ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

RECOLECCIÓN DE DATOS DEL SISTEMA OBD II DE UN AUTOMÓVIL USANDO UN DISPOSITIVO ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

ALONSO RODRIGO ANCHAPAXI SOCASI

alonso.anchapaxi@gmail.com

DIRECTOR: Ing. JULIO CÉSAR CAIZA ÑACATO, MSc.

julio.caiza@epn.edu.ec

Quito, agosto 2016

DECLARACIÓN

Yo, Alonso Rodrigo Anchapaxi Socasi, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Alonso Rodrigo Anchapaxi Socasi

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Alonso Rodrigo Anchapaxi Socasi, bajo mi supervisión.

Ing. Julio César Caiza, MSc.

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

En primer lugar agradezco a Dios quien me ha dado la salud y fortaleza para superar todas las adversidades.

A mis padres Blanca y Rodrigo que con su sabiduría supieron guiarme por el camino del bien, por todos los sacrificios que hicieron para que nunca me falte nada, por la confianza, amor y apoyo durante la carrera y sobre todo por haberme dado la mejor herencia: los estudios.

A mis hermanos Byron y Mercedes que con su manera de ser me muestran que existe otra forma de ver la vida y afrontar los problemas.

A mi director de proyecto de titulación Ing. Julio César Caiza que durante todo el desarrollo del proyecto supo guiarme, aconsejarme y alentarme para la finalización del mismo.

A Luis Salazar y Fabio Fernandes por su comprensión y las facilidades de tiempo brindadas para la culminación de este proyecto.

410N50

DEDICATORIA

A los seres que más amo, mi esposa Erika Johanna, mi hijo el mayor Alonso Alessandro y mi hijo el menor Demian Emiliano. Son la razón de mi vida, el tesoro más grande que Dios me regaló y el motivo de mí existir.

410N50

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

PRESENTACIÓN

A partir del año 1996 todos los automóviles tienen integrado un sistema de diagnóstico electrónico llamado OBD II (On Board Diagnostic Second Generation). Este sistema es el encargado de informar y almacenar los errores detectados por el resto de unidades de control que tiene un automóvil y facilitar el acceso a esta información a través de la interfaz DLC (Data Link Connector). Haciendo uso de esta interfaz, el mercado automotriz ha desarrollado varios escáner portátiles y de alto costo que permiten monitorear el funcionamiento de un automóvil. Otra alternativa es el uso de lectores de códigos OBD que usan el circuito integrado ELM327. Mediante el uso de estos, se han desarrollado varios proyectos que permiten al usuario realizar un diagnóstico de los diferentes sistemas del vehículo mediante el uso de un computador portátil [1]. Otros proyectos ayudan a realizar un monitoreo remoto de las medidas del sistema OBD II de los vehículos [2]–[4]. En este caso se provee de un mecanismo de configuración dinámico y remoto de los parámetros a leer. Asimismo, se provee de reportes estadísticos y por fechas de los diferentes parámetros, la visualización en tiempo real del vehículo, entre otras características.

Los trabajos previos [2]–[4] plantean tres dificultades que son: elevado costo del subsistema ubicado en el automóvil, la ejecución inmediata del proceso de lectura de datos desde el lector ELM327 en cuanto se recibe el mensaje de configuración y el almacenamiento de los datos de respuesta en zonas en donde no exista cobertura GPRS. Debido al elevado costo del subsistema ubicado en el automóvil de los trabajos previos [2]–[4] se plantea el reemplazo del subsistema mediante el re-uso de Smartphones Android que cumplan con todas las funcionalidades.

A continuación se detalla el desarrollo del subsistema que permite resolver las dificultades encontradas y el reemplazo del subsistema ubicado en el automóvil.

RESUMEN

El presente documento ha sido dividido en 6 capítulos.

En el capítulo 1 se presenta un estudio del sistema OBD II como fuente de información y almacenamiento de errores detectados por las unidades de control que tiene un automóvil. Adicionalmente se detallan los dispositivos que permiten acceder a la información almacenada en el sistema OBD II.

Para finalizar, este capítulo incluye el estudio de las metodologías de desarrollo ágil para sistemas móviles, permitiendo identificar las mejores prácticas de las metodologías de desarrollo ágil para sistemas móviles.

En el capítulo 2 se presenta un estudio y análisis de trabajos previos [2]–[4] permitiendo identificar los requerimientos del presente proyecto. Además se estudian los diferentes subsistemas con la finalidad de enmarcar los puntos en los cuales existen deficiencias. Finalmente se detallan teléfonos móviles de bajo costo que son usados en el subsistema automóvil.

En el capítulo 3 se diseña e implementa el subsistema automóvil. Se plantea la solución al algoritmo que recibe el mensaje de configuración vía SMS desde el subsistema servidor. Adicionalmente se describe el mecanismo que permite almacenar en un archivo los datos temporalmente en las zonas de no cobertura GPRS. Finalmente se implementan las mejoras definidas.

En el capítulo 4 se reportan un conjunto de pruebas que permitan verificar la funcionalidad de todo el sistema. El conjunto de pruebas permite: verificar el estado de las configuraciones antes de establecer conexión Bluetooth con el Lector ELM327, verificar la recepción del mensaje de configuración vía SMS y ejecución inmediata del proceso en base a nueva configuración y finalmente se verifica el almacenamiento de datos en zonas de no cobertura GPRS.

En el capítulo 5 se detallan los costos directos e indirectos de implementación de la solución y el ahorro obtenido en relación al hardware del subsistema automóvil trabajo previo [4].

En el capítulo 6 se presentan las conclusiones y las recomendaciones conforme a las experiencias y conocimientos adquiridos en el desarrollo del proyecto.

CAPÍTULO 1

1 FUNDAMENTOS TEÓRICOS

1.1 ESTUDIO DEL SISTEMA OBD II [5]

En la década de los 70 y principios de los 80 varios fabricantes de automóviles empezaron a integrar componentes electrónicos que permitían tener un control y diagnóstico de los automóviles, esto conllevó a que cada fabricante incorporase su propio conector y código de error para identificar una falla del automóvil. A partir del año 1996 la industria automotriz integra en todos sus automóviles un sistema de diagnóstico electrónico llamado OBD II (On Board Diagnostic Second Generation). Este sistema es el encargado de informar y almacenar los errores detectados por las unidades de control que tienen un automóvil, adicionalmente permite el acceso a dicha información a través de la interfaz DLC (Data Link Connector) y de un único conector. En la Figura 1.1 se muestra la interfaz DLC y su respectivo conector.

Figura 1.1 Interfaz DLC y conector

Como se puede observar en la anterior figura, la interfaz DLC cuenta con 16 pines que cumplen funciones específicas. En la Figura 1.2 se muestra la numeración de los pines, mientras que en la Tabla 1.1 se puede observar la función de cada pin.

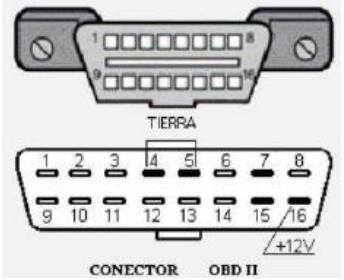

Figura 1.2 Numeración de pines de interfaz DLC [6]

PIN	FUNCIÓN
1	Uso del fabricante
2	Bus (+) J1850 VPM (Variable Pulse Modulation) y PWM (Pulse Width Modulation)
3	Uso del fabricante
4	Tierra (chasis)
5	Señal de tierra
6	Bus de datos CAN alto (J-2284)
7	Línea K ISO 9141-2
8	Uso del fabricante
9	Uso del fabricante
10	Bus (-) J1850
11	Uso del fabricante
12	Uso del fabricante
13	Uso del fabricante
14	Bus de datos CAN (Controller Area Network) bajo (J-2284)
15	Línea L ISO 9141-2
16	Voltaje de batería

Tabla 1.1 Funciones de pines de interfaz DLC [5]

1.1.1 COMPONENTES DEL SISTEMA OBD II

El sistema OBD II tiene varios componentes que son: ECU (Engine Control Unit) conocida comúnmente como la computadora del automóvil, los sensores cuya función es enviar los datos hacia la ECU, la luz indicadora de fallas MIL (Malfunction Indicator Ligth) ubicada en el tablero, y la interfaz DLC. En la Figura 1.3 se muestran los componentes del sistema OBD II.

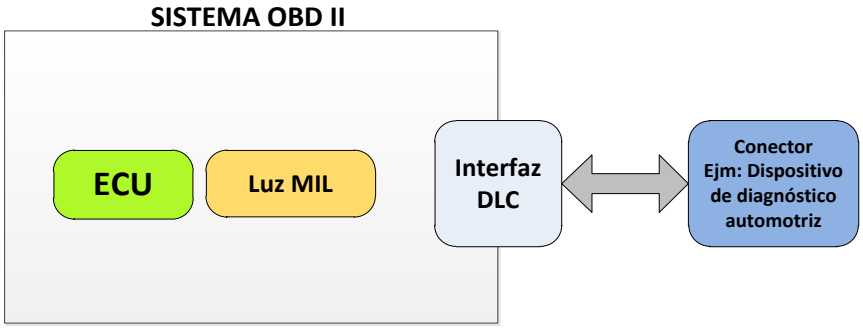

Figura 1.3 Componentes del sistema OBD II [2]

1.1.2 PROTOCOLOS DE COMUNICACIÓN OBD II

Debido a que cada fabricante de automóviles tiene su propia ECU, el sistema OBD II utiliza varios tipos de protocolos. Donde cada uno tiene su propia velocidad de comunicación y niveles de voltaje. En la Tabla 1.2 se presenta un cuadro comparativo de los 5 protocolos de comunicación existentes.

PROTOCOLO	FABRICANTE	COMUNICACIÓN	VELOCIDAD TRANSMISIÓN	VOLTAJE
SAE J1850 PWM	Ford, Lincoln y Mercury	Modulación de ancho de pulso (PWM)	41.6 Kbaud/s	0 – 5 V en modo diferencial
SAE J1850 VPN	General Motors	Modulación de ancho de pulso variable (VPN)	10.4 – 41.6 Kbaud/s	2.2 V – 0L 8 V – 1L
ISO 9141-2	Fabricantes europeos, asiáticos, Chrysler, Jeep y Dodge	Comunicación similar al estándar RS-232	10.4 Kbaud/s	0 – 12 V (se ajustan al voltaje de la batería)
ISO 14230 KWP (Key Word Protocol)	Fabricantes europeos y asiáticos	Comunicación similar al estándar RS-232	1.2 – 10.4 Kbaud/s	0 – 12 V (se ajustan al voltaje de la batería)
ISO 15765 CAN	Compañía Bosch	Red de Área del controlador	250 – 500 Kbps	2.5 – 5 V (CANH) 2.5 – 0 v (CANL)

Tabla 1.2 Comparación de protocolos de comunicación OBD II [2]

Adicionalmente, en la Figura 1.4 se muestran los pines de la interfaz DLC que usa cada protocolo para su comunicación así como los pines de tierra y de voltaje (Vcc).

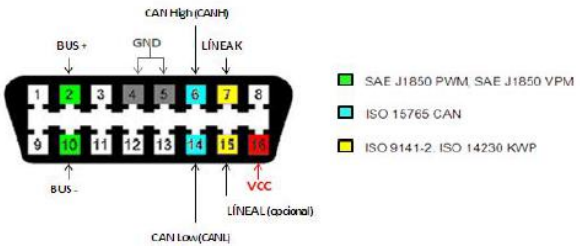

Figura 1.4 Pines de comunicación de interfaz DLC usado por cada protocolo [7]

1.1.3 MODOS DE MEDICIÓN [1]

El sistema OBD II utiliza nueve modos de medición y dependiendo del modo seleccionado se accede a cierto tipo de información de la ECU del automóvil.

Cada modo utiliza los denominados PIDs (Parameter Identification), que son códigos que piden información al automóvil y fueron creados exclusivamente para la comunicación con el escáner automotriz.

A continuación se describen los nueve modos del sistema OBD II:

- Modo 1. Obtención de datos actualizados

Permite acceder a las medidas de los sensores en tiempo real, algunos valores de este modo pueden ser: temperatura, voltajes, presión, potencia, RPM del motor, entre otros dependiendo del sensor que se esté evaluando.

- Modo 2. Obtención de datos almacenados

  Este modo permite acceder a la información almacenada en la memoria de la ECU y que tiene relación con algún fallo que ocurrió en el automóvil. Los valores son tomados en el momento exacto de ocurrir la primera falla en el sensor, si existe más de una falla en el mismo sensor, estos valores no serán almacenados.

- Modo 3. Obtención de códigos de falla

  Este modo no necesita ningún PID ya que la información extraída corresponde a los códigos de falla DTC (Data Trouble Codes) de la ECU. Esta información corresponde a la lectura de un fallo que se produjo debido a que el valor del sensor estuvo fuera del rango óptimo.

  En la Figura 1.5 se puede observar el formato de los códigos de falla correspondiente al modo 3.

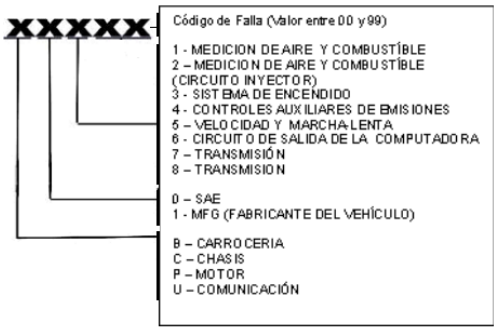

Figura 1.5 Formato de códigos de falla DTC [8]

- Modo 4. Borrado de códigos de falla y valores almacenados

  El modo 4 permite borrar todos los códigos de falla DTC almacenados en la ECU del automóvil y los valores guardados del modo 2. Cuando se borran los códigos de la memoria de la ECU queda sin información y la luz MIL se apaga; si no se corrige adecuadamente el problema, los códigos de falla volverán a aparecer.

- Modo 5. Resultado de las pruebas de los sensores de oxígeno

  Este modo permite acceder a los resultados de las pruebas realizadas al sensor de oxígeno para de esta forma evaluar el funcionamiento del

mismo. El sensor de oxígeno es uno de los más importantes ya que interviene en gran medida en la calidad del aire.

- Modo 6. Resultado de pruebas de otros sensores

  El modo 6 almacena los resultados de las pruebas realizadas por la ECU al sistema de monitoreo no continuo. Ejemplos de sistemas no continuos son: catalizador, sistema secundario de aire, refrigerante de aire acondicionado.

  A continuación respuestas de este tipo de sistemas:

  - "N" Monitoreo no soportado
  - "C" Monitoreo completo
  - "I Todavía no termina el monitoreo de esta prueba.
  - "?" Incongruencia del sistema.Modo 7. Códigos de falla pendientes.

- Modo 7. Códigos de falla pendientes

  El modo 7 muestra todos los códigos de falla pendientes que no hayan sido reparados. Este modo es usado por los técnicos automotrices posterior a la reparación del vehículo ya que les permite ver los resultados después de borrar la información de diagnóstico y luego de un ciclo de conducción, de este modo validan si el problema fue corregido.

- Modo 8. Control de funcionamiento de componentes

  En este modo se puede realizar pruebas en actuadores (bombas de combustible, válvula de ralentí) activándolos o desactivándolos. Si no se cuenta con la experiencia necesaria para hacer estas modificaciones, el uso de este modo no es recomendable.

- Modo 9. Información del automóvil

  El modo 9 permite el acceso a la información del automóvil VIN (Vehicle Identification Number). El VIN es u número único que va a identificar al automóvil en cualquier parte del mundo. La estructura del VIN está conformada por 17 caracteres alfanuméricos (no incluyen los caracteres I, O ni Q) y cada uno tiene un significado. En la Tabla 1.4 se puede observar el significado de cada carácter del VIN.

