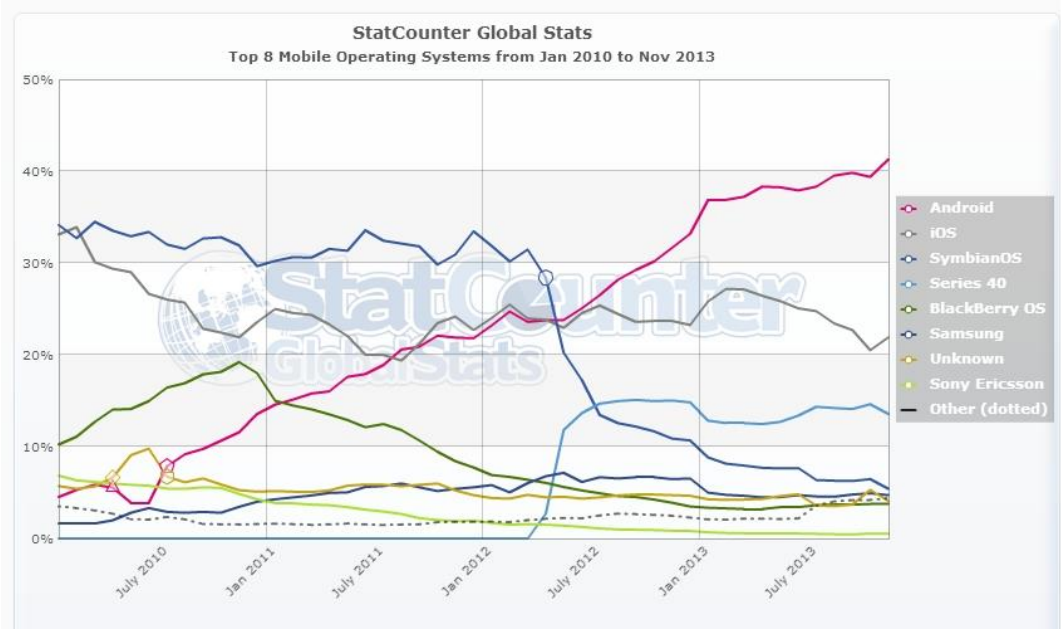

Figura 6: Sistemas Operativos para plataformas móviles – Mundial

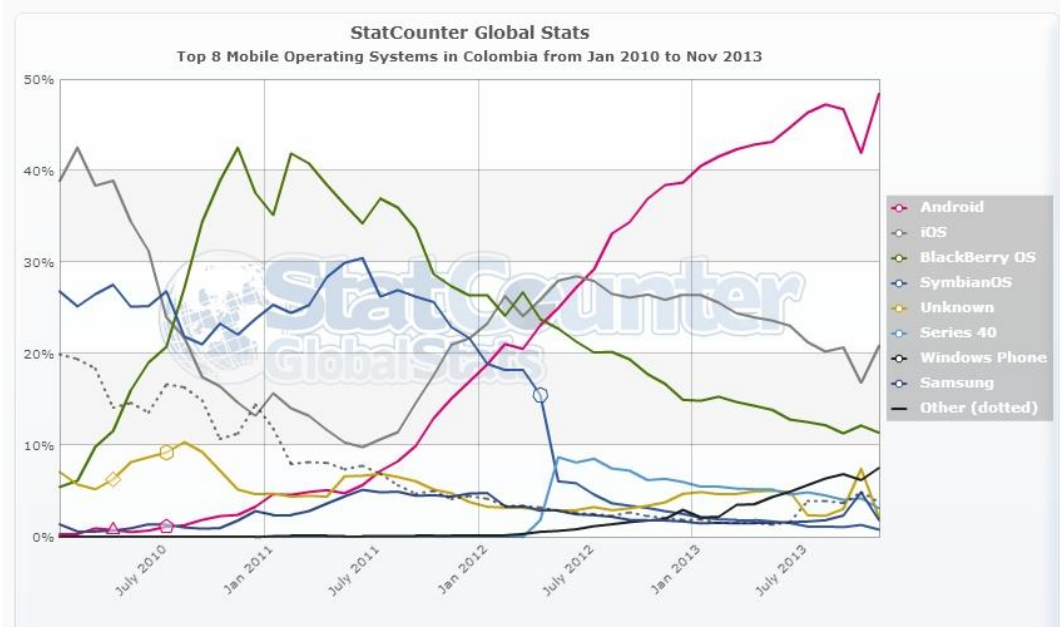

Figura 7: Sistemas Operativos para plataformas móviles – Colombia

Al observar las gráficas anteriores, se evidencia que a Noviembre de 2013 los Sistemas Operativos IOS y Android acaparan cerca del 60% del mercado a nivel mundial y el 70% aproximadamente a nivel nacional.

Así pues, se procederá a realizar un breve análisis de estos dos Sistemas Operativos y su arquitectura, comenzando por Android.

8.1 Android

Android fue inicialmente concebido y desarrollado por la compañía Android Inc, la cual fue fundada en 2003. La compañía recibió el respaldo económico de Google Inc para el desarrollo del nuevo Sistema Operativo. Android Inc. fue adquirida por Google Inc. en julio de 2005.

El Sistema operativo tomó como base el núcleo de Linux (Versión 2.6) y fue lanzado oficialmente en su primera versión (Beta) el día 5 de Noviembre de 2007 junto con la Open Handset Alliance [24], un consorcio de más de 76 compañías de hardware, software y telecomunicaciones (entre las cuales se encuentran Google, HTC, Intel y LG), con la intención de avanzar en los estándares de los sistemas abiertos.

Actualmente se han liberado 11 versiones (principales) del sistema operativo, siendo la versión 4.4 la más reciente. Cada versión se distingue por llevar el nombre de un postre que comienza por una letra del alfabeto que van desde la A (Versión 1.0) hasta la K (Versión actual: 4.4). En el Anexo 1 (Versiones Android OS), se enumeran las diferentes versiones existentes

Arquitectura

La arquitectura de Android está dividida en capas que facilitan la creación de aplicaciones. Dicha distribución permite acceder a las capas más bajas mediante el uso de librerías para de ésta forma evitar la programación a bajo nivel. Una de las características más importantes de la arquitectura Android es que todas las capas están basadas en software libre.

Cada capa usa elementos de la capa inferior para llevar a cabo sus funciones. A este tipo de arquitectura se le conoce como pila. La siguiente Figura muestra la arquitectura de Android (Realizado a partir del diagrama publicado en el sitio oficial de Android developers[9]). En el Anexo 2 (Arquitectura de Android OS) se puede encontrar la descripción detallada de la arquitectura.

[9] http://developer.android.com/images/system-architecture.jpg

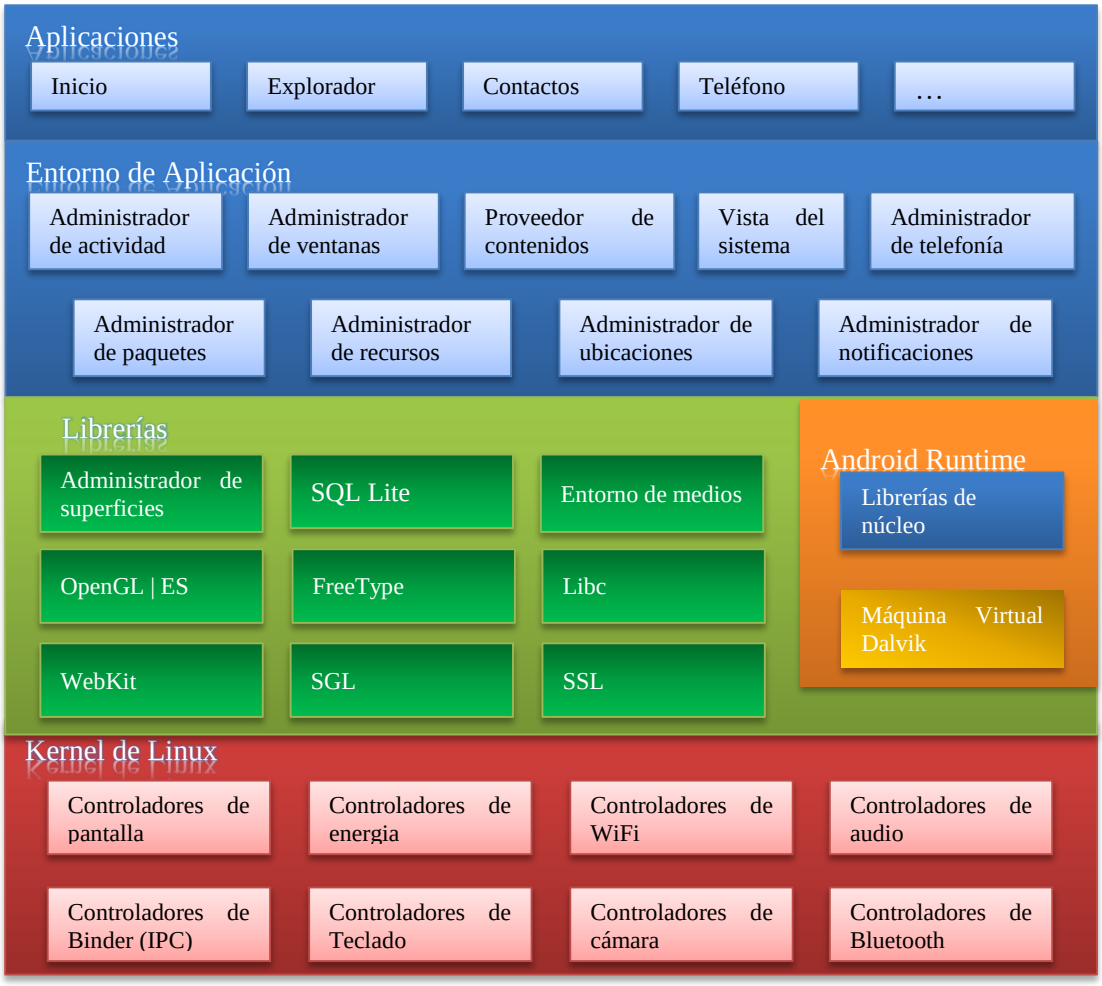

Figura 8: Arquitectura Android

8.2 IOS [26]

El Sistema Operativo iOS fue creado por Apple Inc. tomando como base el Sistema operativo Mac OS X y usando el mismo núcleo (Mach/FreeBSD)

La primera versión del S.O (Sin nombre definido) fue dada a conocer al público el 29 de Junio de 2007. Dicho S.O estaba enfocado a dispositivos iPhone, pero posteriormente fue usado en dispositivos como el iPod Touch, iPad y el Apple TV.

Inicialmente el Sistema Operativo no permitía la instalación de aplicaciones de terceros, pero eso cambió con el lanzamiento del Kit de desarrollo de Software (SDK), dado a conocer el 6 de marzo de 2008. Con dicho lanzamiento, Apple Inc. le dio el nombre de iPhone OS al Sistema Operativo y más adelante, en junio de 2010 lo cambió por iOS

Algo importante con respecto a este Sistema operativo radica en que a diferencia de ANDROID OS, el cual puede ser instalado en diversos dispositivos de múltiples fabricantes, Apple Inc. no permite la instalación de iOS en hardware de terceros.

A la fecha existen 7 versiones principales de este sistema operativo, la cuales se detallan en detalle en el Anexo 3 (Versiones de IOS)

2.2.1 Arquitectura

Similar a la arquitectura de ANDROID OS, iOS cuenta con una arquitectura dividida en cuatro capas de abstracción, las cuales se observan en la siguiente figura (Tomada de [26]) y se describen en detalle en el Anexo 4 (Arquitectura de IOS).

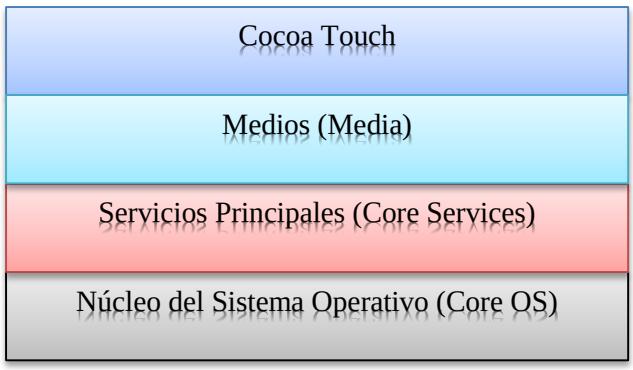

Figura 9: Arquitectura iOS.

9. Ambientes de desarrollo

Luego de revisar los Sistemas Operativos posibles, se procederá a analizar las opciones asociadas a entornos de desarrollo manejados por cada uno.

9.1 Android SDK [27]

El conjunto de Herramientas para el Desarrollo de aplicaciones sobre Android OS se conoce como Android SDK (Standard Development Kit, por sus siglas en ingles). Este SDK se caracteriza por estar compuesto de diversos paquetes, los cuales pueden ser descargados separadamente desde la página Web para desarrollo sobre Android [27]. Sin embargo, para llevar a cabo dicha labor también se puede hacer uso de una herramienta proporcionada por Android conocida como SDK Manager.

Los paquetes disponibles que contiene el SDK según [28] son:

- **SDK Tools**: Contiene las herramientas básicas para desarrollar, probar y hacer depuración de aplicaciones Android, incluyendo un emulador del Sistema Operativo.
- **SDK Platform-tools**: Contiene herramientas dependientes de la plataforma necesarias para desarrollar y depurar aplicaciones Android.