CARACTER	DESCRIPCIÓN
1	- País de fabricación
2	- Fabricante
3	- Tipo de automóvil o división de fábrica

Tabla 1.3 Estructura general del VIN [5]

CARACTER	DESCRIPCIÓN
4 al 8	- Características del automóvil
9	- Dígito de control
10	- Año del modelo
11	- Planta de ensamblaje
12 al 17	- Secuencia de producción

Tabla 1.4 Estructura general del VIN [5] (continuación)

1.2 DISPOSITIVOS DE ACCESO A INFORMACIÓN ALMACENADA EN EL SITEMA OBD II [9]

Como se mencionó anteriormente, la interfaz estándar que maneja el sistema OBD II es la interfaz DLC, a esta interfaz se puede conectar varios tipos de dispositivos que acceden a la información de este sistema.

1.2.1 ESCÁNER DE DIAGNÓSTICO PROFESIONAL

Este tipo de dispositivos son usados por profesionales en los talleres automotrices, son dispositivos de diagnóstico especializado e incluso específicos de cada fabricante que habitualmente tiene un costo entre USD1499 a USD1799 [10]–[12]. En la Figura 1.6 se muestra un dispositivo de este tipo.

Figura 1.6 Escáner de diagnóstico profesional

1.2.2 ESCÁNER DE DIAGNÓTICO CON CONECTOR USB O PUERTO SERIE

En los últimos años han surgido varios conectores que permiten acceder a la información del sistema OBD II y que son capaces de transmitirla a un ordenador a través del puerto USB o puerto serie para ser procesada, este tipo de conectores requiere de un software especializado que comparado con el punto anterior no son costosos y que oscilan en un rango de USD 29,95 a USD 99,95 [13], [14].

En la Figura 1.7 se muestra un escáner de diagnóstico con puerto USB y su respectivo software.

Figura 1.7 Escáner de diagnóstico con puerto USB

1.2.3 LECTOR DE CÓDIGOS OBD II CON ACCESO BLUETOOTH O WIFI

Actualmente podemos encontrar lectores de códigos OBD II que permiten establecer conexión al sistema OBD II vía Bluetooth o WiFi. Este tipo de lectores han revolucionado el mercado ya que su precio oscila entre USD 25 a USD 38,99 [15] y existen aplicaciones gratuitas que pueden ser instaladas en dispositivos móviles como es el caso de Smartphones y Tablets. En la Figura 1.8 se puede observar un lector Bluetooth y otro WiFi.

Figura 1.8 Lector OBD II Bluetooth y WiFi.

En la Tabla 1.5, Tabla 1.6 y Tabla 1.7 muestra un resumen de los diferentes tipos de lectores OBD II USB, Bluetooth y WiFi.

Descripción	Procesador	Protocolos OBD2	Velocidad Trans misión	Voltaje de opera ción	Corriente Nominal	PC	iPhone iPad o iPod	An droid	Symbian	Blue tooth	Wi Fi
ELM327 CANBUS With aluminum shell - No. SC01	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2 J1850 VPW J1850 PWM	9600 o 38400 Baudios	12V	45mA	X					
ELM327 Vgate - No. SC76	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2		12V				X	X	X	

Tabla 1.5 Tipos de lectores OBD II USB, Bluetooth y WiFi

Descripción	Procesador	Protocolos OBD2	Velocidad Transmisión	Voltaje de operación	Corriente Nominal	PC	iPhone iPad o iPod	Android	Symbian	Bluetooth	Wi Fi
ELM327 Bluetooth with Plastic shell - No. SC03	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2				X				X	
ELM327 Plastic with FT232RL Chip No. SC02	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2	9600 o 38400 Baudios	12V	45mA	X					
ELM327 WIFI for iPhone iPad iPod No. SC133	ELM327	ISO 9141-2 ISO 11898(CAN) - usado por Bosh ISO 14230(KWP2000) ISO 15765(CAN)					X			X	X
WiFi OBD2 scanner for Apple iPad iPhone iPod Touch No. SO71							X				X
MINI ELM327 Bluetooth No. SC104	ELM327	ISO14230 (KWP2000) ISO 9141-2				X				X	
ELM327 WIFI for Android iPhone/iPad No. SC133-C	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2	9600 o 38400 Baudios	12V	45mA		X	X			X
WiFi ELM327 Apple iPhone Touch No.SC133-B	ELM327	ISO 9141-2 ISO 14230-4 (KWP2000) ISO 15765-4 (CAN) SAE J1939 - vehículos pesados				X	X			X	X
mini ELM327 WiFi with Switch work with iPhone No. SC157	ELM327	ISO 9141-2 ISO 11898 (CAN) usado por Bosh ISO 14230 (KWP2000) ISO 15765(CAN) SAE J1939 - vehículos pesados					X				X
ELM327 Bluetooth CAN BUS No. SC03-B	ELM327	ISO15765-4 (CAN) ISO14230-4 (KWP2000) ISO9141-2				X				X	
USB ELM327 plastic with FT232RL Chip No. SC02-B	ELM327	ISO 15765-4 (CAN) SAE J1850 PWM SAE J1850 VPW ISO 9141-2 ISO 14230-4 (KPW2000)				X					

Tabla 1.6 Tipos de lectores OBD II USB, Bluetooth y WiFi (continuación)

Descripción	Procesador	Protocolos OBD2	Velocidad Transmisión	Voltaje de operación	Corriente Nominal	PC	iPhone iPad o iPod	Android	Symbian	Bluetooth	Wi Fi
Super Mini ELM327 Bluetooth with Power Switch No. SC158	ELM327	ISO9141-2 ISO14230-4 (KWP2000) ISO15765-4 (CAN) SAE J1939 - vehículos pesados						X		X	
VGATE WIFI OBD Multiscan Elm327 for Android PC iPhone iPad No. SC76-B	ELM327	ISO 15765-4 (CAN) ISO 14230-4 (KWP2000) ISO 9141-2 SAE J1850 VPW SAE J1850 PWM SAE J1939 - vehículos pesados	4800 / 9600 / 10400 / 38400 / 500K Baudios			X	X	X			
MINI ELM327 Bluetooth white No. SC104-B	ELM327	ISO 9141- 2 ISO14230 (KWP2000)				X				X	
MINI ELM327 Bluetooth black No. SC104-C	ELM327	ISO 9141- 2 ISO14230 (KWP2000)				X				X	
MINI ELM327 Interface Viecar 2.0 OBD2 Bluetooth for Android and windows No. SC264	ELM327	ISO 9141-2 ISO14230-4 (KWP2000) ISO15765-4 (CAN)		9 - 16V	25mA	X		X		X	
Elm327 plastic with 2102 chip No. SC02-D	ELM327	ISO 9141-2 ISO 14230-4 (KWP2000) ISO 15765-4 (CAN)				X					
WIFI ELM327 for Android and iOS iPhone/iPad No.SC76-D	ELM327	ISO 9141-2 ISO 11898 (CAN) - usado por Bosh ISO 14230-4 (KWP2000) ISO 15765-4 (CAN) SAE J1939 - vehículos pesados		9 - 16V	45mA	X	X	X			X
Super MINI ELM327 Bluetooth Black No.SC104-D	ELM327	ISO 9141- 2 ISO14230 (KWP2000)				X				X	
Mini ELM327 Bluetooth No.SC104-E	ELM327	ISO 9141- 2 ISO14230 (KWP2000)		12V	45mA	X		X	X	X	
Super Mini ELM327 Bluetooth OBD2 CANBUS scanner No. SC104-F	ELM327	ISO 9141- 2 ISO14230 (KWP2000) ISO 15765-4 (CAN)				X		X	X	X	
Super MINI ELM327 Bluetooth OBD2 White No. SC104-G	ELM327	ISO 9141- 2 ISO14230 (KWP2000) ISO 15765-4 (CAN)				X		X	X	X	

Tabla 1.7 Tipos de lectores OBD II USB, Bluetooth y WiFi (continuación)

1.3 METODOLOGÍA DE DESARROLLO ÁGIL PARA SISTEMAS MÓVILES

El término “ágil” aplicado al desarrollo de software nace en febrero del 2001 en una reunión que se llevó a cabo en Utah. En esta reunión participaron 17 expertos de la industria del software cuyo objetivo fue bosquejar los valores y principios que deberían permitir a los equipos de trabajo el desarrollo rápido de software y responder a los cambios que pueden aparecer a lo largo de un proyecto [16].

El desarrollo ágil de software se enfoca en las personas y los resultandos, pretendiendo evitar los extensos procesos burocráticos de las metodologías tradicionales. Adicionalmente el desarrollo se lo realiza en cortos períodos de tiempo permitiendo minimizar riesgos. A continuación se estudian dos metodologías agiles para el desarrollo de aplicaciones móviles.

1.3.1 MOBILE-D [17], [18]

Esta metodología nace como parte del proyecto finlandés ICAROS en el 2004 y es una mezcla de varias metodologías consolidadas: eXtreme Programming (XP), Crystal methodologies y Rational Unified Process (RUP). El objetivo de esta metodología es conseguir ciclos de desarrollo muy rápidos y en equipos pequeños [16]. En la Figura 1.9 se puede observar el ciclo completo del proyecto, mismo que se encuentra formado por cinco fases: exploración, inicialización, productización, estabilización y prueba del sistema. Cada fase tiene tres días de desarrollo distintos (excepto la fase de exploración): planificación, trabajo y liberación. En casos particulares, se añaden días para acciones adicionales.

Figura 1.9 Ciclo de desarrollo de Mobile-D [16]

Con la finalidad de tener un mejor entendimiento de la metodología Mobile-D, se hizo un análisis de la metodología Mobile-D de los trabajos [17], [18] y a continuación se describen cada una de las fases.

1.3.1.1 Fase Exploración

Se centra en el establecimiento del plan de proyecto y los conceptos básicos que permitirán entender al mismo. A continuación se describen los artefactos usados:

1.3.1.1.1 Tablas de requisitos de usuario

La Tabla 1.8 muestra la plantilla de la tabla y contiene los siguientes elementos:

- Identificador: código alfanumérico que identifica unívocamente al requisito. Existen varios tipos de requisitos de usuario y su sintaxis es: RU <tipo> siendo RU un requisito de usuario, <tipo> identifica el tipo de requisito del usuario, C un requisito de capacidad, R un requisito de restricción. <número> una cifra de dos dígitos que va aumentando en función de cada requisito.
- Título: texto que resume la funcionalidad del requisito.
- Descripción: Información clara y concisa que hace el usuario con sus términos para describir el requisito.
- Prioridad: nivel de preferencia de la implementación del requisito durante la fase de desarrollo de la aplicación. Los valores posibles son “Alta”, “Media” y “Baja”.
- Necesidad: grado en el que se puede prescindir o no de la implementación de un requisito.
- Fuente: procedencia de la cuál ha sido extraído el requisito.

Identificador: RU<tipo>-<número>	
Título:	
Descripción:	
Prioridad:	
Necesidad:	
Fuente:	

Tabla 1.8 Plantilla de tabla de requisitos de usuario

1.3.1.1.2 Diagrama de casos de uso[19]

Los casos de uso nacen de los requisitos de usuario y en esta etapa es la representación gráfica de los mismos, el objetivo es modelar y definir el entorno del sistema (actores) y su funcionalidad principal (casos de uso).

- Actor: es un rol que es llevado a cabo por una persona o cosa.
- Comunicación: permite representar la relación que existe entre el actor y el caso de uso. Las tipos de relaciones son: generalización, asociación, dependencia <<include>> y dependencia <<extends>>.

La Figura 1.10 es un ejemplo de un diagrama de caso de uso.

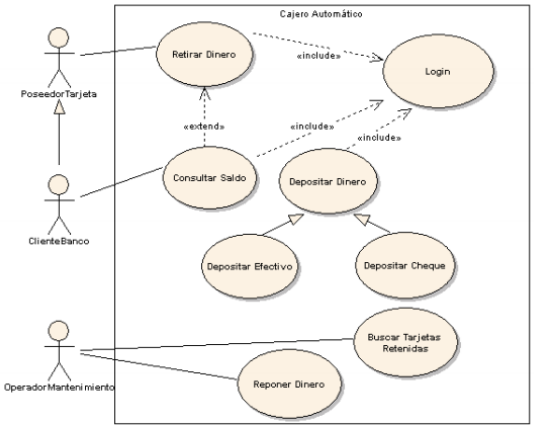

Figura 1.10 Ejemplo de diagrama de casos de uso [20]

1.3.1.1.3 Descripción de casos de uso

Una vez que se cuenta con la representación gráfica de los casos de uso, el siguiente paso es realizar una representación textual de los mismos en base a la plantilla de la Tabla 1.9.

Identificador: CU-<número>	
Nombre:	
Objetivo:	
Precondiciones:	
Postcondiciones:	
Escenario básico:	Escenario alternativo:

Tabla 1.9 Plantilla para representación textual de casos de uso

- Identificador: código alfanumérico que permite identificar de forma unívoca al caso de uso. La sintaxis es la siguiente: CU-<número> siendo CU el

identificador que hace referencia un caso de uso y <número> una cifra de dos dígitos que aumenta por cada caso de uso creado.

- Nombre: resume la funcionalidad requerida del caso de uso.
- Objetivo: breve descripción textual del caso de uso.
- Precondiciones: listado de todas las condiciones y acciones que deben cumplirse antes del iniciar el caso de uso.
- Postcondiciones: estado del sistema tras ejecutarse el caso de uso.
- Escenario básico: son los pasos que deben seguirse para ejecutar la funcionalidad del caso de uso.
- Escenario alternativo: opciones que tiene el escenario básico bajo determinadas circunstancias que pueden suceder en su ejecución.

1.3.1.1.4 Requisitos de software

Existen varios tipos de requisitos por lo que a continuación se mencionan cada uno de ellos.

- Requisito funcional (FU), define el “Qué” debe hacer la aplicación.
- Requisito de interfaz (IN), indica la interfaz que se usará para comunicarse con el usuario y otros sistemas.
- Requisito de operación (OP), define el “Cómo” va a realizar el sistema las tareas para las que fue creado.
- Requisito de recursos (RE), define los valores máximos de consumo de recursos por parte del sistema.
- Requisito de comprobación (CO), indica “Cómo” se debe verificar los datos de entrada y de salida.
- Requisito de seguridad (SE), métodos que tiene el sistema con la finalidad de garantizar integridad, confidencialidad y disponibilidad de datos.

En la Tabla 1.10 se muestra la plantilla para representar los requisitos de software.

Identificador: RU<tipo>-<número>	
Título:	
Descripción:	
Prioridad:	
Necesidad:	
Fuente:	

Tabla 1.10 Plantilla de requisitos de software

- Identificador: código alfanumérico que identifica unívocamente al requisito. La sintaxis es: RS<tipo>-<número>, en donde <tipo> corresponde a un requisito de software detallado anteriormente y <número> a una cifra de dos dígitos que aumenta conforme aparezcan más requisitos de software.
- Título: texto que resume la funcionalidad del requisito de software.
- Descripción: descripción breve, clara y concisa del requisito de software.
- Prioridad: nivel de preferencia de la implementación del requisito de software durante la fase de desarrollo de la aplicación. Los valores pueden ser “Alta”, “Medio” o “Baja“.
- Necesidad: grado en que se puede ´prescindir o no de la implementación de un requisito de software.
- Fuente: identifica la persona o documento del cuál ha sido extraído el requisito.

1.3.1.1.5 Matriz de trazabilidad de requisitos de usuario

Esta matriz permite realizar un seguimiento a los todos los requisitos de usuario a lo largo del ciclo de vida del proyecto con la finalidad de asegurar el cumplimiento de los mismos.

Esta matriz es usada en proyectos grandes y permite al líder del proyecto identificar qué requisitos se ven afectados ante un cambio propuesto por el cliente.

En la Tabla 1.11 se puede observar un ejemplo de matriz de trazabilidad de requisitos de usuario a casos de uso.

	RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08
RSFU-01	X							
RSFU-02	X							
RSFU-03	X							
RSFU-04		X						
RSFU-05			X					
RSFU-06				X				
RSFU-07					X			
RSFU-08						X		
RSFU-09							X	
RSFU-10								X

Tabla 1.11 Matriz de trazabilidad requisitos de usuario a casos de uso

1.3.1.2 Fase Inicialización

En esta fase los desarrolladores preparan los planes para las siguientes fases y establecen el entorno técnico incluyendo el entrenamiento del equipo de desarrollo. A continuación se describen los artefactos usados.

1.3.1.2.1 Diagrama de actividades [21]

Muestran el comportamiento dinámico de la aplicación en sus diferentes estados. Un diagrama de actividad está asociado con la implementación de un caso de uso y cuyo propósito es enfocarse en los flujos manejados por el procesamiento interno. Al contrario, un caso de uso se centra en la interacción del actor y el sistema y no en el procesamiento interno del sistema durante la ejecución del caso de uso.

Un diagrama de actividades está compuesto por tres elementos: *acción*, representa la realización de un paso del flujo de ejecución; las acciones no deben tener transiciones internas, *decisión*, es usada para identificar las diferentes transiciones posibles que depende de un valor booleano y *partición*, es usada para organizar las responsabilidades de las acciones en base a características comunes o áreas funcionales.

1.3.1.2.2 Diagrama de clases

Describe la estructura de la aplicación mostrando las clases con sus respectivos métodos, atributos y visibilidad de los mismos, adicionalmente muestra las relaciones entre clases, que pueden ser asociativas, de herencia, de uso y de agregación. Adicionalmente, un diagrama de clases permite crear un diseño conceptual de la información que se manejará en el sistema a desarrollarse y los componentes que se encargarán del funcionamiento.

Un diagrama de clases está compuesto por: *atributos*, son las características que corresponden a un objeto. *Operaciones*, actividades que puede realizar el objeto. *Interfaz*, conjunto de operaciones o atributos que permite a un objeto comportarse de cierta forma. *Herencia*, es la reutilización de una clase padre ya definida en una clase hija, la clase hija puede tener sus propias operaciones.

1.3.1.3 Fase de Productización

En esta fase se genera el código fuente en base a los diagramas de la anterior fase.

En el *día de planeación*, se realiza un análisis detallado de todos los requerimientos de la fase de exploración, luego en el *día de trabajo* se implementan las funcionalidades requeridas y finalmente en *el día de liberación* se verifica el funcionamiento de todos los módulos de la aplicación mediante casos de prueba. Un caso de prueba se crea a partir de un caso de uso y es la descripción de las actividades que se van a ejecutar con el fin de validar la aplicación.

1.3.1.4 Fase de Estabilización

Esta fase toma como base las recomendaciones y cambios sugeridos por el usuario luego del día de liberación de la fase de productización.

En el *día de planeación* se verifica la viabilidad de las recomendaciones y cambios sugeridos por el usuario, en caso de ser viables se los implementa en el *día de trabajo,* luego se *recopila la información* con la finalidad de ajustar todos los diagramas y finalmente en el *día de liberación* se verifica el funcionamiento de la aplicación incluyendo los módulos en los que fueron implementados los cambios. Fase de pruebas y reparación del sistema

En esta fase se integran todos los módulos de la aplicación. El objetivo de esta fase es la disponibilidad de una versión estable y funcional del sistema.

1.3.2 MÉTODO ÁGIL HÍBRIDO PARA DESARROLLAR SOFTWARE EN DISPOSITIVOS MÓVILES [22]

Esta metodología está enfocada a equipos pequeños y altamente cohesionados. Permite tener una comunicación y documentación ágil, su construcción está basada en características, ciclos cortos de trabajo (un macro ciclo está formado por 1 mes = 30 días), entregas constantes, permanente participación del cliente, sesiones de retroalimentación y fomenta la colaboración entre los miembros del equipo (ganking).

1.3.2.1 Macro ciclo

Constituye un paquete de funcionalidades o características (módulos) del software final. El macro ciclo se divide en 4 micro ciclos de una semana cada uno (5 o 7 días). Dentro de cada micro ciclo, los días corresponden a distintas etapas de desarrollo, enfocadas en desarrollar las características de la semana.

En la Tabla 1.12 se muestra un resumen de actividades de un macro ciclo.

	DÍA 1	DÍA 2	DÍA 3	DÍA 4	DÍA 5	DÍA 6	DÍA 7	
S0	SEMANA DE PLANIFICACIÓN DEL PROYECTO Y DEL PRIMER MACROCICLO							
S1	AR	DP	P	P	P	P	P	T	T	E	FB	C	OBS	TRANS
S2	AR	DP	P	P	P	P	P	T	T	E	BS	C	OBS	TRANS
S3	AR	DP	P	P	P	P	P	T	T	E	FB	C	OBS	TRANS
S4	AR	DP	P	P	P	P	P	T	T	E	BS	C	OBS	TRANS
	SEMANA DE *FEEDBACK*, PLANIFICACIÓN DEL SIGUIENTE MACROCICLO							

Tabla 1.12 Resumen de actividades de un macro ciclo [22]

Dónde:

- S0 a S4: Semana 0 a semana 4.
- AR: Análisis de requerimientos.
- DP: Diseño del producto (entrega semanal: diseño del microciclo).
- P: Programación/Producción.
- T: Testing.
- E: Entrega del producto o característica semanal.
- FB: Sesión de feedback.
- BS: Jornada de brainstorming.
- C: Sesión de comunicación y distensión.
- I: Investigación.
- OBS: Sesión de revisión del avance, detección de obstáculos y dificultades.
- TRANS: Sesión de transición. Preparación de los detalles generales del siguiente micro ciclo.

1.3.2.2 Ciclo de vida de la metodología

El ciclo de vida de la metodología es cíclico y permite mantener un mejor control del avance e ir construyendo software de manera incremental.

En la Figura 1.11 se muestra el ciclo de vida de la metodología en estudio.

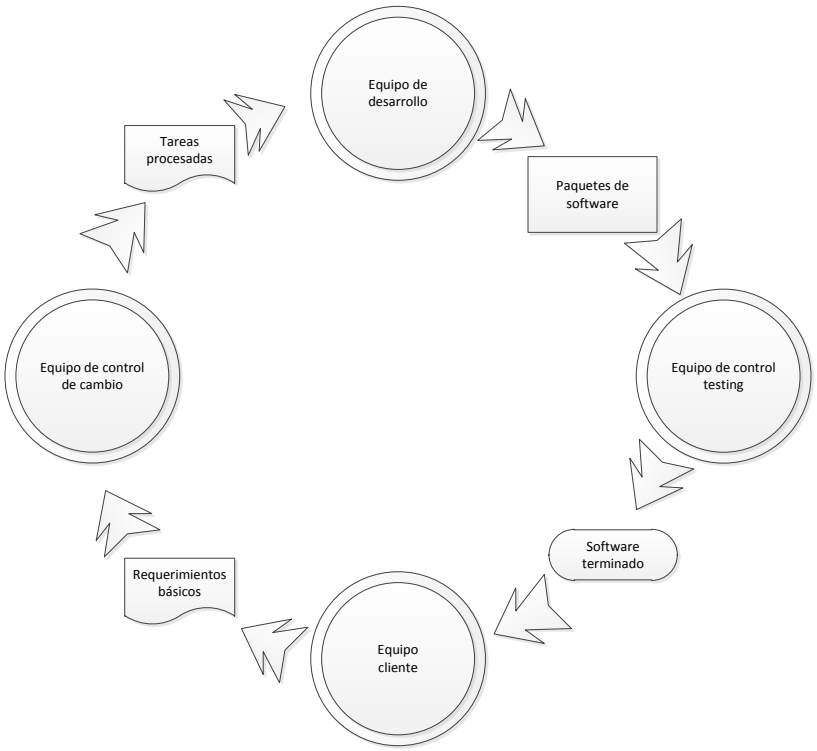

Figura 1.11 Ciclo de vida de metodología ágil híbrida [22]

1.3.2.3 Equipos/roles de la metodología

Como se puede observar en la Figura 1.11, la metodología plantea 4 equipos/roles que cumplen las siguientes funciones.

- Cliente - Equipo cliente: conformado por una persona, tiene la responsabilidad de mantener el contacto con el cliente y hacer suyos los requerimientos que el cliente solicita. Transformar las características funcionales y no funcionales del futuro software en requerimientos o características entendibles por el equipo control de cambio. Adicionalmente es el encargado de representar al equipo completo ante el cliente y hacer las veces de intermediario ante conflictos.
- Diseñador – Equipo control de cambios: conformado por dos o tres personas. Son programadores experimentados o analistas capaces de visualizar el software como un todo. Deben tomar los requerimientos procesados por el equipo cliente y armar la arquitectura de software, además de planificar el product backlog.

- Programador – Equipo de desarrollo: conformado por cinco personas, donde uno es el líder que regula la comunicación entre el equipo y los demás equipos. Adicionalmente administra las prioridades de las tareas y sus responsabilidades.
- Tester – Equipo de testing: conformado por dos personas, uno de ellos se encarga de generar las pruebas necesarias para las distintas características del software y el otro documenta el proceso y hace tracking de los errores y correcciones generados.

Para proyectos y empresas de desarrollo de software pequeñas lo recomendable es mantener un equipo conformado por cuatro personas, es decir; una persona por equipo/rol.

1.3.2.4 Prácticas

- Ganking: estrategia de colaboración que permite a cada miembro del equipo ayudar a quien más lo requiera en una etapa especifica del desarrollo del proyecto. Esta práctica viene a solucionar el constante problema de comunicación de las metodologías ágiles XP o FDD.
- Semana de 40 horas: esta práctica establece que la semana no debe superar las 40 horas de trabajo ya que se considera que es el tiempo ideal para mantener un desarrollo ágil y sostenido.
- Estrategia de diversificación holística: la idea central de esta práctica es incluir múltiples especialidades en un solo equipo. Esto facilita la expansión del equipo dependiendo de la longitud del proyecto.
- Reuniones diarias: esta práctica permite coordinar y establecer un plan de acción válido por día. Permite revisar que es lo que se ha hecho, lo que se hará y cómo se ejecutará una tarea en particular, además permite definir los responsables de cada tarea.
- Entregas semanales: práctica que establece que al final de cada semana se debe entregar software funcional.
- Líder diario: elegir un líder diario dependiendo de la actividad que se presente, el líder puede administrar y gestionar la ayuda recibida por sus pares.

- Unidades de proceso: macro ciclo y micro ciclo: 1 macro ciclo = 4 micro ciclos
- Jornadas de reflexión, retroalimentación y comunicación: práctica que establece que estas jornadas deben formar parte de las tareas de la semana.
- Semana de planificación (Semana 0): esta semana es una etapa crucial en el proceso de desarrollo, ya que permite armar el feature backlog, el product backlog, organizar los equipos/roles, definir el chatlog, en general las forma en la que se llevará el proyecto.
- Programación y testing XP: práctica que establece que el proceso de producción del equipo de desarrollo y el equipo de testing se realice siguiendo los parámetros de la metodología XP.
- Objetivo diario: este objetivo puede ser una o un conjunto de tareas que el líder diario debe revisar su cumplimiento.

1.3.2.5 Artefactos

- Feature backlog: conjunto de tareas escritas en pequeñas fichas que componen una característica del software. Lo recomendable es establecer, seguir y completar un feature backolog a la semana por equipo.

  El feature backlog está formado por tres secciones: “Por hacer”, “En Desarrollo” y “Listo”. Se pasa de la sección “Por Hacer” a “En Desarrollo” cuando se establecen las responsabilidades de cada tarea, se pasa a la sección “Listo”” cuando las tareas son completadas. Al final de cada semana se evalúa el avance en las sesiones de feedback y comunicación.
- Product backlog: es el conjunto de características del producto de software final. Las características se transforman en tareas y se agrupan posteriormente en varios feature backlogs. El product backlog es generado a partir de los requerimientos extraídos por parte del equipo cliente en la semana 0.
- Pizarra de ideas: artefacto auxiliar que sirve para exponer ideas o mejoras al software en desarrollo de tal forma de ofrecer un software más funcional.
- Chatlog: artefacto auxiliar que permite mantener una comunicación fluida entre los miembros de cada equipo. Además puede servir de

documentación para el registro de cambios. Ejemplos de chatlogs pueden ser: grupo de Facebook, conversaciones por twitter, whatsapp entre otros.

- Modelo Canvas: propuesto como artefacto auxiliar, permite visualizar al producto final como un todo más que como la suma de sus partes.

1.3.2.6 Procesos

Durante la ejecución de un macro ciclo se generan tres procesos principales: early game, middle game y late game.

A continuación se detallada cada uno de estos procesos.

1.3.2.6.1 Early game

Fase inicial del proyecto, por lo general hace referencia a la semana 0 y semana 1 (S0 y S1) justo antes de la primera entrega. En esta fase del proyecto se encuentra en una etapa de ajuste, el equipo comienza a acomodarse para generar los primeros avances del software, por esta razón la mayor parte del peso en esta etapa la lleva el equipo control de cambio. En la Figura 1.12 se muestra los equipos/roles que intervienen en esta fase y el ganking que genera cada equipo/rol.

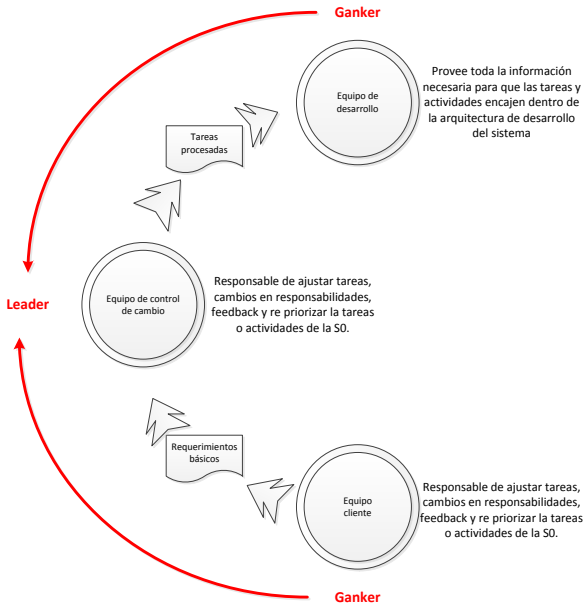

Figura 1.12 Fase Early game

1.3.2.6.2 Middle game

Es la etapa media del macro ciclo y contempla la semana dos y tres (S2 y S3) justo antes de los procesos de testing. En esta etapa se mantiene un desarrollo sostenido, rápido y constante del producto de software.

La Figura 1.13 muestra los equipos/roles que intervienen en la fase middle game y el ganking que genera cada equipo/rol.

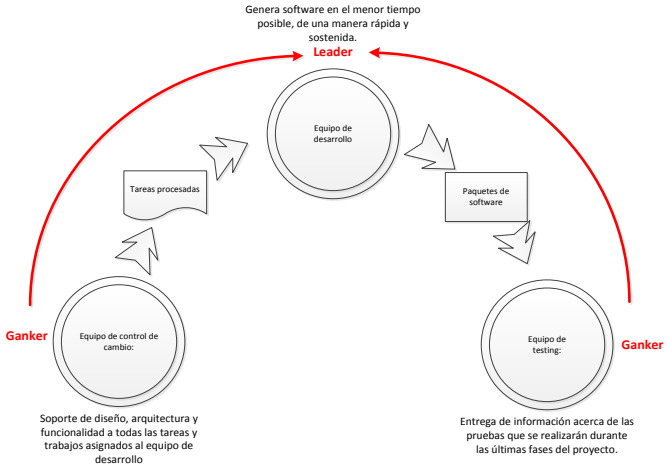

Figura 1.13 Fase Middle game

1.3.2.6.3 Late game

Es la etapa final del macro ciclo e involucra la participación del equipo de desarrollo y del equipo de testing. En la Figura 1.14 muestra el ciclo de vida, los equipos/roles y procesos que intervienen en la metodología estudiada.