- **Documentación**: Copia de la documentación del SDK
- **SDK Platform**: Existe una plataforma SDK para cada versión de Android. Siempre que se construya una aplicación para Android, se debe especificar una plataforma SDK.
- **Imágenes del sistema**: Son usadas por el emulador de Android. Para cada plataforma SDK existe una o más imágenes.
- **Fuentes del Android SDK**: Código fuente del SDK útil si se desea realizar depuración de código
- **Ejemplos de uso del SDK**: Colección de ejemplos que demuestran el uso del SDK
- **APIs de Google**: Complemento e imagen para programar usando dicho API.
- Soporte Android: Biblioteca estática ofrecida con el fin de utilizar APIs potentes que no están disponibles en la plataforma estándar
- **Google Play Billing**: Bibliotecas estáticas y ejemplos que facilitan la integración de servicios de facturación con Google Play[10].
- **Licencia Google Play**: Bibliotecas estáticas y ejemplos que permiten llevar a cabo la verificación de licencia una aplicación cuando se distribuye por medio de Google Play.

Adicional al SDK, Android ofrece un par de opciones para facilitar la creación de aplicaciones, las cuales son:

- **Plugin ADT (Android Development Tools)**: Android ha creado un plugin para ser usado en Eclipse IDE[11]. Dicho plugin extiende facilita en gran medida el desarrollo de aplicaciones.
- **Android Studio**: Nuevo IDE basado en IntelliJ IDEA[12] que ofrece herramientas para desarrolladores Android integradas para el desarrollo de aplicaciones, sustituyendo así a la plataforma Eclipse. Aunque no ha sido liberado en versión estable, ya es posible su uso para fines de prueba.

Por otro lado, Android también ofrece la opción de descargar un Paquete de Herramientas de Desarrollo (ADT Bundle), con todo lo necesario para empezar a desarrolla aplicaciones. Este paquete incluye:

- Eclipse IDE + plugin ADT
- Paquete: Android SDK Tools
- Paquete Android Platform-tools
- La última plataforma Android disponible

[10] Tienda virtual de aplicaciones para dispositivos Android: https://play.google.com/store

[11] Página Web de Eclipse IDE: http://www.eclipse.org/

[12] Página Web de IntelliJ IDEA: http://www.jetbrains.com/idea/

- La última imagen del Sistema Android disponible para ser usada por el emulador.

9.2 IOS SDK [29]

Apple ofrece un conjunto de herramientas estándar (SDK) que se pueden utilizar para crear aplicaciones orientadas a dispositivos iOS. Para programar dichas aplicaciones se debe usar Objective-C, el cual es el lenguaje de programación soportado por iOS y Mac OS.

Objective-C fue creado en el año 1980 y se trata de una extensión del lenguaje C. Añade muchas características adicionales a C y lo que es más importante, una estructura POO. Objective-C utiliza principalmente para el desarrollo de aplicaciones de Mac OS X y iOS, lo cual sedujo a un entregado grupo de seguidores que aprecian sus capacidades y sintaxis.

El SDK puede ser descargado desde su página Web, aunque para ello, se requiere estar registrado como "Mac Developer". Sin embargo, el SDK también puede obtenerse gratuitamente si se descarga desde la tienda virtual de Apple (App Store). Sin embargo, el SDK tiene una restricción importante: sólo funciona en equipos Macintosh que usen procesadores Intel

Por otro lado, similar al SDK de Android. Apple ofrece el SDK empaquetado con un conjunto de herramientas más amplio conocido como Xcode. Este programa es el entorno de desarrollo integrado (IDE) de Apple y trae consigo los componentes necesarios para crear aplicaciones enfocadas a dispositivos iOS.

La siguiente es la lista de herramientas que trae dicho paquete:

- Xcode IDE: Programa principal usado para crear aplicaciones. Está basando en Project Builder, el cual es un IDE creado por la empresa NeXT computer, Inc. (Empresa fusionada con Apple Inc. en 1996) Incluye la colección de compiladores del proyecto GNU (GCC), y puede compilar código en diferentes lenguajes tales como:
  - C
  - C++
  - Objective-C
  - Objective-C++
  - Java
  - AppleScrip
- iOS SDK: Kit de desarrollo de Apple que permite acceder a los elementos "Cocoa Touch". Ofrece las siguientes herramientas:
  - iCloud Storage: Para guardar datos en la "nube".
  - Notification Center: Permite gestionar las notificaciones al usuario.
  - Newstand: Para publicar en este quiosco virtual.
  - Automatic Reference Counting: gestionar memoria de aplicación.
  - Twitter Integration: Tuitea directamente desde la aplicación.
  - Storyboards: Diseñar el "workflow" de la aplicación a implementar.
  - AirPlay: Permite enviar en "streaming" video y audio al Apple TV.

- iMessage

- Código de ejemplo: Ejemplos prácticos para usar como punto de partida
- Simulador iOS: Emulador de dispositivos iOS llamado Aspen, usado para probar la funcionalidad de las aplicaciones desarrolladas sin necesidad de contar con el dispositivo físico.
- Depurador de código: Útil para hacer seguimiento y control de errores sobre las aplicaciones desarrolladas

9.3 Desarrollo Multiplataforma

En la sección anterior se habló sobre iOS y Android, así como de los entornos de desarrollo de cada S.O. Dichos entornos permiten construir aplicaciones orientadas únicamente a su plataforma específica, lo cual presenta inconvenientes al crear aplicaciones multiplataforma, dado que se debe realizar un desarrollo diferente para cada plataforma.

Como alternativa a estos entornos de desarrollo, se pueden usar herramientas no ligadas a una plataforma puntual, que permiten en un menor tiempo crear soluciones multiplataforma.

Con esto en mente, se hablará de dos de las alternativas existentes que permiten generar dicho tipo de contenido.

Aplicaciones Web

La primer opción al crear contenido multiplataforma es por medio de aplicaciones Web, dado que estas usan tecnologías estándar como: HTML5, JavaScript y CCS3, las cuales son soportadas por los navegadores Web de los Smartphone actuales.

Sin embargo, dichas aplicaciones presentan algunas desventajas, tales como:

- Dependencia de una aplicación nativa (Navegador Web del dispositivo) para funcionar.
- Restricciones para interactuar con el hardware de los dispositivos (acelerómetro, giroscopio, Bluetooth, etc)
- Falta de optimización de las aplicaciones creadas, ya que algunos componentes Web no son realizados teniendo en mente los dispositivos móviles, por lo tanto su experiencia de usuario no es la mejor y suelen funcionar de forma más lenta que una aplicación nativa.
- Las aplicaciones Web no pueden ser instaladas nativamente en los dispositivos móviles
- Se requiere de conexión a Internet si se quieren usar, dado que por lo general están alojadas en servidores Web

Para crear este tipo de aplicaciones, existen Frameworks que permiten desarrollar interfaces que lucen similares al de las aplicaciones nativas pero usan estas tecnologías Web (HTML5, JavaScript y CSS3). A continuación se muestran algunos de estos Frameworks, junto con una breve descripción de cada uno:

JQuery Mobile [30]

JQuery Mobile es uno de los Frameworks más populares basado en HTML5 y diseñado para crear Sitios Web responsivos y optimizado para todas las plataformas (dispositivos móviles, tabletas y computadores de escritorio). Está desarrollado a partir de JQuery y JQuery UI. Su implementación es sencilla y posee amplia documentación, además cuenta con una herramienta para la creación la temas personalizados (ThemeRoller for jQuery Mobile [31]) aprovechando las ventajas de CSS3.

El Framework se distribuye con licencia MIT [32], lo cual implica que no tiene prácticamente restricciones y puede ser usado sin ningún pago adicional cuando se destine a la creación de aplicaciones comerciales.

Sencha Touch [33]

Framework creado a partir de la plataforma HTML5 de Sencha para enfocarse al desarrollo de aplicaciones móviles de alto rendimiento escritas en HTML5. Este Framework incluye soporte para diversos sistemas operativos, tales como: Blackberry 10, iOS, Android, BlackBerry, Windows Phone, Tizen, entre otros.

El Framework se distribuye con 3 tipos de licencia dependiendo del enfoque [34].

- **Licencia Open Source:** Licencia pensada para cuando se desea crear una aplicación que será distribuida con el código fuente, bajo una licencia compatible con GNU GPL v3 [35]
- **Licencia de Software Comercial (libre)**: Pensada para cuando se desean crear aplicaciones propietarias en las cuales no se va a distribuir el código fuente. Esta licencia no tiene restricción en el número de aplicaciones que se pueden crear.
- **Licencia OEM comercial:** Licencia adecuada cuando se desea incluir Sencha Touch como parte de un Entorno de desarrollo Integrado (SDK) de uso comercial.

PhoneJS [36]

Framework basado en HTML5 y JavaScript que contiene lo necesario para crear aplicaciones compatibles con los más populares Smartphone y Tabletas. Soporta Sistemas Operativos IOS, Android y Windows Phone.

Según la Página oficial de Phone JS, el Framework ofrece una experiencia de usuario optimizada para ser usada con dispositivos táctiles, al disponer de características tales como:

- Creación componentes de forma nativa
- Navegación entre páginas de forma sencilla
- Manejo de vistas y acceso a la capa de datos
- Enfoque Single Page Application (SPA, Aplicación de una sola página) [37].

El Framework se distribuye bajo 2 licencias [38]:

- Libre para uso no comercial
- De pago por suscripción anual enfocada a uso comercial o gubernamental

Kendo UI [39]

Framework para el desarrollo de sitios web y aplicaciones móviles usando HTML5. Posee componentes basados en JQuery, una simple y consistente interfaz de programación, un Framework MV-VM [40], temas y plantillas. Permite construir aplicaciones que parecen y se usan igual que una aplicación nativa por medio de un simple código de interfaz de usuario usando solo HTML5 y JavaScript.

El Framework se distribuye bajo 2 licencias [41]:

- Licencia de código abierto: Puede ser usada en aplicaciones comerciales o cuando se desea distribuir el código fuente. Sin embargo, esta licencia aplica a la versión reducida del Framework [42] la cual cuenta con menos componentes y características que la versión paga (aproximadamente 40 componentes)
- Licencia de pago con acceso a los más de 70 componentes que posee el Framework, actualización del código fuente por un año y 48 horas de soporte dedicado

Zepto js

Zepto.js es una librería JavaScript minimalista pensada para ser usada exclusivamente en navegadores modernos. El Framework usa una sintaxis compatible con jQuery y tiene como principal objetivo el brindar una opción ligera (menos de 10k).

Al igual que jQuery Mobile, el Framework se distribuye con licencia MIT.

AngularJS [43]

Framework en JavaScript desarrollado por ingenieros de Google, creado con el enfoque SPA y basado en el patrón Modelo Vista Controlador (MVC [44]). Se caracteriza por extender el tradicional HTML con etiquetas propias conocidas como directivas, lo cual permite usar tags personalizados. Entre sus características se encuentran:

- Control de enlace de datos entre la capa de vista y la capa controladora, también conocido como databinding,
- Soporte para consultas ajax con peticiones HTTP,
- Sistema de plantillas,
- Manipulación de datos en JSON,
- Inyección de dependencias,
- Formularios de validación,
- Desacoplamiento del Modelo de Objetos del Documento (DOM [45]) de Javascript,
- Internacionalización i18n y l10n,
- Filtros,
- Soporte para crear unidades de pruebas

Similar a jQuery, Angular también ofrece un Framework enfocado exclusivamente a ambientes móviles conocido como Ionic [46], el cual ofrece un conjunto de componentes creados en HTML, JavaScript y CSS3. Su característica principal radica en la simplicidad y velocidad de sus componentes al reducir la manipulación del DOM.

Este Framework también cuenta con licencia MIT

Aplicaciones hibridas

La segunda opción al hablar de desarrollo multiplataforma es el conocido como "Aplicaciones Hibridas", las cuales son creadas usando un lenguaje de programación genérico que puede ser interpretado y compilado para crear soluciones que pueden ser instaladas nativamente en diversos Sistemas Operativos sin realizar mayores ajustes.

En este punto se puede hablar de dos enfoques al momento de crear dichas aplicaciones. A continuación se hablará de cada enfoque.

Aplicaciones Hibridas Basadas en Aplicaciones Web

Si bien los Frameworks mencionados en la sección anterior ofrecen alternativas para crear soluciones multiplataforma, éstas por si solas no pueden ser distribuidas, instaladas y ejecutadas de forma nativa,

Para solventar ésta limitación, la Fundación de Software de Apache, creó la plataforma de código abierto Apache Cordova [47] o Apache Phonegap

La plataforma está orientada a facilitar que las soluciones Web enfocadas a dispositivos móviles, usando HTML5, JavaScript y CSS3, puedan ser enmascaradas en código nativo para que el paquete resultante pudiera ser entregado, como cualquier otra aplicación, a través de las tiendas de aplicaciones de cada Sistema Operativo.

Adicionalmente, la plataforma presenta las siguientes ventajas:

- ✓ Disponible para iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, y Symbian.
- ✓ Se Integra fácilmente con los Frameworks para desarrollo de páginas Web móviles nombrados en la sección anterior, entre otros.
- ✓ Tiene una cantidad creciente de plugins [48] (varios creados por la comunidad de código abierto), los cuales están escritos en código nativo de cada S.O. y están pensados para ofrecer al desarrollador acceso por medio de JavaScript a ciertas funciones nativas propias de cada dispositivo, tales como: cámara, acelerómetro, Bluetooth, información del dispositivo, lista de contactos, entre otros. Sin embargo, dado que la funcionalidad de cada plugin está escrita en código nativo (Objective C, Java, C#), no todos los plugins disponibles brindan soporte a todos los S.O. contemplados por la plataforma.

Aplicaciones Hibridas Interpretadas

Existe un segundo enfoque dentro de las aplicaciones híbridas, conocido como aplicaciones móviles interpretadas.

La idea central de estas aplicaciones consiste en programar la solución usando un único lenguaje de programación, el cual será transformado y compilado antes de ser empaquetado en la aplicación nativa.

En este grupo se pueden encontrar varias plataformas tales como Xamarin [49] o Appcelerator Titanium [50].