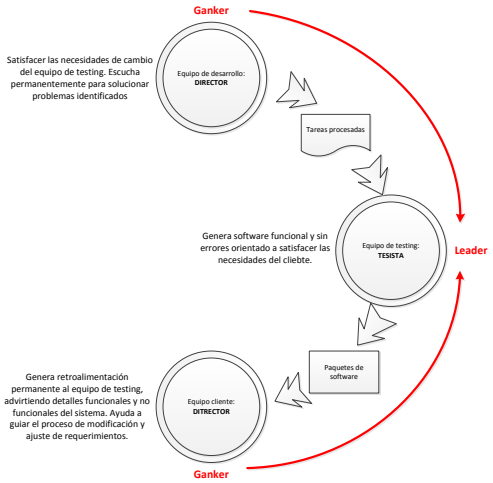

Figura 1.14 Fase Late game

Finalmente en la Figura 1.15 se muestran los equipos/roles, etapas y proceso de la metodología estudiada.

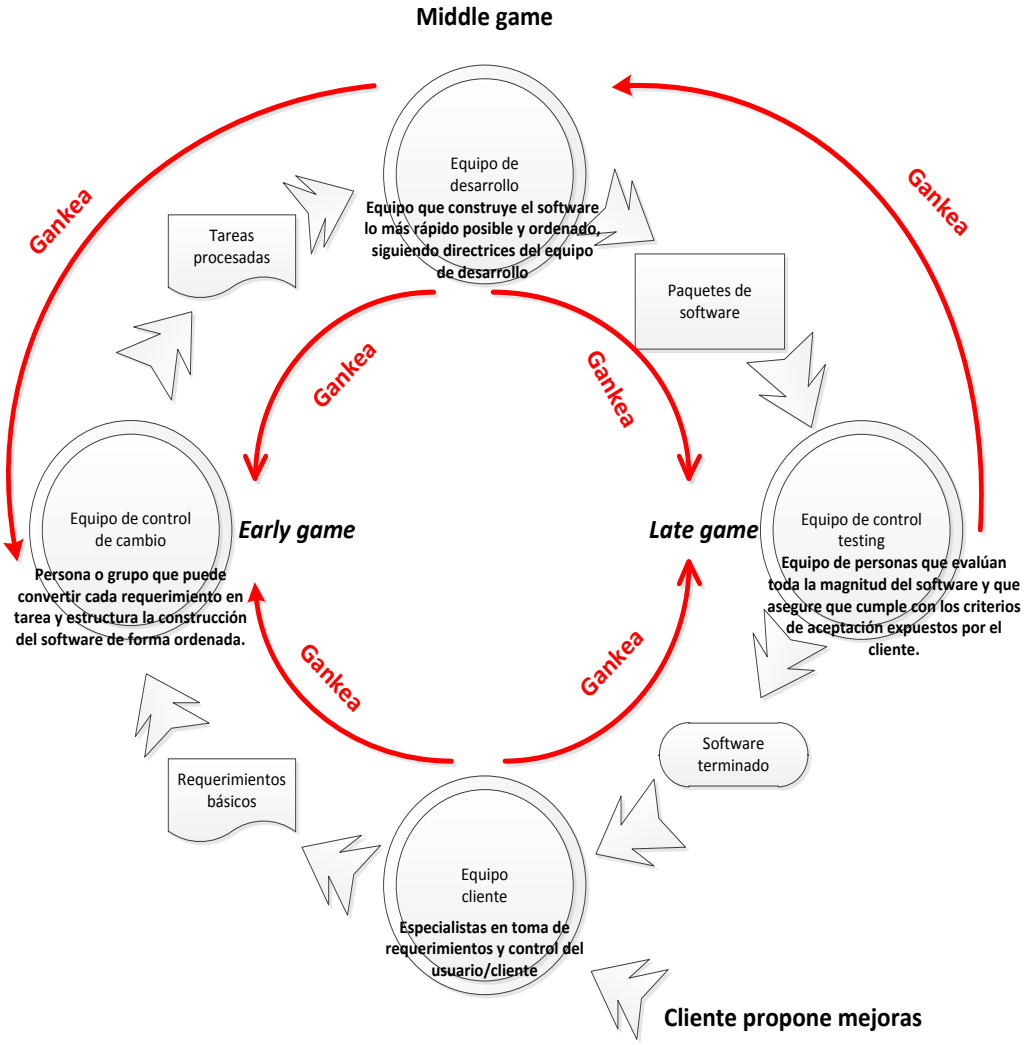

Figura 1.15 Equipos/roles, etapas y procesos de la metodología estudiada

1.3.3 SELECCIÓN DE LAS MEJORES PRÁCTICAS DE DESARROLLO DE LAS METODOLOGÍAS ÁGILES ESTUDIADAS

Producto del estudio de la metodología Mobile-D y metodología ágil hibrida para el desarrollo de software en dispositivos móviles, aquí se mencionan las prácticas de estas metodologías que serán adoptadas en el desarrollo del presente proyecto de titulación. A pesar de que cada metodología tiene diferentes fases y etapas, para el desarrollo del proyecto se logra identificar que las mismas siguen un desarrollo secuencial que comienza con una fase de análisis de requerimientos, la siguiente fase es de diseño, luego sigue la fase de codificación

y finalmente se tiene una fase de pruebas. Tomando en cuenta esta consideración, a continuación se detallan las prácticas que seguirá el presente proyecto en cada una de estas fases.

1.3.3.1 Fase de análisis de requerimientos

En el presente proyecto de titulación será usada la fase de *Exploración* que plantea la metodología Mobile-D, a continuación la justificación de esta selección:

- Uno de los objetivos del presente proyecto plantea un análisis de los trabajos previos [2]–[4] con la finalidad de establecer los requisititos del presente trabajo. Por esta situación se hará uso de todos los artefactos de software estudiados en la metodología Mobile-D.
- Los trabajos previos [2]–[4] en su desarrollo utilizaron varios artefactos de software que son usados en esta metodología. Historias de usuario (en base a estos podemos construir los casos de uso) y requisitos de software.
- En las reuniones con el director de tesis se usaron varios artefactos de software de esta metodología.
- Mobile-D es más robusta en el análisis de requerimientos vs la metodología ágil hibrida, lo que ayudará a definir de mejor manera el contexto.

1.3.3.2 Fase de Diseño

La fase que será usada en el presente proyecto de titulación es la fase de *Inicialización* de la metodología Mobile-D, la justificación de esta selección se detalla a continuación:

- Los artefactos de software generados en la fase de análisis de requerimientos son usados en la implementación de diagramas de actividades y diagramas clases.
- Los trabajos previos [2]–[4] usaron varios artefactos de software de esta metodología. Diagrama de actividades y diagramas de secuencia.
- Los diagramas de este estilo permitirán modelar correctamente y en detalle los procesos que se plantearon resolver al inicio del proyecto. Para la solución de los problemas es muy importante analizar y detallar los

algoritmos, y para esto los diagramas ayudarán debido a sus características inherentes.

1.3.3.3 Fase de Codificación

El proceso que será usado en esta fase será *Middle game* de la metodología ágil hibrida. A continuación la conformación de equipos y su justificación:

- Una persona en el equipo control de cambio (director)
- Una persona en el equipo de desarrollo (autor del proyecto)
- Una persona en el equipo de testing (director).
- Entrega semanal de módulos del presente proyecto. Reuniones semanales con director de tesis.
- Semanas de 40 horas. Debido a limitante personal de tiempo se plantea 15 horas entre semana y 12 en fines de semana. Total de horas semanales 27 horas.
- Jornadas de reflexión, retroalimentación y comunicación, en las reuniones semanales con director de tesis.

1.3.3.4 Fase de Pruebas

El proceso que será usado en esta fase será *Late game* de la metodología ágil hibrida. La conformación de equipos y justificación de esta selección se detalla a continuación:

- Una persona en equipo de desarrollo (director).
- Una persona en el equipo de testing (autor del proyecto).
- Una persona en el equipo cliente (director).
- Los escenarios de pruebas permitirán verificar el correcto funcionamiento de los módulos desarrollados en la fase *Middle game.*

CAPÍTULO 2

2 ESTUDIO DEL CONTEXTO

En el presente capítulo se realizará un estudio y análisis de los trabajos previos [2]–[4] con la finalidad de identificar los requerimientos del proyecto de titulación. Además, para tener un mejor entendimiento de los subsistemas de los trabajos previos [2]–[4] en el presente proyecto de titulación se usan los nombres: Subsistema servidor y Subsistema automóvil que se pueden ver en la Figura 2.1.

Adicionalmente en la Figura 2.1 se observa la arquitectura final del trabajo de titulación [4].

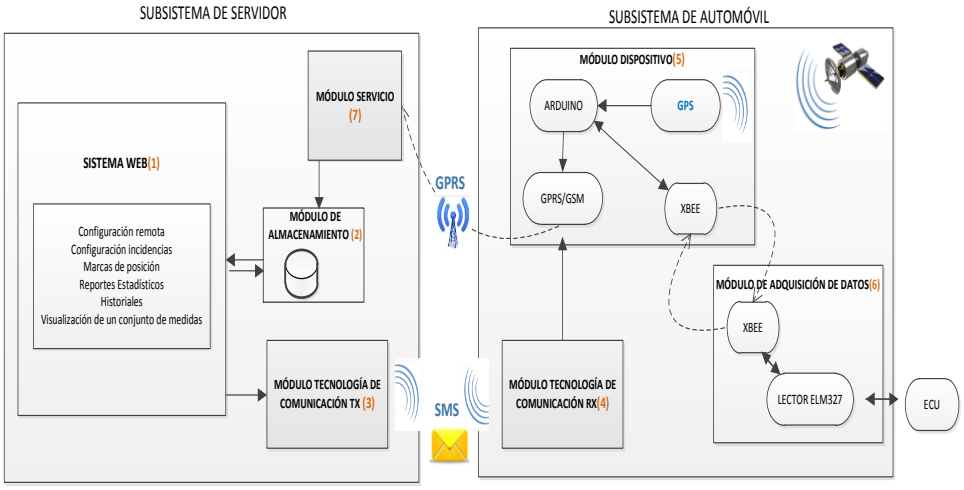

Figura 2.1 Arquitectura final de trabajo previo [4]

2.1 DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN TELEMÁTICA BASADA EN OBD-II (ON-BOARD DIAGNOSTIC) QUE PERMITA OBTENER Y PROCESAR LA INFORMACIÓN DE LOS SENSORES DEL MOTOR DE UN AUTOMÓVIL [2]

El proyecto está conformado por dos subsistemas, uno de transmisión ubicado en el automóvil que es el encargado de establecer conexión con la ECU para procesar la información que luego será enviada al servidor remoto y otro

subsistema de recepción (servidor) que se encarga de recibir la información, verificar la información en base a un formato establecido y guardarla en un sistema de almacenamiento para luego ser visualizada mediante una interfaz web básica. La Figura 2.2 muestra la arquitectura de este trabajo de titulación.

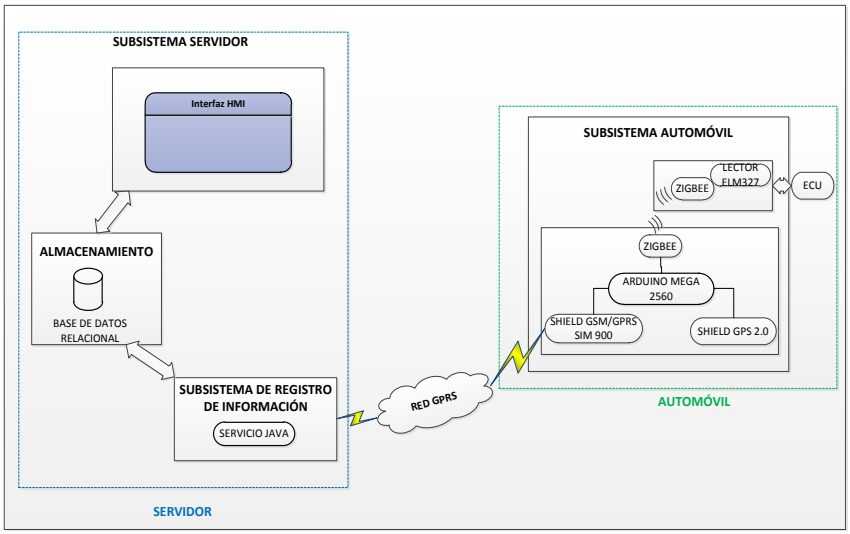

Figura 2.2 Arquitectura del sistema trabajo de titulación [2]

2.1.1 SUBSISTEMA AUTOMÓVIL

Encargado de establecer la conexión con la ECU del automóvil, procesar la información y enviarla al servidor remoto.

Los módulos que conforman el subsistema automóvil son:

- Módulo de adquisición de datos: conformado por un lector ELM327 USB que tiene adaptado un módulo XBee que permite transmitir la información de forma inalámbrica. El lector ELM327 se encuentra conectado directamente a la interfaz DLC del automóvil mientras que el módulo XBee se encuentra conectado a los pines de Tx y Rx que posee el lector ELM327.
- Módulo de procesamiento y transmisión de datos: conformado los siguientes elementos: un *módulo XBee* para la conexión ZigBee con el módulo de adquisición, un *dispositivo Arduino Mega 2560* que se encarga del procesamiento de la información recibida de la ECU y de la creación de la trama que será enviada al servidor remoto y finalmente el *módulo Shield GSM/GPRS SIM 900* que se encarga del envío de la trama al servidor

remoto usando la infraestructura GPRS de la red de telefonía móvil. Cabe mencionar que la comunicación entre el Arduino Mega 2560 y el ELM327 es mediante comandos AT.

En la Figura 2.3 se muestra un diagrama en bloques del subsistema automóvil.

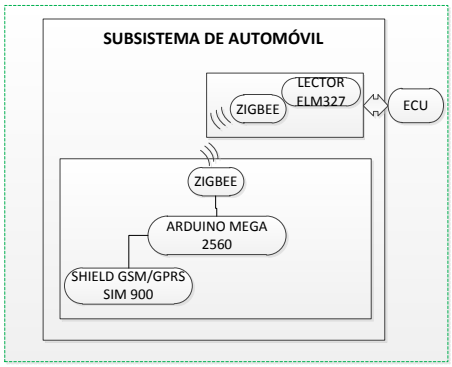

Figura 2.3 Diagrama en bloques del subsistema automóvil

La Figura 2.4 detalla el formato de la trama enviada.

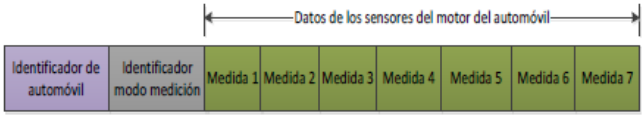

Figura 2.4 Formato de trama generada por subsistema de recepción [2]

2.1.2 SUBSISTEMA SERVIDOR

Este subsistema es el encargado de recibir la información desde el automóvil, validar la información en base a un formato establecido, guardar en una base de datos y luego colocar a disponibilidad del usuario mediante una interfaz web básica.

A continuación se detallan los módulos del subsistema servidor:

- Módulo de servicio: es un programa en Java que se ejecuta en segundo plano y que es el encargado de: permanecer en modo escucha de las tramas enviadas desde subsistema automóvil, una vez que recibe los datos los lee y decodifica en base a un formato establecido; si los datos cumplen con el formato establecido son almacenados en una base de datos.
- Módulo de almacenamiento de información: encargado de recopilar, organizar y guardar los datos provenientes del subsistema automóvil en

una base de datos, garantizando consistencia, integridad y disponibilidad de los mismos.

Los datos almacenados son utilizados como fuente de información de reportes generados por el módulo de visualización de información.

- Módulo visualización de información: este módulo es la parte visible para el usuario y está constituido por una interfaz web básica que le permite acceder a la información generada por los sensores del automóvil y de ser el caso generar reportes con dicha información.

2.2 IMPLEMENTACIÓN DE UN SISTEMA DE ADMINSITRACIÓN REMOTA PARA EL PROCESO DE OBTENCIÓN DE DATOS DEL SISTEMA OBD-II DE UN AUTOMÓVIL [3]

El proyecto se orienta a configurar de manera remota y variable el proceso de lectura del módulo de adquisición de datos, esto se consigue mediante el envío de un mensaje de configuración vía SMS desde el subsistema servidor al subsistema automóvil.

En la Figura 2.5 se observa la arquitectura del proyecto de titulación, en rojo los módulos que fueron añadidos o modificados respecto al trabajo previo [2].

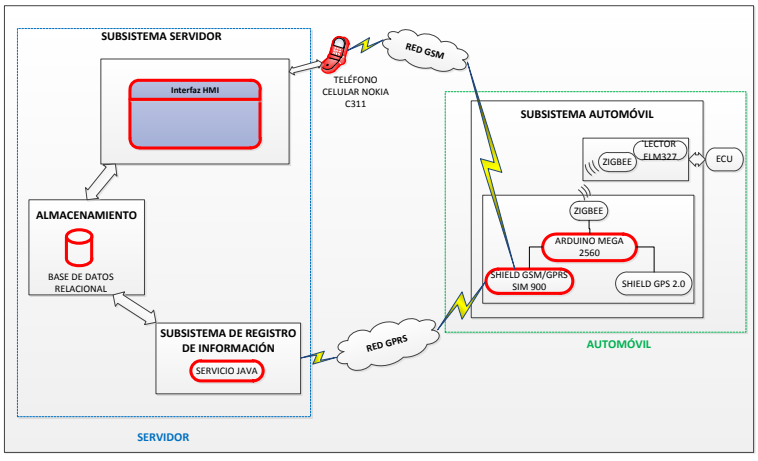

Figura 2.5 Arquitectura del proyecto propuesto en [3] y módulos modificados respecto al trabajo previo [2]

2.2.1 SUBSISTEMA AUTOMÓVIL

Hace uso del módulo GSM que es ofrecido por el dispositivo Shield GSM/GPRS SIM 900, esto permite usar el servicio de SMS para la recepción de un mensaje

de configuración. El mensaje de configuración tiene información respecto a los PIDs que debe leer el lector ELM327 desde la ECU, adicionalmente tiene el identificador del automóvil y el tiempo de muestreo.

Debido a estas modificaciones, el código fuente del Arduino Mega 2560 fue modificado para soportar la medición de n PIDs y por ende el formato de trama que genera el subsistema también fue cambiado. En la Figura 2.6 se puede observar el nuevo formato de trama generado por el subsistema servidor y sus delimitadores.

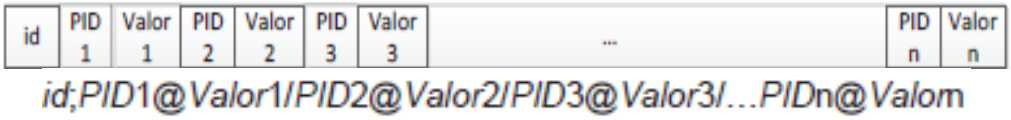

Figura 2.6 Formato de trama generado por subsistema servidor [3]

2.2.2 SUBSISTEMA SERVIDOR

Este subsistema se orienta primordialmente a proveer de una interfaz web a través de la cual el usuario puede configurar los parámetros a medir en el automóvil. Para ello se implementaron las siguientes funcionalidades:

- Módulo de servicio: el servicio Java fue modificado de tal forma que pueda recibir una trama de tamaño variable y que cumpla con el formato establecido.
- Módulo de almacenamiento de datos: se generó un nuevo esquema de base de datos que permite guardar la información de administración y configuración remota, adicionalmente garantiza integridad de información
- Módulo visualización de información**:** se implementa una nueva interfaz web dinámica que mejora la navegabilidad, basada en formularios que facilitan la administración y configuración de varios PIDs soportados por sistema OBD II y que permite al usuario elegir el tiempo de muestreo de dicha información.
- Módulo de procesamiento: constituye la parte central de la arquitectura propuesta en el presente proyecto ya que este módulo interactúa directamente con la base de datos relacional y con la tecnología de comunicación inalámbrica. En este módulo se recibe y procesa la

información de administración y configuración que será enviada al subsistema de recepción ubicado en el automóvil

- Tecnología de comunicación de transmisión: con la finalidad de transmitir el mensaje de configuración al subsistema automóvil, se usó un celular Nokia C311 configurado como módem que envía los datos vía SMS.

2.3 DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA EL MONITOREO DEL ESTADO DE UN MOTOR [4]

La principal ventaja del proyecto es el proveer de un sistema web adecuadamente diseñado que reemplace los formularios del trabajo previo [3]. En el desarrollo del sistema se utilizaron fundamentos de ingeniería de software como por ejemplo se aplicó una metodología, se diseñó usando el Modelo Vista Controlador, se usó un framework para ganar responsividad en la aplicación y se rediseño la base de datos para garantizar escalabilidad.

Asimismo, se presentó aparte de la configuración remota, administración del sistema y reportes, un módulo de configuración de seguimiento de incidencias. Adicionalmente, se agregó el Shield GPS 2.0 para proveer de marcas de posicionamiento a las medidas y poder hacer un seguimiento desde la interfaz gráfica a través de un mapa.

La Figura 2.7 muestra la arquitectura y los módulos del proyecto de titulación.

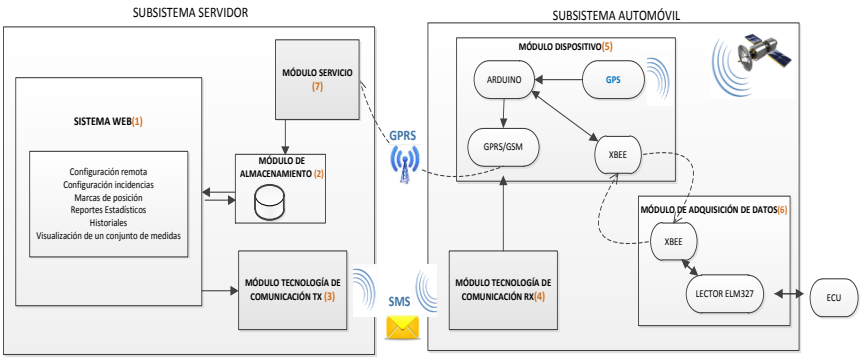

Figura 2.7 Arquitectura y módulos del proyecto de titulación [4]

Con la finalidad de tener un mejor entendimiento de la solución y arquitectura del presente proyecto, a continuación se describen los módulos de todo el sistema:

- Sistema Web (1), permite al usuario administrador y help desk el registro, edición o eliminación de usuarios, dispositivos, automóviles, parámetros de medición e incidencia. También permite realizar la configuración de incidencias con la finalidad de detectar si los valores monitoreados pasan el umbral establecido en la incidencia, en caso de superar dicho umbral se genera un correo electrónico de notificación al cliente.
- Almacenamiento (2), este módulo se realiza el almacenamiento de información generada por cada uno de los formularios del sistema web en la base de datos.
- Comunicación de transmisión (3) y recepción (4), dichos módulos permiten enviar y recibir respectivamente el mensaje de configuración haciendo uso de la red GSM.
- Dispositivo (5), encargado de la decodificación del mensaje de configuración, transmisión de los PIDs recibidos al módulo adquisición de datos (6), recepción de valores de PIDs, marcas de posición, establecimiento de trama que será enviara al subsistema de transmisión e interconexión con la infraestructura GPRS,
- Adquisición de datos (6), este módulo se conecta a la interfaz DLC del automóvil y lee la información de la ECU en función de los PIDs decodificados del mensaje de configuración.
- Servicio (7), la función del módulo es la recepción de la trama generada por el módulo Dispositivo (6) del subsistema de recepción ubicado en el automóvil.

2.3.1 SUBSISTEMA AUTOMÓVIL

Como se mencionó anteriormente, al subsistema automóvil se le agregó un módulo Shield GPS 2.0 con la finalidad de capturar las marcas GPS y de esta forma agregar estas mediciones a la trama de respuesta que será enviada al subsistema servidor. El código fuente del Arduino Mega 2560 fue modificado con la finalidad de soportar al nuevo módulo y enviar un nuevo formato de trama que ahora incluyen las marcas de posicionamiento. En la Figura 2.8 se detalla el formato de la nueva trama de respuesta con sus respectivos delimitadores.

id auto	id configuración	PID 1	Valor 1	PID 2	Valor 2	.....	PID n	Valor n	latitud	longitud

id auto ; id configuración ; PID1 @ Valor1 / ..... / PID n @ Valor n / lat @ Valor lat / long @ Valor long /

Figura 2.8 Formato de trama de respuesta generado por subsistema automóvil y delimitadores

2.3.2 SUBSISTEMA SERVIDOR

El subsistema servidor es configurado con doce módulos que se pueden ver en la Figura 2.9. Se puede notar que el enfoque del proyecto fue aplicar ingeniería de software a los formularios, para generar un sistema web.

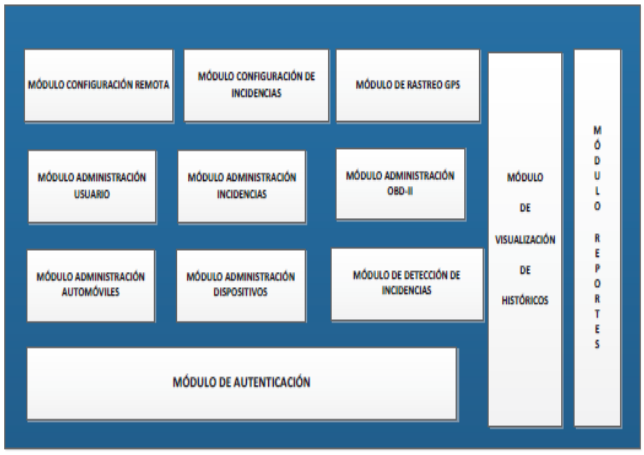

Figura 2.9 Módulos del subsistema servidor

El módulo de autenticación permite validar mediante un nombre de usuario y contraseña si un usuario se encuentra registrado en el subsistema y si es de tipo administrador, cliente o help desk.

El siguiente módulo es el de administración de automóviles el cual se encarga de registrar, editar o eliminar marcas, modelos y automóviles.

El módulo administración de dispositivos es el encargado de registrar, editar o eliminar tarjetas SIM y dispositivos Arduino. El número telefónico registrado en la SIM será al cual se envíen los SMS de configuración.

En el módulo administración de OBD II se puede registrar, editar o eliminar los diferentes modos que son soportados por el sistema OBD II así como los diferentes PIDs.

En el módulo administración de usuarios se puede registrar, editar o eliminar los diferentes usuarios que tienen acceso al subsistema así como también el rol del usuario dentro del subsistema.

El módulo administración de incidencias permite registrar, editar o eliminar las incidencias. Una incidencia es un evento que se produce cuando los valores de ciertos parámetros monitoreados pasan un umbral establecido; por ejemplo se podría configurar una incidencia que indique cuando se ha superado los 50 km por hora. Asimismo, se puede configurar incidencias que dependan de varios parámetros simultáneamente.

El módulo configuración remota permite seleccionar los parámetros y la frecuencia con la que serán obtenidos desde el sistema OBD II de un automóvil.

El módulo configuración de incidencias, permite seleccionar las incidencias registradas previamente y asociarlas a un vehículo que está siendo monitoreado.

El módulo de detección de incidencias, encargado de detectar la incidencia una vez ocurra; asimismo, el usuario será notificado de la incidencia ocurrida mediante un correo electrónico.

El módulo de rastreo GPS, permite que el usuario vea la posición actual del automóvil en conjunto con los parámetros seleccionados a través de un mapa. Adicionalmente este módulo ofrece un historial de las ubicaciones del automóvil.

El módulo de visualización de históricos, ofrece al usuario una interfaz que le permite seleccionar el rango de tiempo de la información que desea observar.

El módulo de reportes, permite que el usuario genere un gráfico estadístico en función de las medidas capturadas o de las incidencias registradas.

2.3.3 ACTUALIZACIONES DE LOS MÓDULOS QUE CONFORMAN LA SOLUCIÓN FINAL TRATADA EN LOS TRABAJOS PREVIOS [2]–[4]

En función de la descripción realizada en la sección 2.1, 2.2 y 2.3, en la Tabla 2.1 y Tabla 2.2 se realiza un cuadro comparativo de todos los módulos que contiene la solución final, y su evolución en los trabajos previos [2]–[4].

MÓDULOS	**Trabajo previo** [2]	**Trabajo previo** [3]	**Trabajo previo** [4]
Sistema Web	Aplicación web básica, con funcionalidad para visualizar información. No transmite mensaje de configuración.	Sistema web dinámico basado en formularios que permiten generar reportes.	Sistema web dinámico con facilidades para registrar, editar y eliminar: usuarios, dispositivos, automóviles, parámetros de medición e incidencia. Brinda facilidades de configuración y seguimiento. Usa Modelo Vista Controlador, y es responsiva, usa la API de Google Maps para mostrar las medidas sobre un mapa. Ante una incidencia genera un correo electrónico de notificación al cliente.
Almacenamiento	Base de datos que almacena información de un conjunto de PIDs fijo. Tablas no normalizadas.	Base de datos que almacena información de administración, configuración remota y de un conjunto variable de medidas de PIDs. Tablas no normalizadas.	Base de datos diseñada cumpliendo criterios de integridad, y para que la misma sea escalable. Almacena información de usuarios, perfiles, modos y PIDs que soporta el sistema, incidencias que soporta el sistema, vehículos, configuración remota de parámetros a rastrear, configuración de incidencias a rastrear, histórico de registros de medidas y de incidencias.
Tecnología de comunicación desde el subsistema servidor al subsistema automóvil.	No se lo implementa.	Constituido por un celular que se encuentra conectado al servidor y configurado para enviar SMS. Es el encargado de transmitir el mensaje de configuración generado por el formulario del sistema web.	Se mantiene el módulo.
Tecnología de comunicación desde subsistema automóvil al subsistema servidor.	No se lo implementa.	Formado por el dispositivo Shield GSM/GPRS SIM 900 (funcionalidad GSM). Encargado de recibir el mensaje de configuración que es enviado por el módulo se transmisión.	Se mantiene el módulo.

Tabla 2.1 Evolución de los módulos de los trabajos previos [2]–[4]

MÓDULOS	**Trabajo previo** [2]	**Trabajo previo** [3]	**Trabajo previo** [4]
Dispositivo	Formado por los dispositivos: Arduino Mega 2560, XBee, Shield GSM/GPRS SIM 900 (funcionalidad GPRS). Es el encargado de transmitir un conjunto fijo de medidas de los PIDs establecidos. Además se encarga de establecer el formato de trama que será enviado al subsistema servidor.	Debido a que ahora se soporta la configuración remota, se modifica el módulo y se le da la funcionalidad de decodificar el mensaje de configuración, guardar la configuración establecida, y en función de ella iniciar el proceso de lectura de los datos de la ECU. Asimismo, toma los datos los encapsula en tramas, y los envía al subsistema servidor.	Se toma el módulo desarrollado y se modifica el algoritmo para agregar marcas de posición y enviarlos en conjunto con los datos de la ECU.
Adquisición de datos	Constituido por el lector ELM327 y el dispositivo XBee. Este módulo va conectado al interfaz DLC de automóvil y la función principal es la de obtener la información de la ECU.	Se mantiene el módulo.	Se mantiene el módulo.
Servicio	Programa realizado en JAVA cuya función es la de recibir la trama, validar el formato de la misma para posteriormente almacenarla en el módulo de almacenamiento.	Se modifica el programa para que esta vez se compruebe el formato y se almacenen los datos considerando que llegan una cantidad variable de mediciones.	Se modifica el programa para que esta vez pueda leer las marcas de posición.