En el este caso de Xamarin la codificación se hace por medio de C#, mientras que con Appcelerator Titanium, el código se escribe en JavaScript.

Ambas plataformas proporcionan librerías que sirven para acceder a los controles nativos del móvil, consiguiendo de esta forma un apartado gráfico y comportamiento bastante similar a las aplicaciones nativas. Sin embargo, con estas opciones no se llega a tantas plataformas como con Apache Cordova/Phonegap.

10. Proveedores de Servicios en la Nube

Según lo planteado, el proyecto a desarrollar contempla funcionalidad alojada en la nube. Es por esto que se debe elegir un proveedor de servicios en la nube que permita como mínimo hospedar los siguientes componentes:

- Repositorio de información: Base de datos del sistema que contendrá, entre otros:
  - Tablas administrativas del sistema (Roles, tablas básicas, etc.)
  - Información de los usuarios del sistema (Perfil de usuario, vehículo)
  - Información de las personas expertas en fallas mecánicas.
  - Información de los datos OBDII capturados por los dispositivos móviles

- Portal Web del Sistema: Contendrá módulos de administración y estadísticos del Sistema. Así mismo, brindará a las personas expertas las opciones necesarias para que puedan analizar los datos OBDII y comunicarle a cada usuario si su vehículo presenta o no una falla mecánica, ya sea por medio de notificaciones directamente al dispositivo móvil o por otro medio de comunicación

Para el estudio de Proveedores de Servicios en la Nube, se han contemplado opciones de uso gratuitas que ofrezcan servicios como alojamiento, almacenamiento y notificaciones:

- Amazon Web Services (AWS) [51]
  AWS ofrece gran cantidad de productos, entre los que se encuentra una solución informática orientada al desarrollo de aplicaciones móviles en entornos iOS y Android. Esta solución ofrece características tales como: soporte para realizar notificaciones e integración

con redes sociales. Sin embargo, se debe usar un SDK diferente para programar en cada entorno soportado.

Adicionalmente, AWS ofrece una capa de uso gratuito [52] la cual está restringida principalmente al número de horas de uso (750 horas mensuales) y a la capacidad de almacenamiento (20GB en bases de datos relacionales MySQL, Oracle y Microsoft SQL Server o 100MB para el servicio NoSQL).

- Microsoft Azure [53]

  Este proveedor también ofrece herramientas para crear y hospedar servicios orientados a aplicaciones móviles. En éste aspecto soporta iOS, Android, Windows Phone y HTML5 y usa como lenguaje de codificación C# o NodeJS. En cuanto almacenamiento ofrece integración con SQL, Oracle, SAP, MongoDB.

  Por el lado de los sitios Web, estos se pueden crear usando ASP.NET, PHP, Node.js o Python.

  La capa gratuita de éste proveedor 60 minutos de capacidad de CPU al día y contempla 20MB de espacio de almacenamiento para los servicios móviles usando SQL exclusivamente y 1GB para servicios Web. Sin embargo, los servicios móviles aún no están disponibles en América Latina y no es claro si se puede usar una misma base de datos para los servicios Web y móviles.

- Google Cloud Platform [54]

  La plataforma en la nube de Google ofrece un producto llamado App Engine [55] con el cual se pueden crear aplicaciones usando Python, Java, PHP o Go como lenguajes de programación. En cuanto al almacenamiento, ofrece opciones tanto en bases de datos relacionales (MySQL) como no relacionales (NoSQL). Así mismo ofrece una solución Web que permite crear aplicaciones orientadas a dispositivos móviles iOS y Android [56].

  La versión gratuita de la plataforma está limitada a 25 instancias por hora, 1GB de almacenamiento (NoSQL), con un máximo de 50Kb de lecturas/escrituras por hora sobre la base de datos, entre otros.

- Parse [57]

  Proveedor de servicios en la nube enfocado principalmente a entornos móviles, el cual ofrece características tales como: notificaciones, integración social, almacenamiento de datos, alojamiento de páginas Web y un módulo de estadísticas de uso, entre otros. En cuanto al almacenamiento de datos, este se realiza por medio de Bases de Datos orientadas a Objetos, las cuales pueden ser descargadas en cualquier momento que se desee migrar de plataforma.
  Parse ofrece SDK's para desarrollar aplicaciones en iOS, Android, Windows Phone, OS X, Xamarin, Unity o JavaScript. La versión gratuita permite realizar un millón de peticiones mensuales, un millón de notificaciones por mes y 1GB de almacenamiento de base de datos.

III DESARROLLO DEL PROYECTO

Una vez definido el estado del arte del proyecto se hablará de las tareas realizadas en las fases de exploración e iniciación, las cuales están enfocadas a cumplir el objetivo específico número tres.

1. Clientes potenciales

La primera actividad realizada en este aspecto consistió en detectar los grupos de clientes potenciales, para posteriormente, seleccionar el conjunto de Stakeholders hacia los cuales se enfocó el desarrollo de la solución.

A grandes rasgos, los siguientes son los grupos de clientes potenciales que puede abarcar el sistema a desarrollar:

- Persona con Smartphone que disponen de un vehículo con soporte OBDII.
- Talleres automotores.
- Empresas de alquiler de vehículos
- Empresas aseguradoras de vehículos
- Entidades gubernamentales

La selección de los grupos de clientes potenciales se ha realizado contemplando únicamente el ámbito nacional, sin embargo, los grupos definidos son perfectamente aplicables a nivel mundial.

Aunque se desea abarcar todos los grupos de clientes descritos anteriormente, el tiempo destinado al proyecto no lo permite y por ende, es necesario acotar este grupo de clientes.

Habiendo dicho esto, el proyecto se enfocará en:

- Personas residentes en Colombia
- Persona con Smartphone que disponen de un vehículo con soporte OBDII.
- Talleres automotores.

2. Stakeholders

Luego de haber acotado el grupo de clientes potenciales, se definieron los stakeholders primarios del proyecto, en donde se identificaron los siguientes:

- **Usuario de la aplicación móvil (Usuario app)**: Cualquier persona que utilice la aplicación móvil.
- **Administrador del Portal**: Persona encargada de mantener el portal Web de la solución
- **Administrador de taller**: Persona encargada de gestionar y mantener lo relacionado con un taller puntual (Usuarios expertos, datos del taller, servicios, etc.).

- **Gerente de taller**: Usuario de alta gerencia que desea obtener información relevante de un taller puntual
- **Usuario experto**: Usuario capacitado para dar solución a los problemas detectados por la aplicación móvil.

Como lo que se pretende es crear un prototipo, no se tendrán en cuenta de momento todos los Stakeholders, ya que el tiempo es limitado. Por ende, en las tareas siguientes solo se considerarán el Usuario de la aplicación móvil y el Usuario experto.

3. Historias de usuario

Como fase inicial de exploración, se ha realizado un levantamiento de requerimientos en forma de historias de usuario teniendo en cuenta los Stakeholders seleccionados.

Para llevar a cabo la definición de cada historia de usuario se siguió el siguiente esquema:

- **ID**: Identificador de la historia de usuario
- **Actor**: Actor al cual está enfocada la historia de usuario
- **Título**: Descripción corta de la historia de usuario
- **Enunciado**: Descripción más detallada de la funcionalidad que debe cubrir la historia de usuario.
- **Criterios de aceptación**: Labores a realizar que validarán la correcta implementación del enunciado

En la siguiente tabla se observa un aparte de las historias de usuario generadas asociadas a la funcionalidad que debe contemplar la aplicación móvil (app). Si se desea profundizar más al respecto por favor consulte el Anexo 5 (historias de Usuario – Aplicación Móvil) y el Anexo 6 (Historias de Usuarios – Portal Web).

ID	Actor	Título	Enunciado	Criterios de aceptación
1	Usuario App	Seguridad	Como usuario no debo poder ingresar a las opciones de la aplicación móvil si no estoy autorizado para ello	Cuando intento acceder a la aplicación móvil no es posible sin antes tener un usuario registrado en el sistema
2	Usuario App	Crear cuenta	Como usuario de la aplicación móvil (app) debo poder crear una cuenta para acceder al sistema, la cual debe	Puedo crear una nueva cuenta desde la app
				La app me permite leer los términos de uso y la política de privacidad y me exige aceptarlos si quiero crear el usuario

			quedar guardada en la nube	La app me confirma la creación de la cuenta y me da acceso al sistema
				Si intento acceder desde otro dispositivo con los datos de mi nuevo usuario, el sistema me lo permite
3	Usuario App	Autentica-ción	Como usuario de la app debo ingresar mis datos de usuario la primera vez que quiera usar el sistema.	Puedo autenticarme ante el sistema
				La app no me da permisos de uso si no estoy autenticado o mis datos son inválidos
17	Usuario App	Captura de datos OBDII	Como usuario de la app, debo tener una opción de menú que me permita visualizar los datos más relevantes que están siendo capturados en vivo por el dispositivo OBDII	Puedo acceder a una opción de menú que me muestra los datos recibidos desde el dispositivo OBDII

Tabla 4: Aparte del Anexo 5 (Historia de usuario – Aplicación móvil)

4. Selección de las herramientas de desarrollo

Una vez definidos los clientes y sus requerimientos y luego de estudiar y analizar las opciones disponibles en cuanto Dispositivos de Hardware OBDII, Sistemas Operativos Móviles, entornos de desarrollo y proveedores de servicio en la nube, se han elegido las siguientes herramientas para llevar a cabo el proyecto:

4.1 Sistema Operativo: Android OS

La elección del sistema operativo se ha tomado teniendo como base lo siguiente:

- Es el S.O. más común en Smartphones tanto a nivel nacional, como mundial
- No es necesario tener un Sistema Operativo puntual para poder desarrollar sobre la plataforma, dado que para crear aplicaciones iOS es requerido contar con un equipo ejecutando Xcode sobre Mac OS.
- Se tiene facilidad de acceso a la tecnología (personalmente), dado que para el desarrollo del proyecto se cuenta con acceso a varios Smartphone Android.
- Ya existe código libre para leer datos de dispositivos OBDII usando Android [58], lo cual facilitará el proceso de desarrollo

- ❖ Conocimiento del lenguaje y la plataforma: El optar por otro S.O implicaría un periodo de aprendizaje debido a que no se cuenta con los conocimientos básicos para generar una aplicación como la planteada en el presente proyecto en lenguaje Objective-C.
- ❖ Costos asociados al desarrollo: Al tratarse de un S.O libre, los costos de licencias, publicación y desarrollo son menores a los de sus competidores.

4.2 Interfaz de Hardware OBDII: ELM327 Bluetooth

Como que el desarrollo se enfocará en Android, se debe optar por usar la interfaz Bluetooth del dispositivo ELM327, dado que como se mencionó anteriormente, es la versión que funciona sobre dicho Sistema Operativo.

4.3 Entorno de desarrollo Móvil: Multiplataforma

Para desarrollar la aplicación móvil, se optará por un enfoque multiplataforma basado en tecnologías Web, puesto que conocen las tecnologías Web a utilizar (HTML5, JavaScript, CSS3) y gracias a éste enfoque, la aplicación móvil podrá usarse a futuro en otros Sistemas Operativos

Así pues, se usará Apache Cordova para empaquetar la aplicación de forma nativa en Android y al mismo tiempo aprovechar una de las ventajas otorgadas por el Framework, el cual permite crear plugins escritos en lenguaje nativo. Con ello se pretende crear el enlace de comunicación entre el módulo encargado de leer los datos del dispositivos OBDII (escrito también en código nativo de Android) y la aplicación Web.

Como Framework principal de desarrollo se usará Angularjs, debido a que éste emplea un patrón MVC que ayuda a la limpieza y entendimiento del código elaborado. Otra razón importante es que Angularjs se integra sin problemas con Apache Cordova.

Como Framework de interfaz gráfica se usará Ionic puesto que luego de realizar pruebas de concepto con jQuery Mobile, Dojo Mobile y Ionic, éste último fue el que mejores sensaciones arrojó en cuanto a facilidad de uso, funcionalidad suavidad y velocidad. Así mismo, al ser un Framework creado y dependiente de Angularjs, genera menos dependencias de terceros.

4.4 Proveedor de servicios en la nube: Parse

El proveedor que se usará es Parse, dado que ofrece entre otras ventajas, las siguientes:

- ✓ SDK en JavaScript que se integra fácilmente con Apache Cordova – Angularjs, evitando de ésta forma a futuro, el tener que cambiar un esquema de comunicación para cada Sistema Operativo
- ✓ Posibilidad de crear páginas Web usando las mismas tecnologías empleadas en la aplicación móvil (HTML5, JavaScript, CSS3)
- ✓ La versión gratuita tiene menos limitaciones y se puede usar en etapa de pruebas con mayor continuidad.
- ✓ Posibilidad de descargar los datos si se desea migrar a otro proveedor.

- ✓ Ambiente de comunicación seguro, el cual brinda dos métodos de validación de seguridad para evitar que terceros accedan a la información registrada en el repositorio de datos.
- ✓ Comunicación mediante formato JSON, el cual simplifica y agiliza el intercambio de información entre la aplicación móvil y el portal Web

5. Configuración del entorno de desarrollo

Una vez definida las herramientas se procederá a configurar el entorno de desarrollo. A continuación se describen a grandes rasgos las herramientas y los pasos básicos que se deben llevar a cabo para realizar dicha tarea.