Tabla 2.2 Evolución de los módulos de los trabajos previos [2]–[4] (continuación)

2.3.4 ESTUDIO DE LOS MÓDULOS DEL SUBSISTEMA VEHÍCULO ACTUAL [3]

Con la finalidad de identificar los requerimientos del presente trabajo de titulación y debido a que el subsistema automóvil será reemplazado por un teléfono inteligente Android de bajo costo, en la presente sección se hará un estudio más detallado de los proceso que son usados en los módulos de dicho subsistema.

2.3.4.1 Interacciones entre los componentes del subsistema automóvil

En la Figura 2.10 se detallan las interacciones entre los componentes del subsistema automóvil.

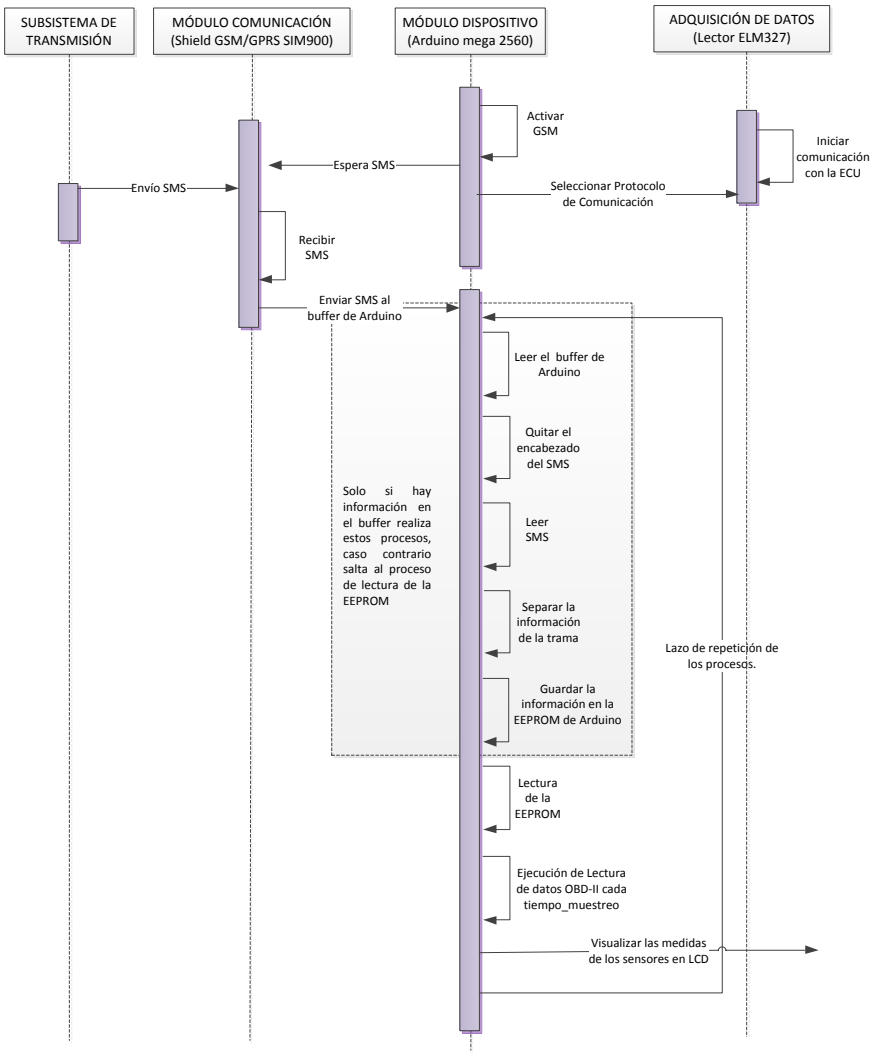

Figura 2.10 Interacciones entre componentes del subsistema automóvil [3]

Como se puede apreciar en el diagrama de secuencia, el dispositivo siempre comprobará como primer paso si hay un nuevo mensaje de configuración en el buffer, si no hay avanza directamente a la lectura de la EEPROM.

Cuando se tiene un mensaje de configuración en el buffer, el dispositivo quita el encabezado del SMS y lo lee, separa la información de la trama y guarda la información en la EEPROM del Arduino.

En cualquier caso, se pasa a la lectura de la EEPROM para cargar la información de los PIDS que se deben leer, y posteriormente se realiza la lectura acorde al tiempo de muestreo, se forma la trama y se envían los datos.

En esta parte del proceso se puede identificar el primer problema debido a que en caso de que llegue un mensaje de configuración luego de que el proceso normal

pase la lectura del buffer, es necesario esperar que el dispositivo finalice una iteración completa que incluye la ejecución de lectura de datos OBD II cada tiempo de muestreo. Tomando como ejemplo el tiempo de muestréo mínimo que es de 1 minuto y que el proceso de lectura del buffer e iteración se efectúe en 1 segundo, el tiempo que debe esperar a que se ejecute un nuevo mensaje de configuración es de 59 segundos. La Figura 2.11 muestra en rojo el tiempo máximo que se esperaría hasta leer el nuevo mensaje de configuración y aplicarlo.

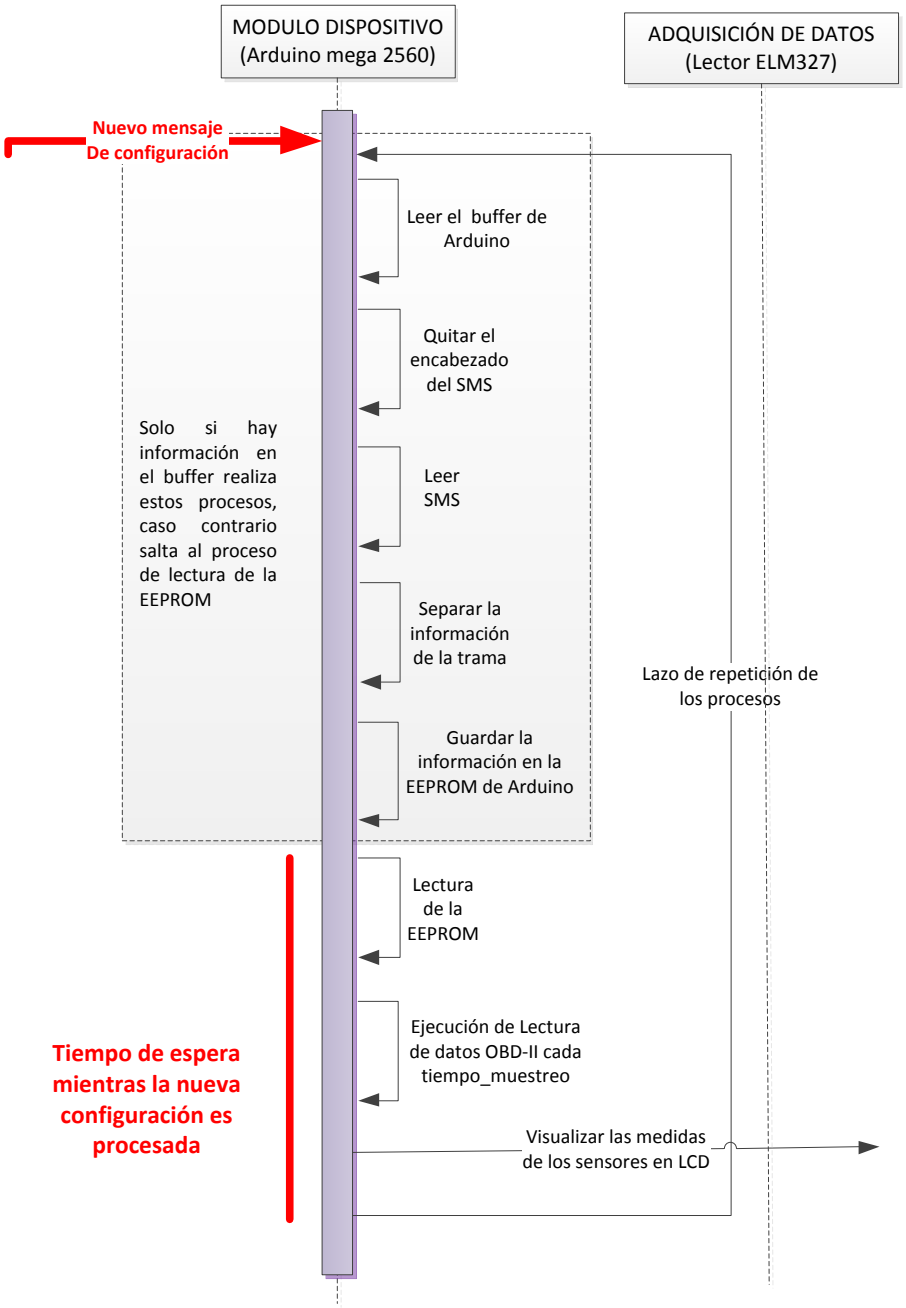

Figura 2.11 Tiempo de ejecución de un nuevo mensaje de configuración [3]

2.3.4.2 Comunicación entre módulo dispositivo y módulo de servicio

La comunicación entre el módulo dispositivo y módulo de servicio se hace a través del dispositivo Shield GSM/GPRS SIM 900 y el uso de la infraestructura GPRS.

Bajo este esquema y producto del análisis de los trabajos previos [2]–[4], se pudo notar que en el caso de no existir cobertura GPRS en el dispositivo Shield GSM/GPRS SIM 900 ubicado el automóvil la información se pierde ya que no existe ningún módulo de almacenamiento de información ante la ausencia de Internet.

Este segundo problema del subsistema automóvil será corregido en el presente proyecto de titulación.

2.3.4.3 Costo del subsistema de recepción

En la Tabla 2.3 se detallan los costos incurridos en el hardware del subsistema automóvil.

DESCRIPCIÓN		COSTO
Arduino Mega 2560		$ 55,00
Lector ELM327		$ 37,00
MÓDULO ZIGBEE	XBee serie 1	$ 70,00
	XBee Regulated	$ 25,00
	XBee Adapter	$ 17,00
Shield SIM900 Shield		$ 90,00
Shield GPS 2.0		$ 45,00
TOTAL		**$ 339,00**

Tabla 2.3 Costos del hardware del subsistema automóvil

Como se puede observar existe un elevado costo en el subsistema automóvil por lo que en el presente trabajo de titulación se propondrán alternativas para reducir dicho costo.

2.3.5 ANÁLISIS DE SMARTPHONES DE BAJO COSTO

Como se pudo observar en la sección anterior, el costo del hardware del subsistema de recepción ha llegado a la cifra de $339,00 siendo esto una limitante para la implementación en un escenario real. En la presente sección se

hará un análisis de Smartphones de bajo costo que podrían ser usados para reemplazar el módulo dispositivo del subsistema automóvil.

2.3.5.1 Smartphones de bajo costo

La tendencia de renovación de teléfonos inteligentes o Smartphones actualmente es superior a dos años. Locamente las operadoras de telefonía celular indican que el tiempo de renovación de equipos celulares es de 24 meses [23].

En nuestro mercado no se encuentran Smartphones nuevos de bajo costo (alrededor de $80) [24], [25]; por lo tanto se ha decidido tomar como referencia terminales 2014, que estarían próximos a ser renovados y que por ello tienen un precio bajo. Esto a su vez permitirá cumplir con uno de los objetivos del presente proyecto de titulación que es el re-uso de Smartphones con sistema operativo Android. En la Tabla 2.4 se muestra un catálogo de Smartphones con sistema operativo Android con precios de venta al público del 2014 de Movistar y Claro con el precio referencial 2016 tomando como fuente tiendas online locales. El precio referencial de tiendas online locales se toma debido que Movistar y Claro no ofrecen un precio actualizado de estos terminales ya que están fuera de su actual catálogo.

Marca/Modelo	***PVP 2014 MOVISTAR***	***PVP 2014 CLARO***	***PVP 2016***
Huawei Ascend P6	*$ 729,00*	*No dispone*	*$ 130,00*
Huawei Ascend Y320	*$ 299,00*	*$ 137,00*	*$ 50,00*
LG L1 II	*$ 239,00*	*No dispone*	*$ 60,00*
LG L3 II	*$ 319,00*	*$ 158,00*	*$ 50,00*
LG L7 II	*$ 435,00*	*$ 310,00*	*$ 80,00*
Samsung Galaxy Ace 3	*$ 409,00*	*$ 226,00*	*$ 70,00*
Samsung Galaxy Ace S III Mini	*$ 629,00*	*$ 569,00*	*$ 80,00*
Samsung Galaxy Fame	*$ 429,00*	*$ 254,00*	*$ 60,00*
Samsung Galaxy S4	*$ 1.129,00*	*$ 1.200,00*	*$ 150,00*

Tabla 2.4 Cuadro comparativo de Smartphones Android con PVP 2014 y PVP 2016 [26]

Debido a que uno de los objetivos del presente proyecto de titulación es el re-uso de Smartphones con sistema operativo Android que no superen los $80 se concluye que las mejores opciones son: LG L1 II, LG L3 II, LG L7 II, Samsung Galaxy Ace 3, Samsung Galaxy Ace S III Mini, Samsung Galaxy Fame.

Debido a que se cuenta con el acceso a los Smartphone LG L3 II y Samsung Galaxy Ace 3 y principalmente cumplen con los objetivos del presente proyecto de titulación, estos dos modelos son los seleccionados para el desarrollo del proyecto.

2.3.5.2 Especificaciones técnicas LG L3 II

CARACTERÍSTICAS	LG E425 L3 II
Pantalla táctil	✓
Tamaño pantalla	3.2"
Sistema operativo	Android 4,1
Resolución de cámara	3,15 MP
Radio	✓
Capacidad de almacenamiento	✓ 4GB interno, hasta 64GB externo
WiFi	✓
Bluetooth	✓
GPS	✓
Transmisión de datos	GPRS/EDGE/HSDPA
Microprocesador	Qualcomm MSM7225A - 1 Ghz
RAM	512 MB
SIM card	NORMAL

Tabla 2.5 Especificadores técnicas del Smartphone LG L3 II [26]

2.3.5.3 Especificaciones técnicas Samsung Galaxy Ace 3

CARACTERÍSTICAS	Samsung Galaxy Ace 3
Pantalla táctil	✓
Tamaño pantalla	4''
Sistema operativo	Android 4,2
Resolución de cámara	5 MP
Radio	✓
Capacidad de almacenamiento	✓ 4GB interno, hasta 64GB externo
WiFi	✓
Bluetooth	✓
GPS	✓
Transmisión de datos	GPRS/EDGE/HSDPA
Microprocesador	Dual Core - 1 Ghz
RAM	1GB
SIM card	NORMAL

Tabla 2.6 Especificadores técnicas del Smartphone Samsung Galaxy Ace 3 [26]

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DE LOS SUBSISTEMAS DE LECTURA DE DATOS

A continuación se detallan las fases de análisis de requerimientos, diseño y codificación, considerando las mejores prácticas de las metodologías del capítulo 1.

3.1 ANÁLISIS DE REQUERIMIENTOS

3.1.1 REQUISITOS DE USUARIO

En la siguiente sección se muestran los requisitos de usuario producto del análisis del capítulo anterior. El identificador de los requisitos de usuario pueden ser de dos tipos: RUR (Requisito de Usuario de Restricción) y RUC (Requisito de Usuario de Capacidad).

Identificador: RUR-01	
Título:	Reemplazar módulo dispositivo por un Smartphone.
Descripción:	El subsistema automóvil deberá usar un Smartphone de bajo costo (alrededor de $80) que reemplazará al **módulo dispositivo** ubicado en el automóvil.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.1 Requisito de usuario para reemplazar el módulo dispositivo de trabajo previo [4]

Identificador: RUR-02	
Título:	Tecnología de transmisión compatible.
Descripción:	El módulo de adquisición de datos del subsistema automóvil deberá soportar una tecnología de transmisión compatible con el Smartphone de bajo costo.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.2 Requisito de usuario para garantizar compatibilidad entre módulo dispositivo y módulo adquisición de datos de trabajo previo [4]

Identificador: RUC-01	
Título:	Verificar parámetros de configuración.
Descripción:	Verificar que todos los parámetros de configuración necesarios para que inicie la aplicación móvil se encuentren configurados correctamente.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.3 Requisito de usuario para verificar parámetros de configuración de aplicación móvil

Identificador: RUC-02	
Título:	Ejecución inmediata de mensaje de configuración.
Descripción:	El subsistema automóvil deberá ejecutar el mensaje de configuración inmediatamente luego de haberlo recibido.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.4 Requisito de usuario para ejecución inmediata de mensaje de configuración

Identificador: RUC-03	
Título:	Almacenamiento de información en zonas sin cobertura GPRS.
Descripción:	El subsistema automóvil deberá almacenar el mensaje de respuesta con la información de los sensores en zonas en las que no exista cobertura GPRS.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.5 Requisito de usuario para almacenamiento de información en zonas sin cobertura GPRS

Identificador: RUR-03	
Título:	Integración de subsistemas.
Descripción:	El subsistema automóvil deberá integrarse al subsistema servidor del trabajo previo [4].
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	Cliente

Tabla 3.6 Requisito de usuario para integración con el subsistema servidor de trabajo previo [4]

3.1.2 DIAGRAMAS DE CASOS DE USO

En la Figura 3.1 se puede observar el diagrama de casos de uso producto del análisis de los trabajos previos [2]–[4].

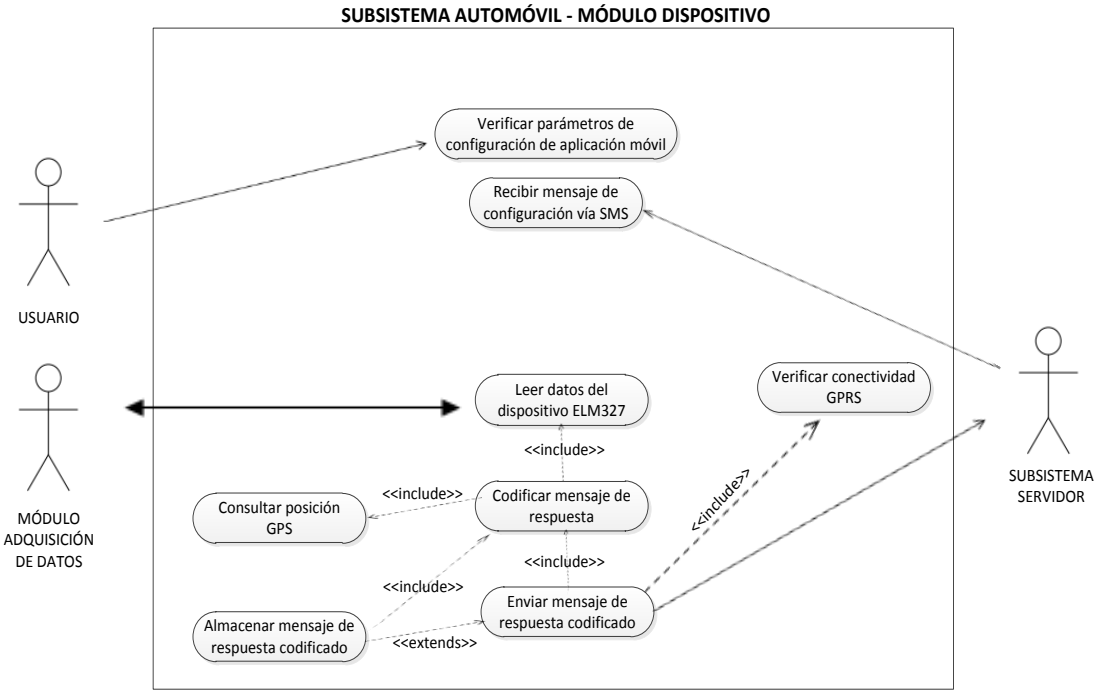

Figura 3.1 Diagrama de casos del módulo dispositivo

3.1.3 DESCRIPCIÓN DE CASOS DE USO

En la siguiente sección se presentan las tablas de descripción de los casos de uso.

Identificador: CU-01	
Nombre:	Verificar parámetros de configuración de aplicación móvil.
Objetivo:	El usuario verifica que la aplicación móvil cuenta con los parámetros necesarios para su funcionamiento.
Precondiciones:	El Smartphone deberá estar encendido y contar con lo siguiente: Bluetooth, GPS, plan de datos y SMS activo.
Postcondiciones:	La aplicación móvil establece conexión con lector ELM327 y se mantiene a la espera del mensaje de configuración vía SMS.

Escenario básico	**Escenario alternativo**
1. La aplicación móvil muestra un botón para verificar que todos los parámetros de configuración se encuentran habilitados correctamente.	
2. El usuario presiona el botón.	
3. La aplicación móvil verifica que todas las configuraciones fueron habilitadas y muestra un mensaje de confirmación.	
4. En caso de no tener habilitadas las configuraciones correctamente, la aplicación móvil muestra al usuario el menú de la configuración con las conexiones pendientes de habilitar.	4.1. El usuario habilita GPS. 4.2. El usuario habilita Bluetooth. 4.3. El usuario habilita datos móviles.

Tabla 3.7 Caso de uso: Verificar parámetros de configuración de aplicación móvil

Identificador: CU-01	
Escenario básico	**Escenario alternativo**
5. La aplicación móvil muestra la lista de dispositivos Bluetooth cercanos.	
6. El usuario busca la MAC address del lector ELM327 en la lista de dispositivos Bluetooth cercanos.	6.1. Si no aparece la MAC address del lector ELM327 en la lista de dispositivos cercanos, revisar conexión del lector ELM327 en el conector DLC en el automóvil.
7. El usuario presiona sobre la MAC address del lector ELM327 e introduce el PIN de acceso (depende del modelo de Smartphone aplica este paso).	
8. Si el PIN es el correcto, el lector ELM327 queda pareado con la aplicación móvil.	8.1. Si el PIN de acceso es incorrecto, repetir paso 6.
9. La aplicación móvil establece conexión con el lector ELM327.	9.1. Si la conexión falla, la aplicación móvil muestra un mensaje para reintentar conexión.
10. La aplicación móvil configura conexión con lector ELM327.	
11. La aplicación móvil queda a la espera del mensaje de configuración vía SMS.	

Tabla 3.8 Caso de uso: Verificar parámetros de configuración de aplicación móvil (continuación)

Identificador: CU-02	
Nombre:	Recibir mensaje de configuración vía SMS.
Objetivo:	La aplicación móvil recibe el mensaje de configuración desde el subsistema servidor vía SMS.
Precondiciones:	La aplicación móvil inicializada correctamente y conectada a lector ELM327.
Postcondiciones:	Mensaje de configuración en formato .txt guardado en unidad de almacenamiento interno.

Escenario básico	**Escenario alternativo**
1. La aplicación móvil en modo escucha de eventos.	
2. La aplicación móvil detecta evento SMS.	
3. La aplicación móvil valida si el mensaje de configuración recibido vía SMS cumple con el formato establecido.	3.1. En caso de no cumplir con el formato establecido, aplicación móvil descarta mensaje recibido y vuelve a paso 1.
4. La aplicación móvil guarda el mensaje de configuración en formato .txt en unidad de almacenamiento interno del dispositivo móvil.	

Tabla 3.9 Caso de uso: Recibir mensaje de configuración vía SMS

Identificador: CU-03	
Nombre:	Leer datos del dispositivo ELM327
Objetivo:	La aplicación móvil lee datos del dispositivo ELM327
Precondiciones:	Parámetros para procesos de adquisición de datos almacenados en memoria correctamente.

Tabla 3.10 Caso de uso: Leer datos del dispositivo ELM327

Identificador: CU-03	
Postcondiciones:	La aplicación móvil almacena en memoria el formato de respuesta con los valores de los sensores.

Escenario básico	Escenario alternativo
1. La aplicación móvil verifica la existencia de archivo de configuración .txt.	1.1. Esperar el siguiente ciclo de lectura para verificar la existencia de archivo de configuración .txt.
2. La aplicación móvil decodifica el mensaje de configuración y lee los PIDs.	
3. La aplicación móvil envía comandos OBD II en función de PIDs leídos en el anterior paso.	3.1. En caso de no cumplir con el formato establecido, aplicación móvil descarta mensaje recibido y vuelve a paso 1.
4. La aplicación móvil interactúa con el sistema OBD II y lee desde la ECU los valores de los sensores en función de comandos enviados en anterior paso.	4.1. Aplicación móvil captura error y espera al siguiente ciclo de lectura.
5. La aplicación móvil toma los valores de los sensores y lo almacena en memoria.	

Tabla 3.11 Caso de uso: Leer datos del dispositivo ELM327 (continuación)

Identificador: CU-04	
Nombre:	Consultar posición GPS.
Objetivo:	La aplicación móvil consulta la posición GPS
Precondiciones:	Tener almacenado el formato de respuesta con los valores de los sensores.
Postcondiciones:	La aplicación móvil almacena posición GPS.

Escenario básico	Escenario alternativo
1. La aplicación móvil establece conexión GPS con proveedores disponibles.	
2. La aplicación móvil obtiene la última posición conocida.	
3. La aplicación móvil recibe posición actual de proveedor GPS.	3.1. En caso de no recibir posición actual, aplicación móvil regresa a paso 1.
4. La aplicación móvil almacena posición GPS en memoria.	4.1. La aplicación móvil captura error y espera al siguiente ciclo de lectura.

Tabla 3.12 Caso de uso: Consultar posición GPS

Identificador: CU-05	
Nombre:	Codificar mensaje de respuesta con marcas de posición.
Objetivo:	La aplicación móvil codifica mensaje de respuesta en base a formato establecido por el subsistema servidor.
Precondiciones:	Posición GPS y formato de respuesta con los valores de los sensores almacenados en memoria.
Postcondiciones:	Mensaje de respuesta codificado de acuerdo a formato establecido por el subsistema servidor.

Escenario básico	Escenario alternativo
1. La aplicación móvil lee la posición GPS y los valores de los sensores almacenados en memoria.	

Tabla 3.13 Caso de uso: Codificar mensaje de respuesta

Identificador: CU-05	
2. La aplicación móvil concatena valores leídos en base a formato de banderas establecido.	
3. La aplicación móvil mantiene respuesta codificada en memoria de acuerdo a formato establecido por subsistema de recepción.	3.1. En caso de no recibir posición actual, la aplicación móvil regresa a paso 1.

Tabla 3.14 Caso de uso: Codificar mensaje de respuesta (continuación)

Identificador: CU-06	
Nombre:	Verificar conectividad GPRS.
Objetivo:	La aplicación móvil verifica conectividad GPRS.
Precondiciones:	Formato de respuesta almacenado en memoria.
Postcondiciones:	La aplicación móvil establece conexión GPRS.

Escenario básico	Escenario alternativo
1. La aplicación móvil detecta conectividad con red GPRS.	
2. Aplicación móvil establece conectividad con red GPRS.	2.1. Si no existe conectividad con red GPRS, la aplicación móvil guarda en la memoria el mensaje de respuesta codificado.

Tabla 3.15 Caso de uso: Verificar conectividad GPRS

Identificador: CU-07	
Nombre:	Almacenar el mensaje de respuesta codificado al no haber conexión GPRS.
Objetivo:	La aplicación móvil guarda mensaje de respuesta codificado en unidad de almacenamiento interno del dispositivo móvil.
Precondiciones:	Mensaje de respuesta codificado, almacenado en memoria correctamente. No existir conectividad con red GPRS.
Postcondiciones:	Mensaje de respuesta codificado guardado en la unidad de almacenamiento interno del Smartphone.

Escenario básico	Escenario alternativo
1. La aplicación móvil lee de la memoria el mensaje de respuesta codificado.	
2. La aplicación móvil establece la ruta de almacenamiento de mensaje de respuesta codificado.	
3. La aplicación móvil guarda el mensaje de respuesta codificado en la unidad de almacenamiento interno del Smartphone.	

Tabla 3.16 Caso de uso: Almacenar mensaje de respuesta codificado

Identificador: CU-08	
Nombre:	Enviar mensaje de respuesta codificado.
Objetivo:	La aplicación móvil envía el mensaje de respuesta codificado al subsistema servidor.

Tabla 3.17 Caso de uso: Enviar mensaje de respuesta codificado

Identificador: CU-08	
Precondiciones:	Mensaje de respuesta codificado, almacenado en la memoria o en la unidad de almacenamiento interno de Smartphone. Existir cobertura GPRS
Postcondiciones:	Mensaje de respuesta enviado usando conexión UDP.

Escenario básico	Escenario alternativo
1. La aplicación móvil lee de memoria o de unidad de almacenamiento interno de dispositivo móvil el mensaje de respuesta.	
2. La aplicación móvil establece parámetros de conexión UDP para enviar mensaje de respuesta.	
3. La aplicación móvil envía mensaje de respuesta codificado a subsistema de transmisión.	

Tabla 3.18 Caso de uso: Enviar mensaje de respuesta codificado (continuación)

3.1.4 REQUISITOS DE SOFTWARE

En esta sección se detallan los requisitos de software de la aplicación móvil que son obtenidos del análisis de requisitos de usuario.

Identificador: RSOP-01	
Título:	Reemplazar módulo dispositivo por Smartphone.
Descripción:	El subsistema automóvil deberá usar un Smartphone de bajo costo (alrededor de $80) que reemplazará al módulo dispositivo ubicado en el automóvil.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUR-01

Tabla 3.19 Requisito de software operacional: Reemplazar módulo dispositivo por Smartphone

Identificador: RSOP-02	
Título:	Tecnología de transmisión compatible.
Descripción:	El módulo de adquisición de datos del subsistema automóvil deberá soportar una tecnología de transmisión compatible con el Smartphone de bajo costo.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUR-02

Tabla 3.20 Requisito de software operacional: Tecnología de transmisión compatible

Identificador: RSIN-01	
Título:	Verificación de parámetros de configuración de aplicación móvil.
Descripción:	El usuario verifica los parámetros de configuración de la aplicación móvil mediante un botón.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUC-01

Tabla 3.21 Requisito de software de interfaz: Verificación de parámetros de configuración de aplicación móvil

Identificador: RSFU-01	
Título:	Ejecución inmediata de mensaje de configuración.
Descripción:	El subsistema automóvil deberá ejecutar el mensaje de configuración inmediatamente luego de haberlo recibido.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUC-02

Tabla 3.22 Requisito de software funcional: Ejecución inmediata de mensaje de configuración

Identificador: RSFU-02	
Título:	Almacenamiento de información en zonas sin cobertura GPRS.
Descripción:	El subsistema automóvil deberá almacenar el mensaje de respuesta con la información de los sensores en zonas en las que no exista cobertura GPRS.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUC-03

Tabla 3.23 Requisito de software funcional: Almacenamiento de información en zonas sin cobertura GPRS

Identificador: RSFU-03	
Título:	Integración de subsistemas.
Descripción:	El subsistema de recepción deberá integrarse al subsistema de transmisión del trabajo previo.
Prioridad:	Alta
Necesidad:	Esencial
Fuente:	RUR-03

Tabla 3.24 Requisito de software funcional: Integración de subsistemas

3.1.5 MATRIZ DE TRAZABILIDAD DE REQUISITOS DE USUARIO

	RUR-01	RUR-02	RUC-01	RUC-02	RUC-03	RUR-03
RSOP-01	X					
RSOP-02		X				
RSIN-01			X			
RSFU-01				X		
RSFU-02					X	
RSFU-03						X

Tabla 3.25 Matriz de trazabilidad de requisitos de usuario a requisitos de software

3.2 DISEÑO

En base al análisis realizado en la anterior sección, se llevará a cabo la fase de diseño de la aplicación móvil; aquí se describirá la arquitectura y el detalle de cómo va a ser implementada.