5.1 Herramientas transversales

Dado que se eligió un entorno de desarrollo multiplataforma, se tienen varios componentes en común. Por ende, se procederá a hablar de las herramientas transversales que servirán para desarrollar tanto la aplicación en la nube, como la aplicación móvil:

- ❖ **SVN**: Subversion o SVN es un sistema de control de versiones que permite llevar un histórico sobre las moficaciones realizadas a los proyectos. Para ello se usará el repositorio comercial provisto por xp-dev (https://xp-dev.com/) ya que se cuenta con una suscripción anual.
- ❖ **Java JDK**: El JDK de Java es un requerimiento casi obligatorio para el resto de herramientas a utilizar. Adicionalmente, el JDK se usará para el desarrollo de la aplicación móvil. Con esto en mente, se sugiere descargar la versión más reciente de Java JDK, la cual se puede actualmente se encuentra en la página Web oficial de Oracle: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.
  Una vez instalado el JDK, se puede abrir una consola de comandos y digital "java -version". Si después de ejecutado dicho comando no se muestra el número de la versión, se debe verificar la instalación. Para más información, consultar la página Web oficial.
- ❖ **NodeJS**: NodeJS es un entorno de programación basado en JavaScript y está pensado para crear aplicaciones escalables. Esta herramienta será utilizada para la instalación de varios Frameworks a utilizar durante el proyecto. El instalador de puede descargar desde la página oficial de NodeJS (http://nodejs.org/). Se sugiere instalarlo preferiblemente con la configuración que trae por defecto. Al igual que Java JDK, se puede abrir una consola de comandos y digital "**npm -v**". Esto deberá mostrar la versión del Framework instalado.
- ❖ **Git**: Git es un sistema de control de versiones de código abierto y será necesario para poder instalar y usar adecuadamente otras herramientas, tales como Bower. El instalador puede ser descargado desde la página oficial del proyecto (http://git-scm.com/)

- **Yeoman**: Este Framework puede ser fácilmente instalado usando NodeJS. Para ello, basta con digitar el siguiente comando en consola: "**npm install -g yo**". Esta instrucción descargará 3 herramientas:
  - **Yo:** Sirve para crear la estructura inicial de la aplicación Web
  - **Grunt:** Usado para construir, pre visualizar y ejecutar tareas sobre el proyecto, tales como pruebas, compilación, unificación y reducción de tamaño de archivos, entre otros. Más información en http://gruntjs.com/.
  - **Bower:** Usada para el manejo de paquetes y dependencias. Más información en: http://bower.io/

  Una vez terminado el proceso, se puede ejecutar el comando "**yo -v**" para validar la correcta instalación del Framework.
  Si se desea, se pueden obtener instrucciones más detalladas sobre la instalación de Yeoman en la página Web del Framework: http://yeoman.io/
- **Ruby:** Aplicación requerida para el correcto funcionamiento de SASS. El instalador se puede descargar desde la página oficial: (http://www.rubyinstaller.org/). Una vez finalizada su instalación, se puede verificar que funcione adecuadamente ejecutando en la consola el comando "**gem -v**".
- **Compass:** Framework necesario que funciona junto con Ruby para generar hojas de estilos en cascada. Su instalación se realiza por consola mediante el comando "**gem install compass**"
- **SASS:** Sass es un Framework de metalenguaje para hojas de estilo en cascada (CSS), el cual se usará para automatizar algunas tareas de generación de hojas de estilos. Su instalación se realiza por medio de Ruby mediante el comando "**gem install sass**". La verificación de la instalación se realiza mediante el comando "**sass-v**".
- **Apache Ant:** Este Framework será necesario para ejecutar algunas tareas de automatización y generación de código, principalmente por de NodeJS. Para instalarlo se debe descargar su distribución binaria la cual se encuentra en la página oficial del proyecto (http://ant.apache.org/bindownload.cgi). Una vez descargado, se puede descomprimir en el lugar deseado y posteriormente se debe crear una variable del sistema llamada **ANT_HOME** que apunte a la carpeta donde quedó descomprimida la distribución (Ej: **c:\desarrollo\Ant\;**). Adicionalmente se debe adicionar a la variable **PATH** del sistema la ruta a la carpeta "**bin**" de Ant (Ej: **%ANT_HOME%\bin;**).
  Para comprobar la instalación se puede ejecutar por consola el comando "**ant -version**"
- **Angularjs:** Framework para desarrollo de aplicaciones HTML escrito enteramente en javascript (https://angularjs.org/). Para su uso es suficiente con incluir el tag <script> en la página web referenciado a la versión que se quiera usar. En el caso puntual del proyecto se usará la versión 1.2.8 disponible en: http://ajax.googleapis.com/ajax/libs/angularjs/1.2.8/angular.js
- **Protractor:** Framework de pruebas e2e (End-to-End) para aplicaciones creadas con **Angularjs** (http://angular.github.io/protractor/). El framework es un programa escrito en Node.js capaz de correr pruebas contra una aplicación corriendo en un explorador Web e interactuando como si de un usuario real se tratara.
  Dado que Angularjs será el Framework base en el desarrollo tanto de la aplicación móvil como el portal Web, se realizarán las pruebas usando éste Framework.

- **Parse:** Proveedor de servicios en la nube elegido (http://parse.com). Para su uso se debe crear una cuenta y posteriormente crear una aplicación. Esto creará un identificador el cual permitirá:
  - Identifficar y autorizar el acceso al repositorio de datos del proyecto
  - Acceder al servicio de envío de notificaciones
  - Usar el SDK de Parse.

  Aunque Parse proporciona SDK's creados en lenguaje nativo para interactuar con sus servicios en este caso se usará la versión en Javascript, dado que dicho SDK se puede usar tanto en la aplicación móvil como en el portal Web. Así mismo, cabe resaltar que el SDK de Javascript cuenta con un nivel de seguridad adicional dado que para poder usarlo se requiere de una llave única, la cual es asignada al momento de crear la cuenta en el portal Web de Parse. Todo ello pensado para garantizar la seguridad al momento de establecer la comunicación.

5.2 Aplicación móvil

Una vez configuradas las herramientas y Frameworks transversales, se procederá a configurar el entorno de desarrollo de la aplicación móvil.

- **Android SDK**: El SDK de Android se puede descargar desde la página Web para desarrolladores de dicha plataforma (http://developer.android.com/sdk/index.html?hl=sk). En este caso se ha descargado el "SDK Tools for Windows".
  Una vez instalado se aconseja seguir la recomendación del instalador y abrir el SDK Manager para descargar las imágenes y demás herramientas a usar más adelante.
  Desde el SDK Manager se pueden descargar y mantener actualizadas las imágenes de Android sobre las cuales se desea crear la app, así como también algunas herramientas extras, tales como el controlador USB que permitirá hacer depuración de código directamente sobre un Dispositivo Android.
  Los paquetes recomendados para descarga son: Tools, Android 4.3 (API18), Extras -> Android support Library y Extras -> Google USB Driver.
  Finalmente, se debe agregar a la variable **PATH** del sistema la ruta a las carpetas: "**platform-tools**" y "**tools**", las cuales están ubicadas en el directorio de instalación del SDK (EJ: **C:\desarrollo\android-sdk\platform-tools;C:\tesis\android-sdk\platform-tools;**)
  Para comprobar la instalación se puede ejecutar por consola el comando "**adb version**"

- **Apache Cordova/Phonegap**: Phonegap o Apache cordova se puede instalar por línea de comandos usando NodeJS o descargando un archivo comprimido. En este caso se usará la opción por línea de comandos, para ello se debe ejecutar la instrucción "**npm install --global cordova**". Con el comando "**cordova -v**" se puede verificar la correcta instalación del Framework. Para más detalles sobre el Framework se puede consultar las direcciones https://cordova.apache.org/ o http://phonegap.com/.
  Una vez instalado, se debe hacer una modificación para que al momento de dar soporte para Android, se busque la version del API que se ha descargado. En este caso se ha

usado la versión 18 (Android 4.3 API18). Para realizar este cambio se debe modificar el archivo "**project.properties**" ubicado en la siguiente carpeta: **C:\Users\%USUARIO%\.cordova\lib\android\cordova\3.XX\framework**.
El cambio consiste en asignar el API adecuado a la variable **target**. (EJ: **"target=android-18").**

- **AngularJS Generator**: Como su nombre lo indica, se trata de un generador que permite crear esquemas de proyectos basados en **AngularJS** implementando las buenas prácticas de desarrollo. Para instalarlo se debe ejecutar el siguiente comando en la consola: "**npm install –g nenerator-angular**". Para detalles sobre el generador, se puede consultar el repositorio del proyecto, el cual se encuentra en: https://github.com/yeoman/generator-angular

- **Ionic Framework**: Framework de presentación enfocado a dispositivos móviles creado por AngularJS que usa una versión reducida de dicho Framework. La instalación de Ionic también se realiza por consola mediante el comando "**npm install –g ionic**". Al finalizar, se puede verificar la instalación mediante el comando "**ionic -v**". Para más detalle sobre el Framework se puede visitar la página Web oficial: http://ionicframework.com/

5.3 Portal Web

Por último se configurarán las herramientas exclusivas para la creación del portal Web.

- **Parse-angular-demo:** Esqueleto de proyecto creado Brandid (www.brandid.com) el cual se puede descargar de la página Web: https://github.com/brandid/parse-angular-demo. Se eligió este proyecto como base dado que ha sido creado teniendo en cuenta buenas prácticas de desarrollo y está orientado a su ejecución en ambientes de producción real [59]

- **Consola parse:** Como su nombre lo indica, se trata de un programa que corre por línea de comandos y nos permitirá crear la aplicación localmente y subirla al servidor de parse (https://www.parse.com/downloads/windows/Parse/latest). Para su funcionamiento basta con descargar el archivo y descomprimirlo

5.4 Ambiente de pruebas

Para el desarrollo del Sistema se han contemplado dos tipos de pruebas, la cuales se describen a continuacuión:

Pruebas automáticas:

Inicialmente la metodología se planteó para realizar el desarrollo orientado a pruebas (TDD), pero finalmente se ha optado por hacer el desarrollo orientado a comportamiento o BDD (Behavior-Driven Development [63]), dado que se le considera la evolución de TDD.

Por eso se optó por usar Protractor, ya que usa Jasmine [64] como interfaz de Framework para pruebas (Framework BDD) y se integra perfectamente con Angularjs.

A grandes rasgos lo que hace Protractor es ejecutar un servidor de pruebas Selenium [65] que se encarga de controlar el navegador Web y sobre este ejecutar las pruebas de comportamiento.

Para ello se deben crear los archivos que especifican la configuración del ambiente de pruebas, así como las pruebas a ejecutar de comportamiento a ejecutar.

Así pues lo primero que se debe hacer es crear el archivo de configuración del entorno de pruebas. A continuación se puede observar el archivo **conf.js** creado para ejecutar una prueba sobre la aplicación móvil:

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/wd/hub',
  specs: ['spec.js'],//lista de archivos de pruebas a ejecutar
  multiCapabilities: [{
    browserName: 'chrome'//browser en el cual se probara
  }]
}
```

Como se observa en el script anterior, se ha definido un archivo llamado **spec.js.** Este archivo será el encargado de contener la prueba de comportamiento a ejecutar, la cual en este caso corresponden a una prueba de autenticación que se ejecuta en el sistema móvil. El detalle de este script se puede observar con mayor detalle a continuación:

```
describe('Doc2Car movil', function() {
        var fs = require('fs');
        var nextButton = element(by.id('next'));
        var loginButton = element(by.id('bLogin'));
        var loginInButton = element(by.id('bLoginIn'));
        var logoutButton = element(by.buttonText('OK'));
        var menuWelcomeMenu = element(by.id('idx'));
        var menu5Menu = element(by.id('menu5'));
        var iUserTxt = element(by.id('iUser'));
        var iPwdTxt = element(by.id('iPwd'));
        var width = 360;
        var height = 640;
        var imgNumber=10;
        function takeScreenshot(){
                browser.sleep(500);
                browser.takeScreenshot().then(function (png) {
                        writeScreenShot(png, './img/'+imgNumber+'.png');
                        imgNumber++;
                });
        }
        function writeScreenShot(data, filename) {
                var stream = fs.createWriteStream(filename);
                stream.write(new Buffer(data, 'base64'));
                stream.end();
        }
        beforeEach(function() {
                browser.get('http://localhost:9000/#/begin/true');
                browser.driver.manage().window().setSize(width, height);
        });
        it('should begin', function() {
                //intro
                takeScreenshot();
                nextButton.click();
                takeScreenshot();
                nextButton.click();
                takeScreenshot();
                nextButton.click();
                takeScreenshot();
                nextButton.click();
                takeScreenshot();
                nextButton.click();
                takeScreenshot();
                loginInButton.click();
                //autenticacion
                takeScreenshot();
                iUserTxt.sendKeys('rmcruzv');
                iPwdTxt.sendKeys('*****');
                loginButton.click();
                browser.sleep(10000);
        });
});
```

Una vez codificadas las pruebas de comportamiento es suficiente con arrancar el servidor de pruebas e indicarle a Protractor cuál es el archivo de configuración para que este comience a ejecutarlas. Esto se logra con los siguientes comandos:

- `webdriver-manager start`
- `rotractor conf.js`

Pruebas unitarias:

Para facilitar la realización de las pruebas unitarias se usó Grunt, el cual en resumen es un programa en node.js que sirve para ejecutar diferentes tareas automáticas. Así pues, se han

usado scripts (Ej. Anexo 7: Archivo de Configuración Grunt.js) que automatizan los siguientes procesos sobre el código:

- Verificación de errores y precauciones en los archivos .js.
- Minimización de códigos (HTML, *.css, *.js) que quita espacios, saltos de línea, comentarios.
- Afeado de código: Cambio de nombres de variables para reducir el tamaño de los archivos generados, especialmente en archivos *.js.
- Minimización del tamaño de las imágenes.
- Concatenación de archivos.
- Ejecución de pruebas automáticas de Protractor.
- Copia de archivos a carpeta de distribución.
- Iniciación del servidor de pruebas unitarias.
- Ejecución de tareas que validan los cambios en los archivos y los envían a la carpeta de producción para que estén disponibles en el servidor de pruebas unitarias. Ej.: Los cambios en los archivos de definición de hojas de estilos son escuchados por compas, el cual recrea y publica el archivo *.css correspondiente.

Estos procesos se agrupan en tareas que ejecutan varias otras a la vez. En el caso puntual del script hay tareas para diferentes propositos:

- **grunt test**: Compilar y ejecutar las pruebas automáticas de protractor
- **grunt build**: Pre-construir la aplicación para su distribución en un ambiente de producción
- **grunt serve**: Arrancar el servidor de pruebas realizando antes tareas de validación, compilación y copiado a la carpeta de producción
- **grunt**: Opción por defecto que ejecuta todas las tareas anteriores

Así por ejemplo si se desea arrancar el servidor de pruebas se debe ejecutar por consola la siguiente instrucción:

- `Grunt serve`

En resumen, usando Grunt fue posible realizar las pruebas unitarias tanto de la aplicación móvil como del Portal Web en un ambiente local por medio de en un navegador Web de escritorio. A continuación se muestran un par de figuras que evidencia la ejecución en un ambiente local

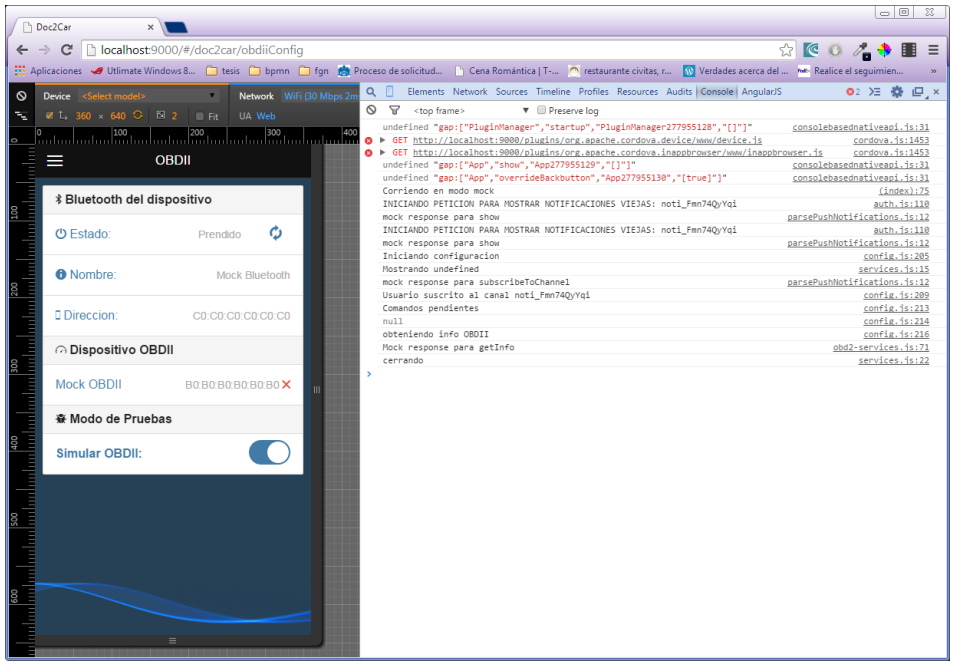

Figura 10: Pruebas unitarias locales - doc2car

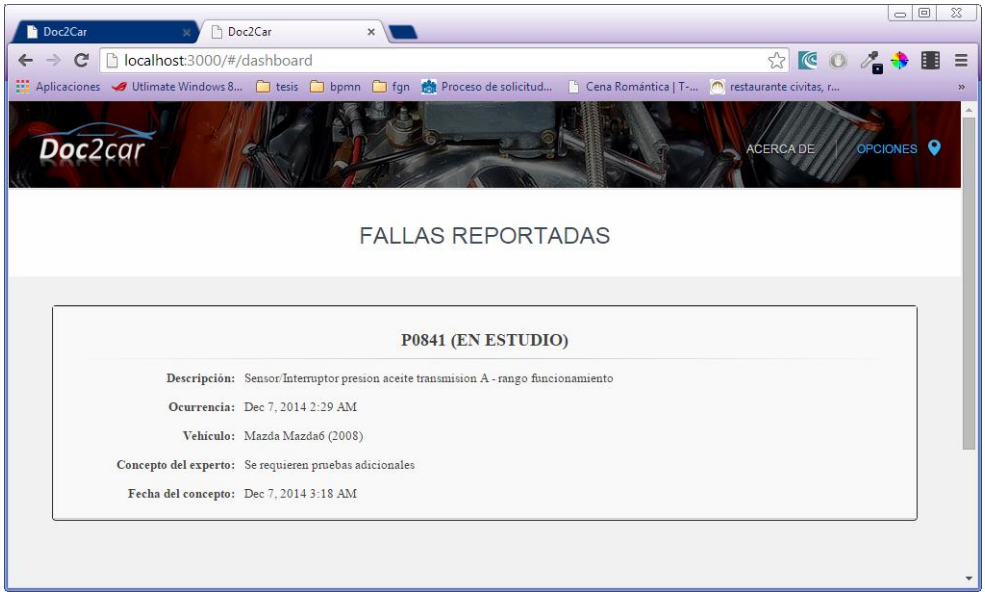

Figura 11: Pruebas unitarias locales - Portal Web

6. Diseño de la solución propuesta

6.1 Diagrama de componentes

La arquitectura propuesta está dividida principalmente en dos grandes grupos: Por un lado se tienen los componentes alojados en el proveedor de servicios en la nube (Portal Web y Repositorio de datos) y por el otro, los componentes pertenecientes a la aplicación móvil. Así mismo se puede observar la presencia del patrón MVC (Modelo Vista Controlador) en los dos componentes principales. En este sentido, se puede ver que ambos sistemas usarán un mismo repositorio de datos, el cual estará alojado en la Nube. Finalmente se debe resaltar que la lógica de envío de notificaciones se realizará por medio del servidor de notificaciones provisto por el proveedor de servicios en la nube (Parse) y del lado de la aplicación cliente se consumirán dichas notificaciones mediante la utilización de un plugin creado en código nativo Android (plugin que también ofrece soporte para iOS).

La siguiente Figura muestra el diagrama de componentes de la solución propuesta:

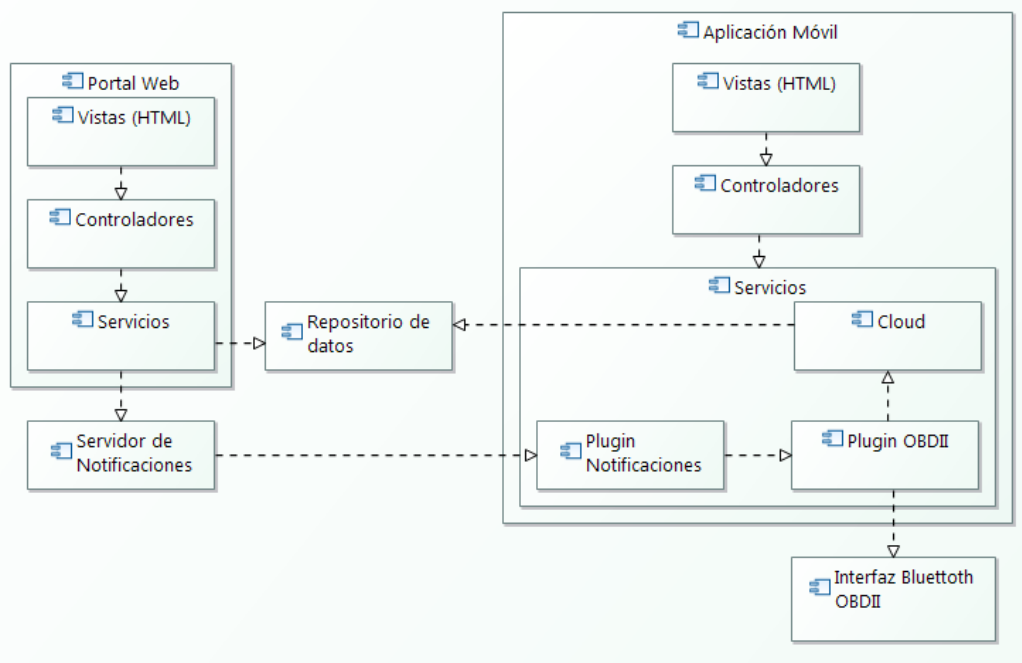

Figura 12: Diagrama de componentes

6.2 Diagrama de Base de Datos

Una vez definida la arquitectura se procedió a realizar el diagrama de base de datos, pero debido a que el modelo de datos que ofrece Parse es no-SQL [60], el diagrama varía un poco dado que se asemeja más a un diagrama de clases que a uno de entidad-relación. En la siguiente figura se puede observar dicho modelo.

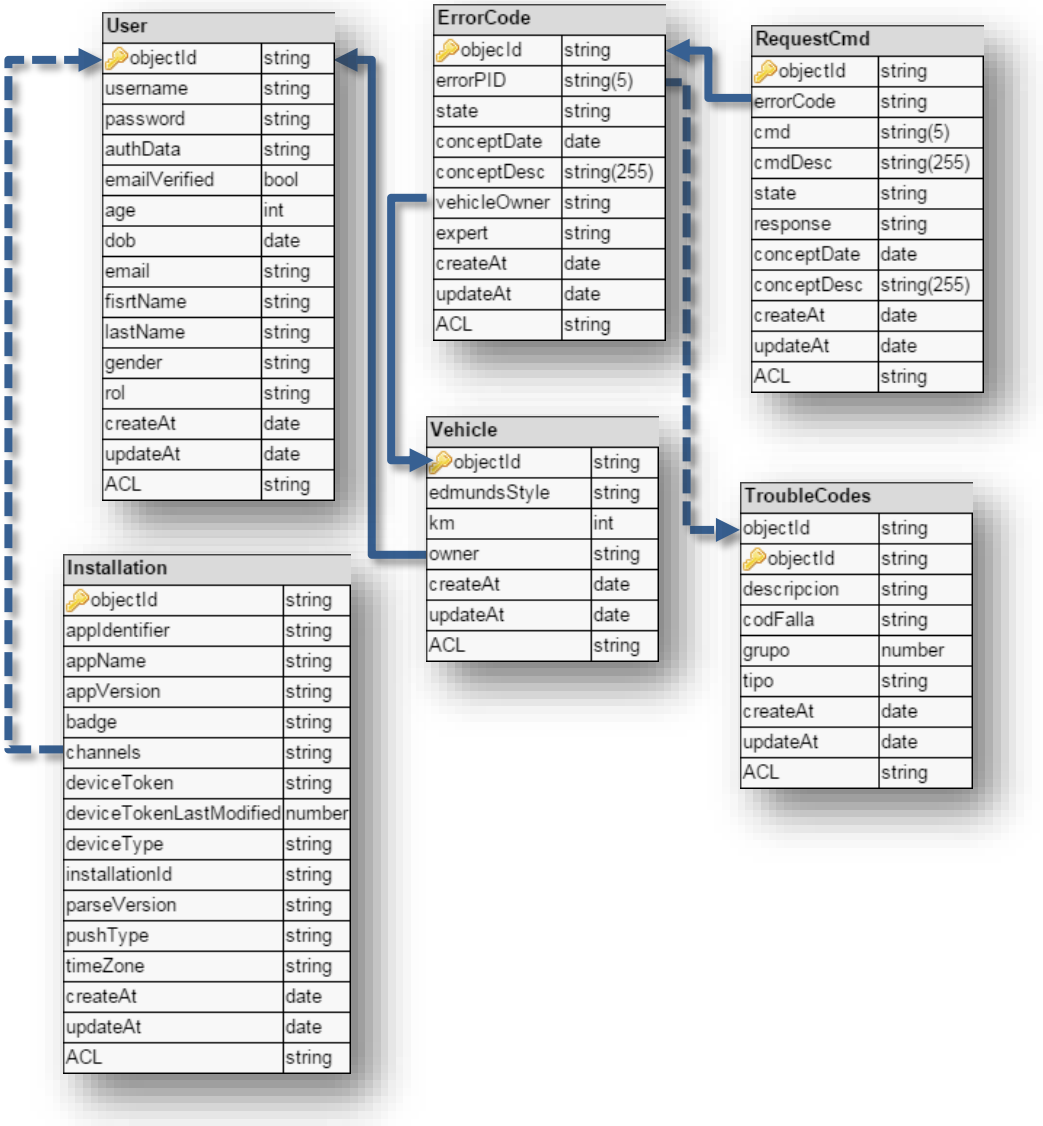

Figura 13: Diagrama de Base de Datos no-SQL

Como se puede apreciar, todas las tablas (o clases) contienen una llave única (objectId) y los campos createAt, updateAt y ACL (agregados automáticamente por el proveedor de servicios Parse luego de crear cada clase).

A continuación se hace una descripción del contenido de cada clase:

- **User**: Esta clase se usa para persistir la información de los usuarios de la aplicación móvil y de aquellos usuarios expertos que pueden acceder al portal Web. Para validar

el acceso se hace uso del campo "rol" de la clase, el cual tiene como valores posible: *MOVIL* o *EXPERTO*.

- **Installation**: Esta clase es creada automáticamente cuando se activa el servicio de notificaciones Push ofrecido por Parse. Dicha tabla es poblada cuando el usuario de la aplicación móvil acepta suscribirse al servicio de notificaciones, creando así un canal (channel) exclusivo para el usuario definido por el prefijo "noti_" seguido del identificador del usuario "objectId".

- **Vehicle**: Clase pensada inicialmente para contener los datos del vehículo del conductor, sin embargo, posteriormente se optó por usar el Vehicle API de edmunds Developer Network [61]. Esta decisión se tomó debido a que se encontraron varias ventajas:
  - Se ahorra tiempo en la búsqueda de datos de prueba.
  - Es de uso gratuito
  - Las consultas son por medio de peticiones GET
  - La información se maneja en formato JSON
  - Contiene la información de todos los vehículos comercializados es Estados Unidos, por ende todos soportan la regulación OBDII
  - El único requerimiento para su uso es no guardar la información localmente, por ende solo se guarda la identificación final del vehículo en la variable: "edmundsStyle".

- **TroubleCodes**: En esta tabla se almacenó la lista de códigos de falla estándar (ISO9141), para ello se usó una pequeña parte de los códigos definidos por la regulación. Los datos de prueba fueron tomados de [62].

- **ErrorCode**: Tabla que contiene los errores capturados por la aplicación móvil.

- **RequestCmd**: Tabla que contiene los datos de las peticiones de ejecución de comandos realizada por el usuario experto.

6.3 Diagrama de Despliegue

Luego de aclarar los aspectos asociados a los componentes y herramientas usadas, se puede observar a continuación el diagrama de despliegue del Sistema, junto con las herramientas involucradas en cada componente del Sistema.

El diagrama también ilustra los protocolos de comunicación usados en el sistema, mostrando que para la comunicación con el proveedor de servicios en la nube (Portal Web, repositorio de datos, servicio de notificaciones) se usaron peticiones REST (JSON) bajo un protocolo HTTP. Así mismo, se observa que para la comunicación entre la aplicación móvil y la interfaz de hardware OBDII se usó un Socket de comunicación sobre una red Bluetooth, el cual permitió enviar y recibir comandos en formato Hexadecimal.

Adicionalmente en el diagrama se ubican las herramientas y tecnologías usadas para la creación de cada componente del sistema, así como también las herramientas transversales usadas durante el desarrollo del proyecto.

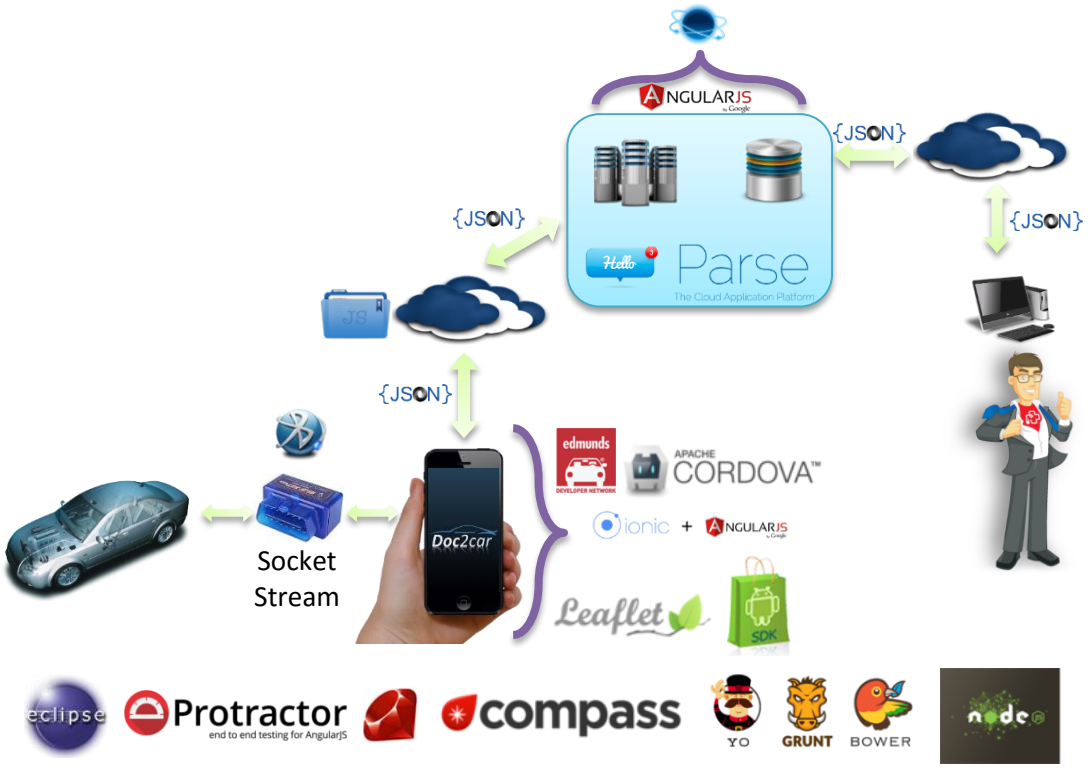

Figura 14: Diagrama de despliegue y herramientas usadas

7. Descripción del sistema creado (doc2car)

El sistema que se desarrolló lleva como nombre **doc2car** y está formado por dos proyectos: el primero enfocado a la funcionalidad de la aplicación móvil y el segundo destinado a la funcionalidad del Portal Web, tal como se puede observar en la sección de "Diagrama de componentes".

A continuación se mostrará un aparte de las principales funciones abarcadas por cada proyecto. Para ver el detalle completo del Sistema, por favor referirse al Anexo 8.

7.1 Aplicación móvil

La aplicación móvil contiene la funcionalidad enfocada a los conductores de vehículos y está dividida en varios módulos, teniendo como principales los siguientes:

- **Configuración**: Este módulo está disponible en la primera opción del menú y le permite al usuario ajustar las variables de la aplicación

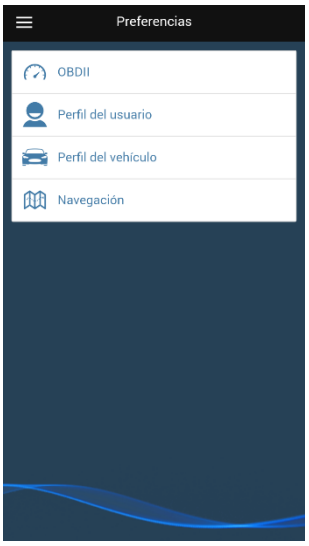

Figura 15: submenú de configuración - doc2car

- **OBDII**: En esta pantalla el usuario puede ver la información básica del Bluetooth, prenderlo, enlazar la aplicación con una Interfaz de Hardware OBDII o habilitar la aplicación para que se ejecute en "Modo de pruebas".

  Adicionalmente, esta pantalla se encarga de hacer la ejecución de los comandos solicitados por el experto y del envío de los resultados a la nube.

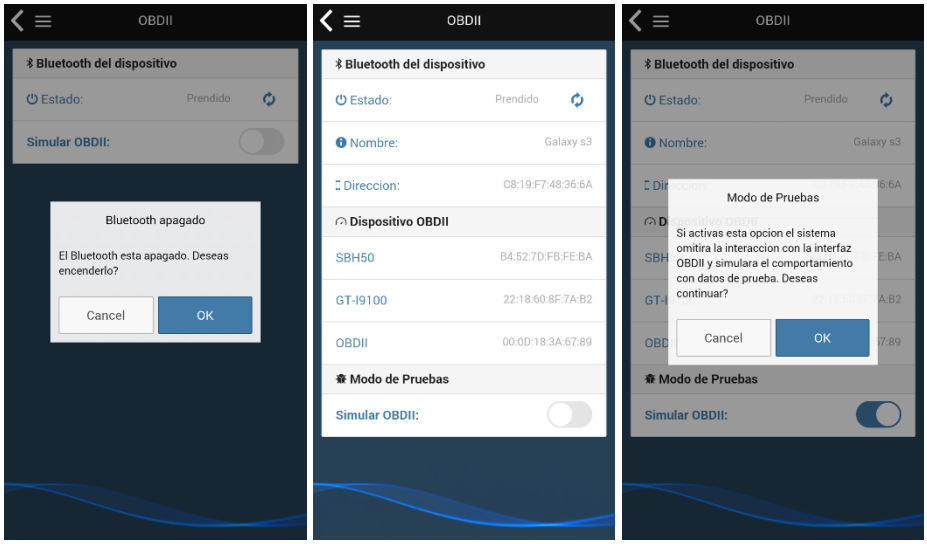

Figura 16: Pantallas de Configuración OBDII - doc2car

Si se activa el modo de pruebas, la aplicación creará una interfaz Bluetooth dummy y hará uso de los mockups creados en la etapa de desarrollo para simular el comportamiento de comunicación con la interfaz OBDII. Este modo reportará 2 códigos de falla de prueba: **P0003** y **P0841**

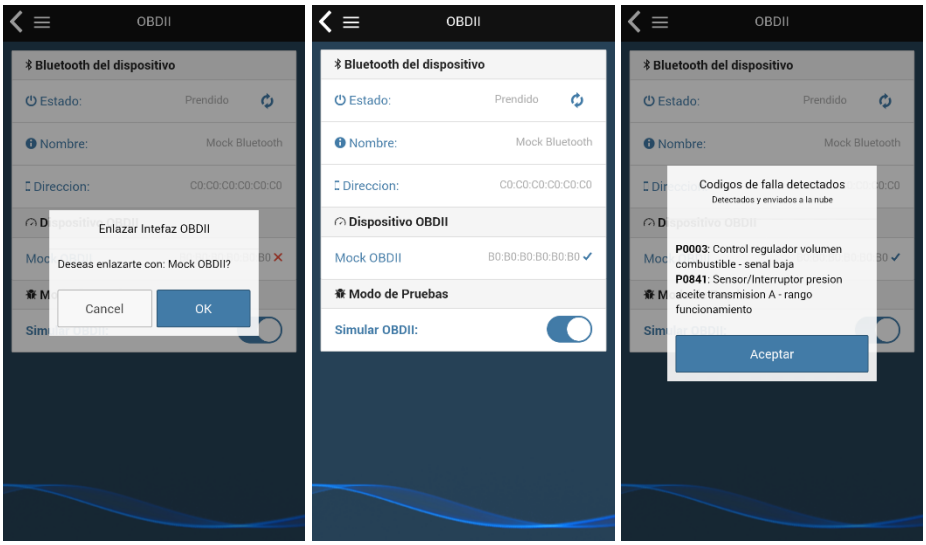

Figura 17: OBDII - Modo de Prueba

- **Sensores**: Esta pantalla está compuesta de seis pestañas y en cada una de ellas se muestra el resultado de ejecutar los comandos básicos definidos en el plugin de conexión con la interfaz OBDII. Estos comandos se ejecutan cada segundo en promedio. Sin embargo cuando se activa el modo de pruebas la información mostrada es simulada con base en los valores máximos permitidos para cada comando.

  Entre los comandos que se ejecutan se encuentran: Velocidad, revoluciones, nivel de combustible, temperatura del líquido refrigerante, consumo de combustible actual, presión del acelerador, presión barométrica y códigos de falla detectados, entre otros.

Figura 18: Pantallas de Sensores - doc2car

- **Notificaciones**: Dado que la aplicación móvil se ejecuta en segundo plano, está en capacidad de abrirse al recibir una notificación y mostrar el concepto emitido por el experto. Del mismo modo puede mostrar la solicitud de ejecución de comandos adicionales si así lo ha solicitado el usuario experto.

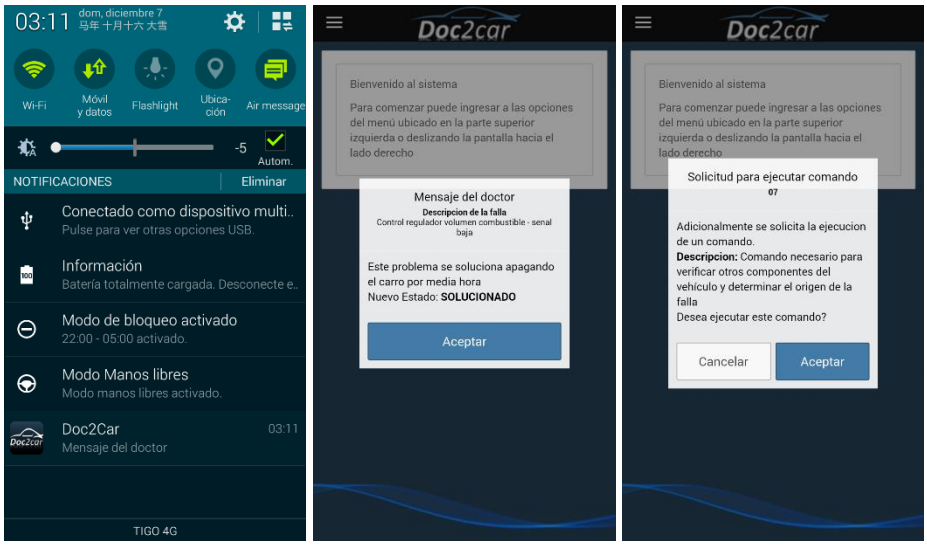

Figura 19: Notificaciones - doc2car

7.2 Portal Web

El portal Web se ha alojado en el proveedor de servicios en la nube de Parse y se puede consultar por medio de la siguiente URL:

- **doc2car.parseapp.com**

Este portal Web contiene la funcionalidad básica que le permite a un usuario experto ingresar al sistema, consultar la lista de códigos de falla reportados (DTC), ver el detalle de cada falla, emitir un concepto y en caso de ser necesario, solicitar la ejecución de nuevos comandos en el vehículo del conductor que reporta la falla. A continuación se muestra a grandes rasgos la funcionalidad de dicho portal

- **Autenticación**: Pantalla que valida el acceso al sistema por parte de un usuario. Este proceso al igual que en la aplicación móvil se realiza contra el repositorio de datos de Parse.

  Para probar su funcionamiento se pueden usar los siguientes datos de prueba:
  - Usuario: **experto**
  - Contraseña: **12345**

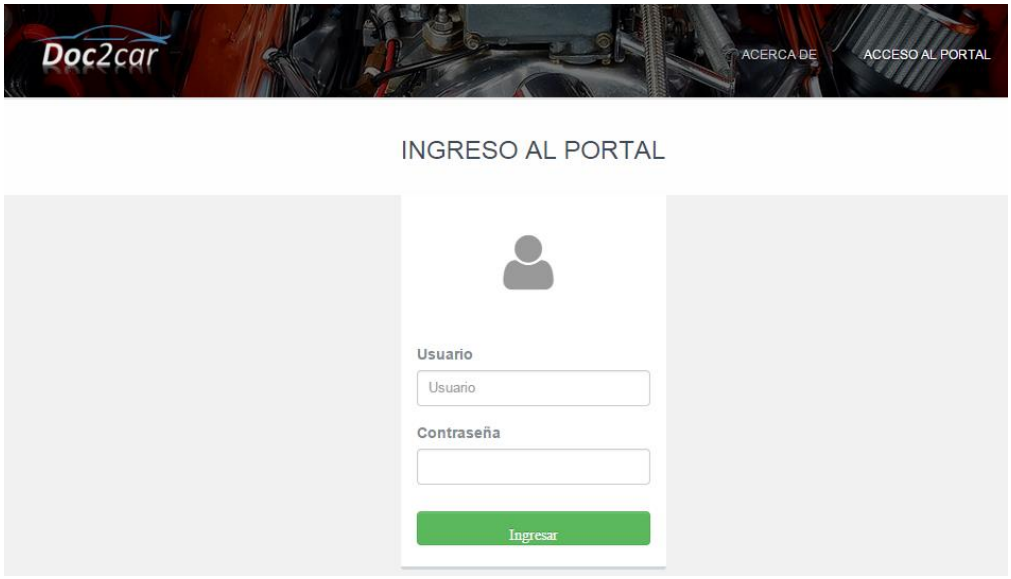

Figura 20: Pantalla de Autenticación - Portal Web

- **Fallas reportadas:** La siguiente pantalla se le muestra al usuario luego de realizar el proceso de autenticación, La información que se muestra en la pantalla corresponde a la lista de fallas que han sido reportadas por los conductores que usan la aplicación móvil

Figura 21: Lista de Fallas Reportadas – Portal Web

- **Detalle de la falla deportada:** Si el usuario quiere ver el detalle de cada falla, puede hacerlo seleccionando el registro deseado, con lo cual se abrirá una ventana emergente, la cual se compone de 3 pestañas.
  La primera pestaña muestra el resumen de la falla.

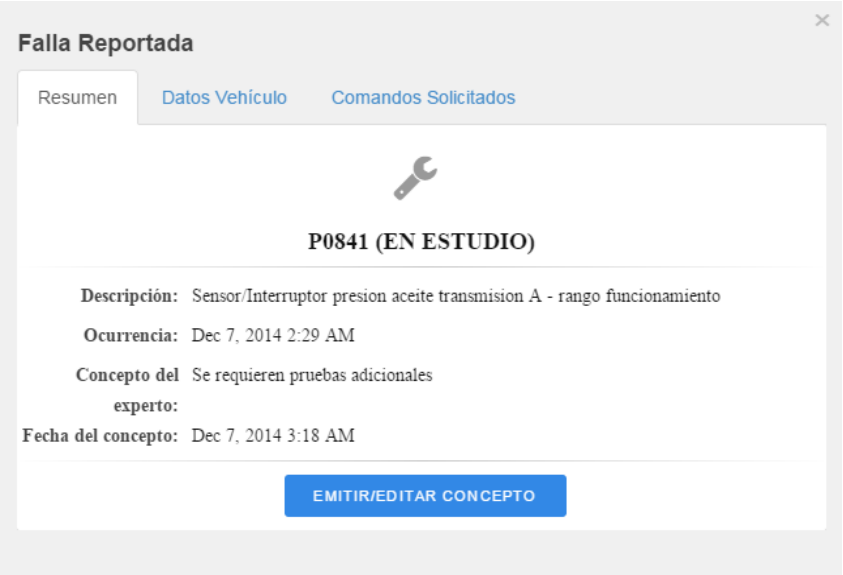

Figura 22: Resumen de la falla - Portal Web

La segunda pestaña muestra los datos más relevantes del vehículo, los cuales son consultados nuevamente al portal de **edmunds** usando el identificador de estilo que se ha guardado en el repositorio de datos del sistema.

Figura 23: Datos del vehículo afectado - Portal Web

La última pestaña corresponde a la lista de comandos que el experto ha solicitado al conductor para que sean ejecutados remotamente. En caso tal que el usuario experto no haya solicitado la ejecución de comandos, dicha pestaña no aparecerá.

Figura 24: Comandos solicitados - Portal Web

- **Emitir concepto:** Tanto en el resumen de la falla, como en la lista de comandos solicitados se puede emitir el concepto del usuario experto, en cuyo caso se abrirá una segunda ventana emergente. Esta ventana permite adicional a la emisión del concepto, la ejecución de comandos en el vehículo del conductor, en cuyo caso se le solicitarán los datos del comando a ejecutar y una descripción sobre el comando solicitado.

  Una vez diligenciado el formulario, el usuario podrá ver un mensaje de confirmación de la operación y se actualizará la lista de fallas. Si la falla ha sido solucionada, esta no aparecerá más en el listado.

Figura 25: Emitir concepto / Solicitar Comando - Portal Web

IV VALIDACIÓN DEL SISTEMA

Durante todo el proceso de creación del sistema planteado se realizaron pruebas orientadas al comportamiento, tal como se definieron en la sección anterior (Ambiente de pruebas). Esto permitió que todo el software desarrollado funcionase acorde a las pruebas que definieron su comportamiento.

Para ello los pasos que se realizaron para validar cada módulo del sistema fueron:

i. Definir el comportamiento de aceptación con base en la historia de usuario.
ii. Codificar la prueba de comportamiento en Javascript orientada a ser probada con Protractor. Como ejemplo se muestra el siguiente fragmento de código que representa la codificación realizada para validar el proceso de autenticación de la aplicación móvil.

```
describe('Doc2Car movil', function() {
  var fs = require('fs');
  var width = 360;
  var height = 640;
  var imgNumber = 10;
  function takeScreenshot() {
    browser.sleep(500);
    browser.takeScreenshot().then(function(png) {
      writeScreenShot(png, './img/' + imgNumber + '.png');
      imgNumber++;
    });
  }
  function writeScreenShot(data, filename) {
    var stream = fs.createWriteStream(filename);
    stream.write(new Buffer(data, 'base64'));
    stream.end();
  }
  beforeEach(function() {
    browser.get('http://localhost:9000/#/begin/false');
    browser.driver.manage().window().setSize(width, height);
  });
  if('deberia autenticar el usuario', function (){
    // autenticacion
    takeScreenshot();
    element(by.id('bLoginIn')).click();
    element(by.id('iUser')).sendKeys('rmcruzv');
    element(by.id('iPwd')).sendKeys('12345');
    element(by.id('bLogin')).click();
    browser.sleep(5000);
  });
});
```

iii. Modelar la interfaz gráfica acorde al comportamiento deseado. En este punto las interfaces se modelaron usando el mismo Framework de desarrollo ya que tanto Angular como Ionic implementan el patrón MVVM [66], el cual permite abstraer totalmente la vista del modelo. Por lo tanto las interfaces funcionaban aunque no hubiese un modelo que lo soportara. El siguiente fragmento muestra lo que sería la interfaz inicial de autenticación.

```
<ion-view title="Autenticarse" hide-back-button="true">
 <ion-content has-header="true" padding="true">
  <ion-nav-buttons side="left">
   <button class="button button-icon icon ion-chevron-left" nav-clear ng-click="#"></button>
  </ion-nav-buttons>
  <div class="card form pop-in" >
   <div class="item item-text-wrap">
    <div class="list">
     <label class="item item-input"><span class="input-label">Usuario</span>
      <input type="text" ng-model="user.username" id="iUser">
     </label>
     <label class="item item-input"><span class="input-label">Contrase&ntilde;a</span>
      <input type="password" ng-model="user.password" id="iPwd">
     </label>
     <div class="assertive" ng-if="error.message">{{error.message}}</div>
     <button class="button button-block button-positive" nav-clear ng-click="login()" id="bLogin">
      Ingresar</button>
     <a class="button button-block button-clear button-positive" nav-clear>Olvidaste tu
     contrase&ntilde;a?</a>
    </div>
   </div>
  </div>
 </ion-content>
</ion-view>
```

El siguiente es el mismo fragmento en su versión final

```
<ion-view title="Autenticarse" hide-back-button="true">
 <ion-content has-header="true" padding="true">
  <ion-nav-buttons side="left">
   <button class="button button-icon icon ion-chevron-left" nav-clear
       ng-click="goBack()" ng-if="data.showBack" id="menuLogin"></button>
  </ion-nav-buttons>
  <div class="card form pop-in" >
   <div class="item item-text-wrap">
    <div class="list">
     <label class="item item-input"><span class="input-label">Usuario</span>
      <input type="text" ng-model="user.username" id="iUser">
     </label>
     <label class="item item-input"><span class="input-label">Contrase&ntilde;a</span>
      <input type="password" ng-model="user.password" id="iPwd">
     </label>
     <div class="assertive" ng-if="error.message">{{error.message}}</div>
     <button class="button button-block button-positive" nav-clear ng-click="login()" id="bLogin">
      Ingresar</button>
     <a class="button button-block button-clear button-positive" nav-clear  ui-
     sref="doc2car.forgot">Olvidaste tu contrase&ntilde;a?</a>
     <a class="button button-block button-clear button-positive" nav-clear  ng-
     click="goToBegin()">Mostrar Introducci&oacute;n</a>
    </div>
   </div>
  </div>
 </ion-content>
</ion-view>
```

iv. Simular la lógica en la capa de modelos y servicios para poder ejecutar la prueba satisfactoriamente.
El siguiente fragmento muestra la implementación de la lógica simulada usada para probar la ejecución de comandos en la aplicación móvil.

Esta lógica se mantuvo finalmente en la aplicación y se dejó activa para poder probar la funcionalidad asociada a OBDII sin necesidad de tener una interfaz de Hardware.

```
var exec = function(successCallback, errorCallback, method, params) {
 // modo de prueba
 if ($rootScope.runMode.mockMode) {
  var data;
  switch (method) {

   ...
   case 'runCommand':
    var mockResult = getMockValue(params); // se genera un valor simulado
    var cmdName = params ? params[0] : '';
    data = {
      result : mockResult,// aca va la posible opcion de resultado del comando
      cmdName : cmdName,
      units : ''
    };
    setTimeout(successCallback(data), 1000);
    break;
   case 'runStandardCommand':
    var cmdList = params[0].split('::');
    data = {
      result : '032501',// aca va la posible opcion de resultado del comando
      requestCmdId : cmdList[0],
      cmdName : cmdList[1],
      cmdDesc : cmdList[2],
     // posibles estados (usado para comandos solicitados por el experto)
      state : 'EJECUTADO'
    };
    setTimeout(successCallback(data), 1000);
    break;

   ...
   default:
    if (successCallback) {
     setTimeout(successCallback(), 1000);
    }
    break;
  }
 } else {// codigo invocado cuando se ejecuta en el dispositivo
  cordova.exec(successCallback, errorCallback, pluginName, method, params);
 }
};
```

v. Iniciar el servidor de pruebas del proyecto.
El servidor de pruebas se ejecuta por medio de Grunt, tal como se observa a continuación:

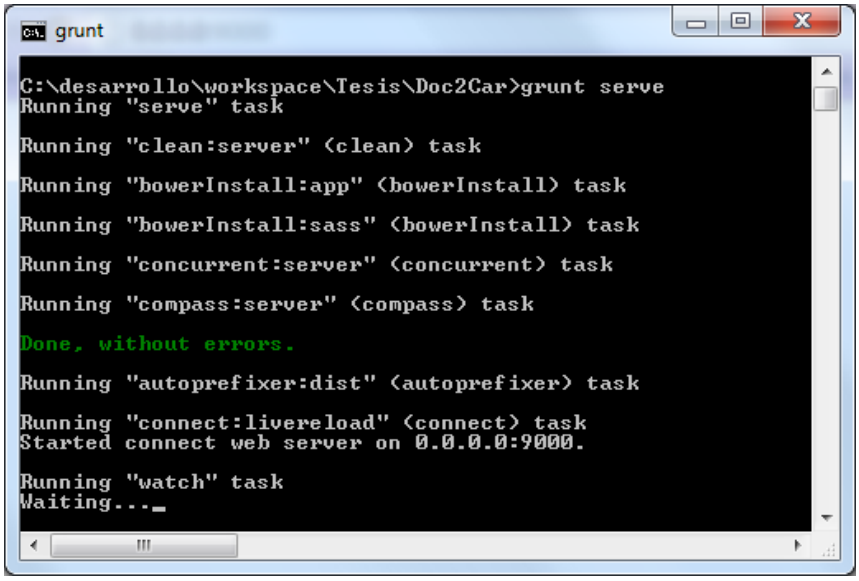

Figura 26: Servidor de pruebas

vi. Ejecutar las pruebas de comportamiento.
Para ello se debe arrancar el Webdriver y posteriormente ejecutar las pruebas con Protractor, tal como se muestra en las siguientes imágenes.

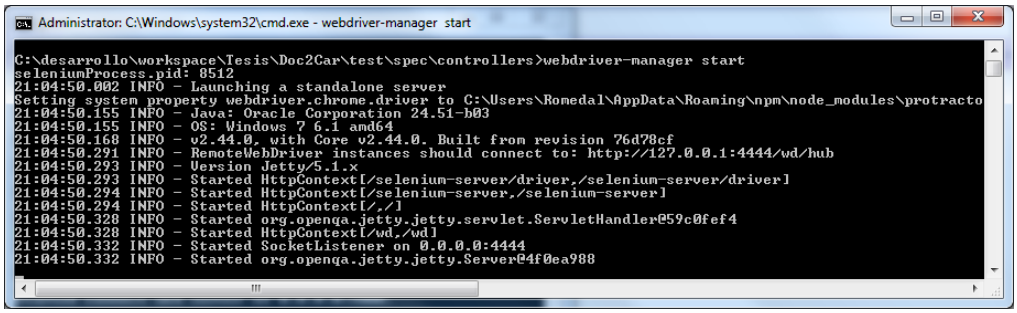

Figura 27: Arranque de Webdriver para ejecución de pruebas

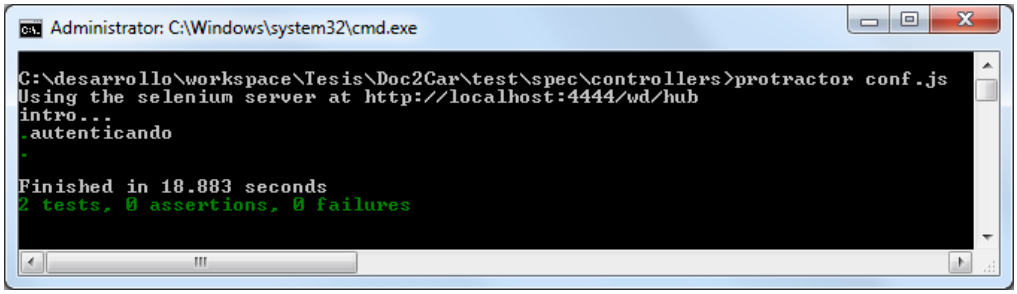

Figura 28: Ejecución de pruebas de comportamiento

vii. Si la prueba no es exitosa se retorna al paso iii.
viii. Luego de aprobada la prueba de comportamiento se reemplaza la lógica simulada por la lógica real de los servicios.
ix. Si la prueba no es exitosa se retorna al paso viii.
x. Si la prueba es exitosa se da por aprobado el comportamiento desarrollado y se procede a realizar siguiente modulo.

Como evidencia de este proceso se adjuntan como anexos, los archivos de pruebas escritos en Javascript. Estas pruebas se ejecutaban en automático cada vez que se generaban las versiones de producción de los proyectos.

Sin embargo, con éste modelo de pruebas y desarrollo no se pudo cubrir toda la funcionalidad contemplada, en especial en lo relacionado al tema de las notificaciones ya que para comprobar y validar detalladamente su funcionamiento fue necesario realizar las pruebas sobre un dispositivo móvil real con capacidad de recibir dicho tipo de contenido desde el Portal Web.

Así pues, se ha creado un video que comprueba el funcionamiento completo del sistema en un ambiente real.

V CONCLUSIONES Y TRABAJOS FUTUROS

1. Conclusiones

Al crear la solución **doc2car** se espera haber logrado reducir la brecha que existe entre conductores y usuarios expertos, dado que mediante el sistema desarrollado ahora es posible centralizar la información de las fallas detectadas en los diferentes vehículos, para que personal experto en el análisis e interpretación de dicha información pueda emitir conceptos oportunos y adecuados tan pronto como la falla es reportada.

Esto permitirá que cada vehículo monitorizado tenga menos posibilidades de circular con desperfectos no detectados, logrando así en muchos casos el reducir las emisiones contaminantes causadas por vehículos defectuosos.

Adicionalmente, está claro que un conductor que se entere y corrija oportunamente un desperfecto en su vehículo, evitará gastos innecesarios en los cuales muy seguramente se vería inmerso si la falla no es detectada, informada y analizada a tiempo y el vehículo sigue circulando y agravando su avería.

Todo esto se logró gracias a la integración de diversas tecnologías que permitieron crear un producto enriquecido en contenido y funcionalidad, el cual podrá ser extendido a diferentes plataformas móviles con relativamente poco esfuerzo, ya que la arquitectura sobre la cual se cimienta así lo permite.

Por otro lado se puede decir que gracias a las tecnologías usadas y especialmente a las relacionadas con HTML5, CSS3, Javascript y Nodejs fue posible realizar el desarrollo y la validación de la solución de una forma muy amigable, práctica y automatizada, lo cual hubiese sido más complejo en el apartado móvil de haberse realizado el desarrollo en lenguaje nativo. Todo esto gracias a la posibilidad de realizar casi todas las pruebas bajo un navegador Web de escritorio.

2. Trabajo futuro

La combinación creada entre sistemas los OBDII y las tecnologías de comunicación y de servicios en la nube abre grandes posibilidades de trabajos futuros. Entre las identificadas se encuentran las siguientes:

- Dotar de mayor funcionalidad al Portal Web contemplando las historias de usuario asociadas a los demás tipos de usuarios identificados.
- Enriquecer el sistema de comunicación entre el usuario experto y el conductor para que este sea de dos vías y así el conductor pueda tener contacto con el usuario experto.
- Dar soporte a nueva funcionalidad en el plugin OBDII para así obtener mayor información del vehículo.
- Extender la funcionalidad del plugin OBDII a otros sistemas operativos móviles como iOS yWindoes Phone.

- Usar el sistema de notificaciones (Push) desarrollado para hacerle llegar diversa información a los conductores, tales como recomendaciones mecánicas, recordatorios de revisiones preventivas, publicidad, ofertas y demás.
- Darle la opción al conductor de poder ejecutar comandos personalizados en la aplicación móvil.
- Dotar de mayor funcionalidad al módulo de navegación para así poder grabar los recorridos realizados y combinarlos con la información obtenida del sistema OBDII para de esta forma generar datos estadísticos tales como: consumo de combustible vs distancia recorrida, distancia recorrida vs tiempo, velocidad vs consumo de combustible, etc.

  Crear una base de conocimiento que permita ir asociando fallas a posibles causas y soluciones propuestas para en un futuro automatizar en lo posible el proceso de análisis y emisión de conceptos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. "On-Board Diagnostics (OBD)" [Online]. Available: http://www.epa.gov/obd/.

[2]. "On-Board Diagnostics (OBD) Program" [Online]. Available: http://www.arb.ca.gov/msprog/obdprog/obdprog.htm.

[3]. "Torque (OBD2 & Car)" [Online]. Available: https://play.google.com/store/apps/details?id=org.prowl.torquefree.

[4]. "DashCommand (OBD ELM Scanner)" [Online] Available: https://play.google.com/store/apps/details?id=com.palmerperformance.DashCommand&hl=en.

[5]. "Local and Remote Notification Programming Guide" [Online] Available: https://developer.apple.com/library/IOS/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html.

[6]. Consejo nacional de investigación de las academias Nacionales, "Cambio climático: Evidencia, impacto y opciones". 2013. [Online]. Available: http://nas-sites.org/americasclimatechoices/files/2013/04/136909453-Cambio-Climatico-Evidencia-Impactos-y-Opciones.pdf. pp. 2. [Accessed: 05-Jul-2013].

[7]. ETAP, "The carbon trust helps UK businesses reduce their environmental impact", Junio 2007. [Online]. Available: http://www.docstoc.com/docs/33410320/The-Carbon-Trust-helps-UK-Businesses-reduce-their-environmental-Impact. [Accessed: 15-Jul-2013].

[8]. ANDI, "Documento caracterización industria de vehículos" [Online]. Available: http://www.andi.com.co/downloadfile.aspx?Id=bf24d519-86ef-4fa0-9933-fcd150738d04, [Accessed: 07-Jul-2013].

[9]. "On-Board Diagnostic II (OBD II) Systems - Fact Sheet / FAQs" [Online]. Available: http://www.arb.ca.gov/msprog/obdprog/obdfaq.htm, [Accessed: 02- Jul-2013].

[10]. "OBDLink LX Bluetooth Scan Tool" http://www.scantool.net/scan-tools/smart-phone/obdlink-lx.html

[11]. "OBDPro Smart Switch" http://www.obdpros.com/product_info.php?products_id=137

[12]. Abrahamsson, P., Keynote: Mobile software development – the business opportunity of today, en Proc. of the International Confrence of Software Development, 2005.

[13]. VTT,"Mobile-D: An Agile Approach for Mobile Application Development" [Online]. Available: http://agile.vtt.fi/mobiled.html. [Accessed: 17-Ago-2013]

[14]. Beck, K., Extreme Programming Explained: EmbraceChange, Addison-Wesley, 2000.

[15]. Cockburn, A., Crystal Clear, A Human-PoweredMethodology for Small Teams, Addison-Wesley, 2004.

[16]. Kruchten, P., The Rational Unified Process: An Introduction, Addison-Wesley Professional, 1999.

[17]. Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003

[18]. "Overview of OBD and Regulations" [Online]. Available: http://www.autoshop101.com/forms/h46.pdf

[19]. "Understanding On-Board Diagnostics (OBD)" [Online]. Available: http://www.epa.state.il.us/air/vim/faq/obd.html

[20]. "Introduction to OBDII" [Online] Available: http://www.lbcc.edu/attc/documents/OBD2.pdf

[21]. “ELM327 OBD to RS232 Interpreter” [Online] Available: http://elmelectronics.com/DSheets/ELM327DS.pdf

[22]. “Presentation of ELM interfaces” [Online]. Available: http://www.outilsobdfacile.com/diagnostic-interface-elm-327.php

[23]. “Portal de aplicaciones de Windows Phone – Use Freely ELM327!” [Online]. Available: http://www.windowsphone.com/en-us/store/app/use-freely-elm327/9eef601d-e6c5-44b8-a0e4-a954e807dce2

[24]. “Industry Leaders Announce Open Platform for Mobile Devices” [Online] Available: http://www.openhandsetalliance.com/press_110507.html

[25]. “Android version history” [Online]. Available: http://en.wikipedia.org/wiki/Android_version_history

[26]. “iOS Technology Overview” [Online]. Available: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf

[27]. “Android SDK”. [Online]. Available: https://developer.android.com/sdk/index.html?hl=i

[28]. “Android SDK” [Online]. Available: http://developer.android.com/sdk/exploring.html

[29]. “SDK Compatibility Guide”. [Online]. Available: https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/cross_development/cross_development.pdf

[30]. “Página Web oficial de jQuery Mobile” [Online]. Available: http://jquerymobile.com/

[31]. “Sitio Web para la creación de temas personalizados” [Online]. Available: http://themeroller.jquerymobile.com/

[32]. "Licencia MIT" [Online]. Available: http://en.wikipedia.org/wiki/MIT_License

[33]. “Página oficial de Sencha Touch” [Online]. Available: http://www.sencha.com/products/touch/

[34]. “Licenciamiento de Sencha Touch” [Online]. Available: http://www.sencha.com/products/touch/license/

[35]. “Licencia GNU GPL” [Online]. Available: http://www.gnu.org/copyleft/gpl.html

[36]. “Sitio oficial de Phone JS” [Online]. Available: http://phonejs.devexpress.com/

[37]. “Definición de SPA” [Online]. Available: http://en.wikipedia.org/wiki/Single-page_application

[38]. “Licencia Phone JS” [Online]. Available: http://phonejs.devexpress.com/EULA

[39]. “Página oficial de Kendo UI” [Online]. Available: http://www.telerik.com/kendo-ui

[40]. “Definición del enfoque MV-VM” [Online]. Available: http://en.wikipedia.org/wiki/Model_View_ViewModel

[41]. “Licencia Kendo UI” [Online]. Available: http://www.telerik.com/kendo-ui#both-commercial-and-open-source-licensing

[42]. “Versión reducida de Kendo UI” [Online]. Available: http://www.telerik.com/kendo-ui/open-source-core

[43]. “Sitio oficial de Angularjs” [Online]. Available: https://angularjs.org/

[44]. “Patron MVC” [Online]. Available: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[45]. “Definición de DOM” [Online]. Available: http://en.wikipedia.org/wiki/Document_Object_Model

[46]. “Sitio Oficial de Ionic” [Online]. Available: http://ionicframework.com/

[47]. “Sitio oficial de Apache Cordova” [Online]. Available: https://cordova.apache.org/

[48]. “Plugins creados para la plataforma Apache Cordova” [Online]. Available: http://plugins.cordova.io/

[49]. “Sitio oficial de Zamarin” [Online]. Available: https://xamarin.com/

[50]. “Sitio oficial de Appcelerator Titanium” [Online]. Available: http://www.appcelerator.com/titanium/

[51]. “Sitio oficial de Amazon Web Services: Acerca de AWS” [Online]. Available: http://aws.amazon.com/es/what-is-aws/

[52]. “Capa de uso gratuito AWS” [Online]. Available: https://aws.amazon.com/es/free/

[53]. “Sitio oficial de Microsoft Azure” [Online]. Available: http://www.azure.microsoft.com/es-es/

[54]. “Sitio Oficial de Google Cloud Platform” [Online]. Available: https://cloud.google.com/

[55]. “Sitio oficial de Google App Engine” [Online]. Available: https://cloud.google.com/products/app-engine/

[56]. “Solución Móvil de Google App Engine” [Online]. Available: https://cloud.google.com/solutions/mobile/

[57]. “Sitio oficial de Parse” [Online]. Available: https://www.parse.com/

[58]. “Librería para leer datos OBDII en Android” [Online]. Available: https://github.com/pires/android-obd-reader

[59]. “Parse x Angular JS Boilerplate by BRANDiD” [Online]. Available: http://blog.parse.com/2013/10/17/guest-post-parse-x-angular-js-boilerplate/

[60]. “Your Ultimate Guide to the Non-Relational Universe” [Online]. Available: http://nosql-database.org/

[61]. “edmunds developer network” [Online]. Available: http://developer.edmunds.com/index.html

[62]. “Información sobre los códigos de avería” [Online]. Available: http://www.tallerestoande.org/c%C3%B3digos-aver%C3%ADas-obd-ii/

[63]. “BehaviorDrivenDevelopment” [Online]. Available: http://behaviour-driven.org/

[64]. “Jasmine” [Online]. Available: http://jasmine.github.io/1.3/introduction.html

[65]. “SeleniumHQ” [Online]. Available: http://www.seleniumhq.org/

[66]. “Polyak, N. (2012). MVVM Pattern Made Simple” [Online]. Available: CodeProject. http://www.codeproject.com/Articles/278901/MVVM-Pattern-Made-Simple

LISTA DE ANEXOS