3.2.1 ARQUITECTURA DE LA APLICACIÓN MÓVIL

En la Figura 3.2 se muestra la arquitectura del subsistema automóvil propuesto.

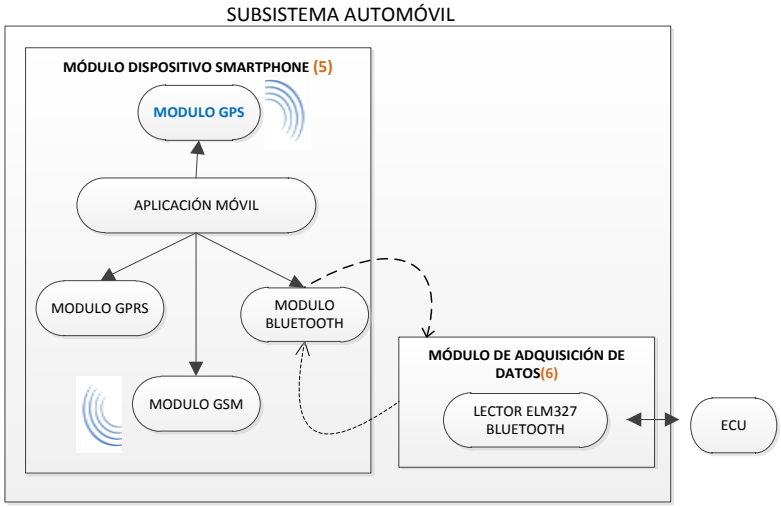

Figura 3.2 Arquitectura del subsistema automóvil

3.2.1.1 MÓDULO DISPOSITIVO SMARTPHONE

Constituye el principal cambio en relación al trabajo previo [4] ya que uno de los objetivos del presente proyecto de titulación es reemplazar los dispositivos: Arduino Mega 2560, ZigBee, Shield GSM/GPRS SIM 900 y Shield GPS 2.0

El dispositivo Android tiene la siguientes interfaces de conexión:

- GPS, permite reemplazar al dispositivo Shield GPS 2.0 y es el encargado de obtener marcas de localización que acompañan a las mediciones.
- GPRS, permite reemplazar al dispositivo Shield GSM/GPRS SIM 900 y es usado para enviar los datos leídos desde la ECU del automóvil al subsistema servidor.
- GSM, permite reemplazar al dispositivo Shield GSM/GPRS SIM 900 y es usado para recibir el mensaje de configuración vía SMS que es enviado por el subsistema servidor.
- Bluetooth, permite reemplazar al dispositivo ZigBee y es usado para establecer conexión con el módulo de adquisición de datos que se encuentra conectado a la interfaz DLC del automóvil. A través de este módulo se envían al módulo de adquisición de datos los comandos OBD II que son recibidos desde subsistema servidor, adicionalmente este módulo se encarga de recibir los valores de los sensores leídos desde la ECU en función del comando OBD II enviado.

3.2.1.2 MÓDULO DE ADQUISICIÓN DE DATOS

Este módulo está formado por el lector ELM327 tipo Bluetooth, está conectado a la interfaz DLC del automóvil y permite establecer comunicación entre la ECU y el módulo dispositivo Smartphone vía Bluetooth. En la Figura 3.3 se muestra el lector ELM327 tipo Bluetooth.

Figura 3.3 Lector ELM327 tipo Bluetooth [3]

3.2.2 DIAGRAMAS DE ACTIVIDADES

Una vez concluida la fase de análisis de requisitos, con la información de los casos de uso, y definida la arquitectura del subsistema automóvil se procede a elaborar los diagramas de actividades de la aplicación móvil. En las siguientes figuras se muestran los diagramas de actividad de la aplicación móvil a desarrollarse.

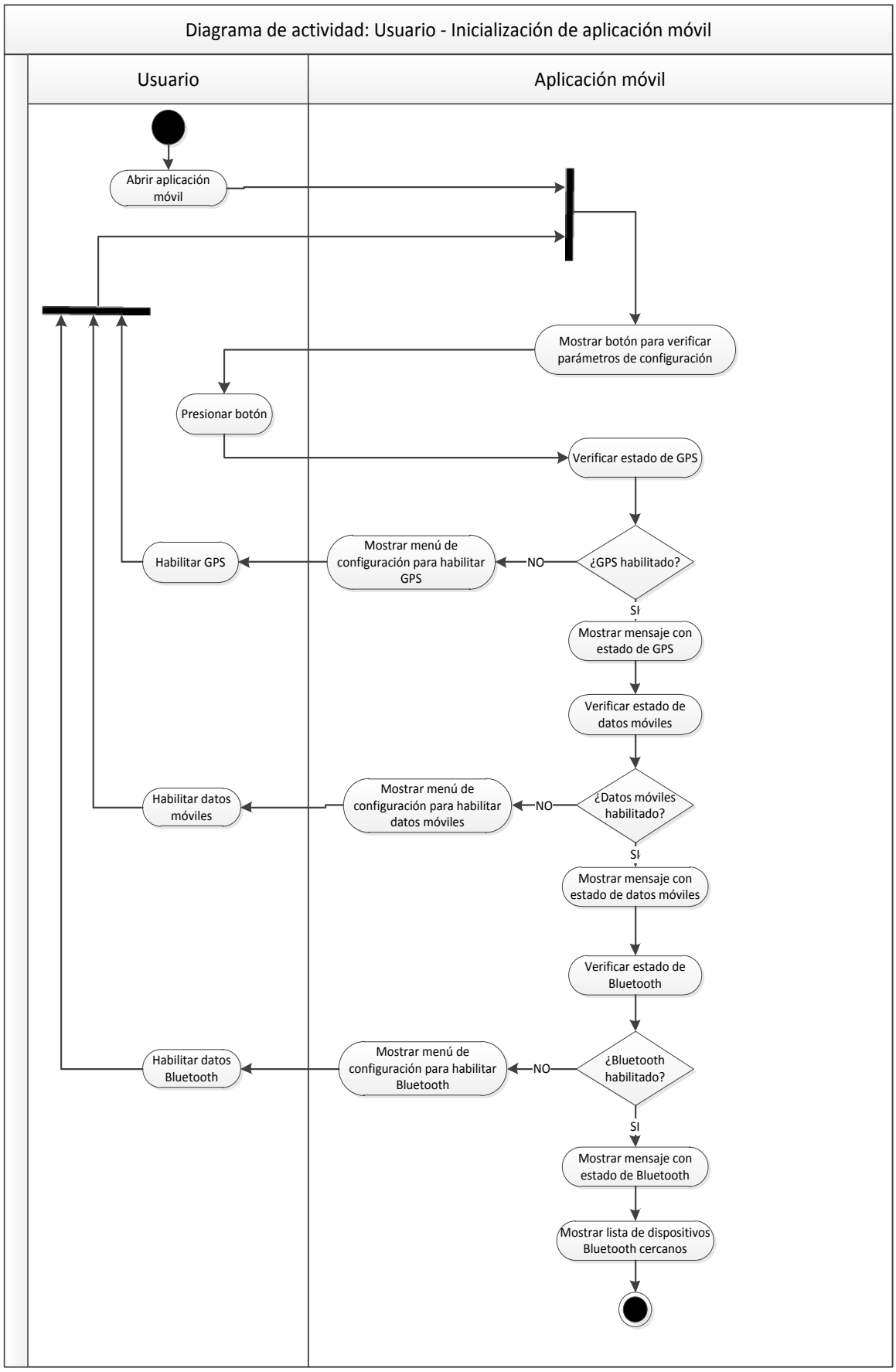

Figura 3.4 Diagrama de actividad: Usuario – Inicialización de aplicación móvil

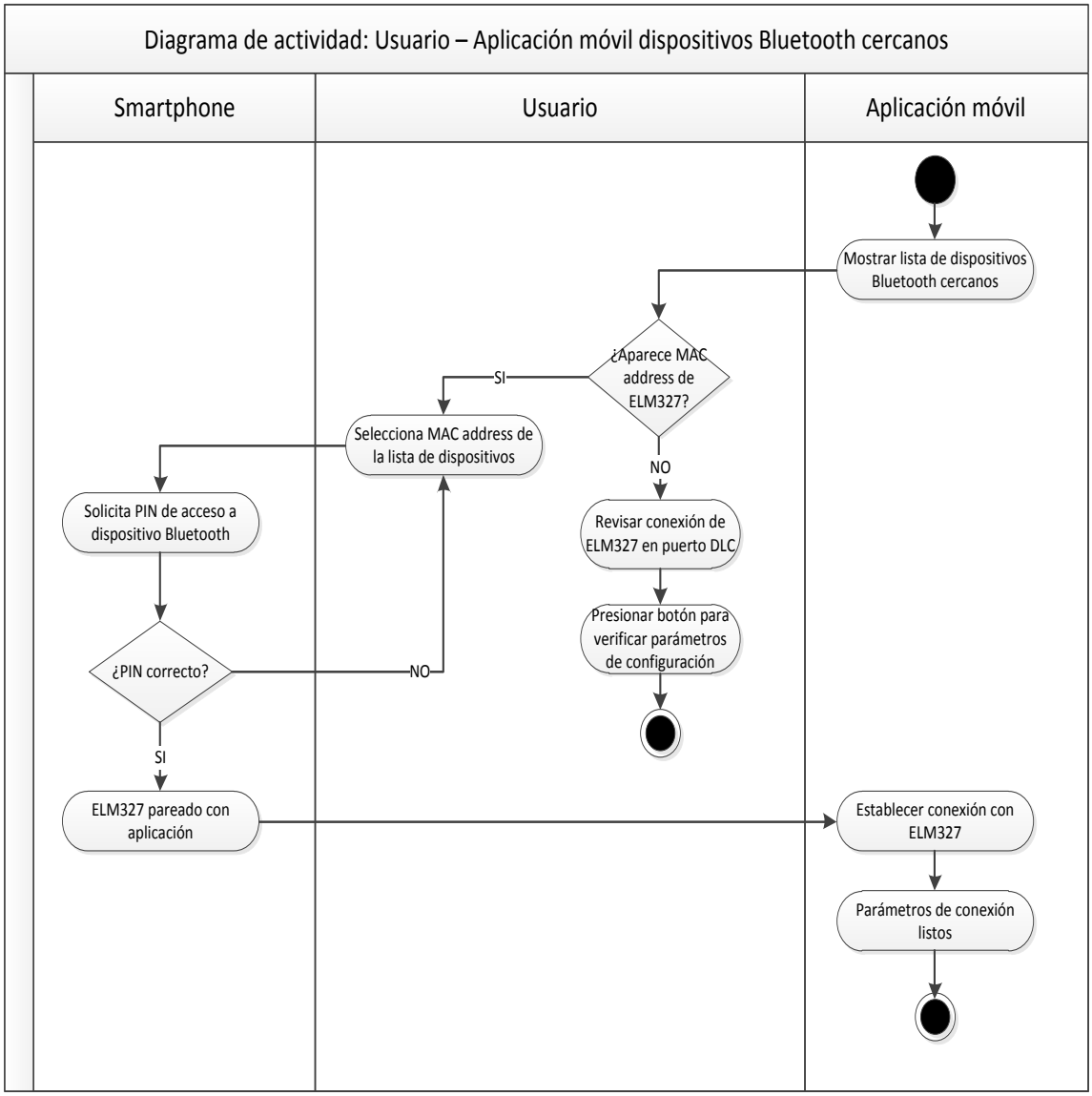

Figura 3.5 Diagrama de actividad: Usuario – Aplicación móvil dispositivos Bluetooth cercanos

En la

Figura 3.6 Diagrama de actividad: Subsistema de transmisión – Aplicación móvil decodificación de mensaje de configuración

se muestra el proceso para ejecución inmediata del mensaje de configuración que llega por SMS. Se debe tomar en cuenta que el sistema operativo Android tiene una clase llamada *BroadcastReceiver* que al implementarla en una aplicación puede detectar cuando un evento ocurre y tomar una acción inmediata. Una vez que se recibe el mensaje de configuración vía SMS se interrumpe el proceso e inmediatamente se ejecutan las acciones del siguiente diagrama de actividad.

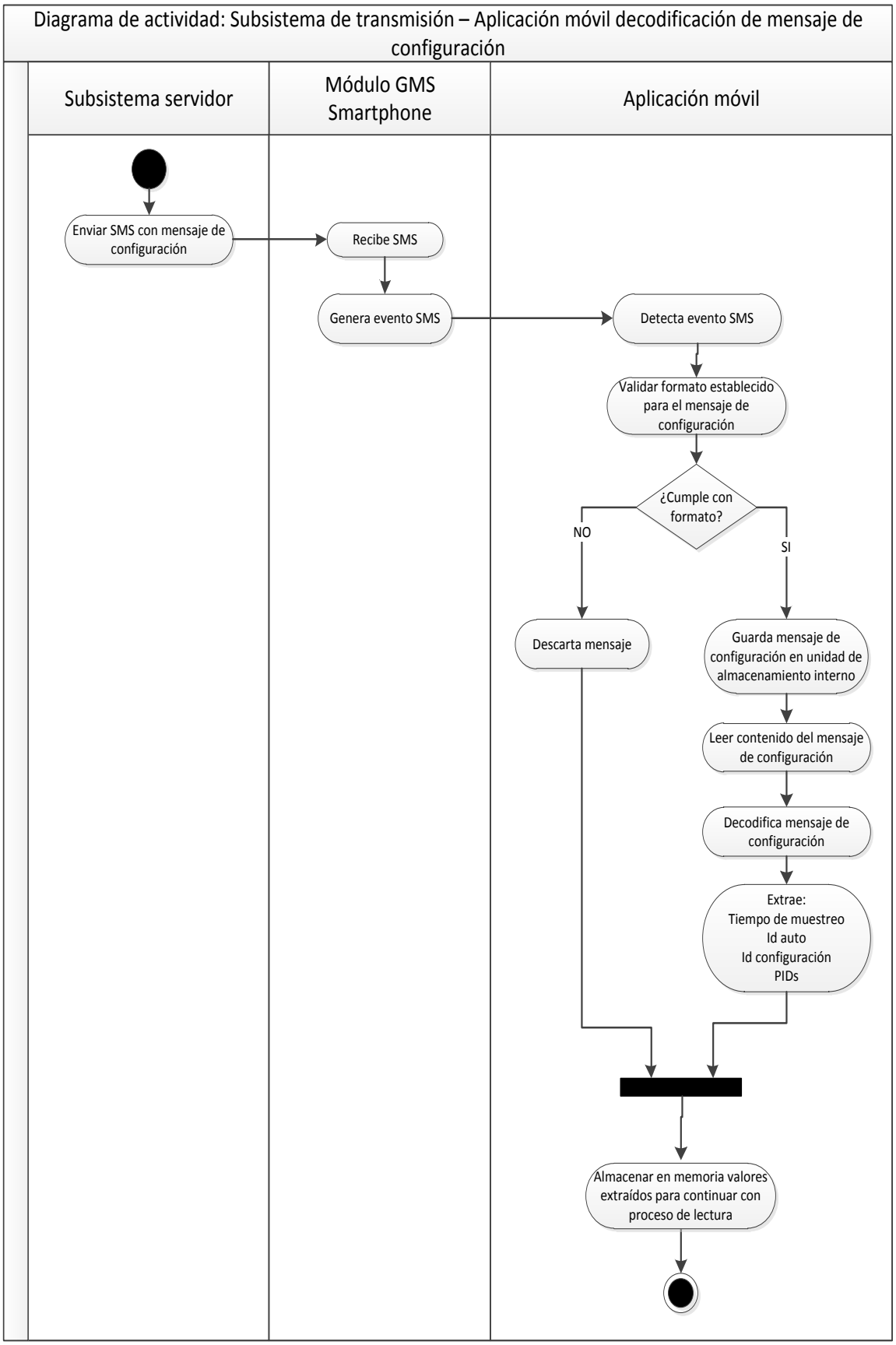

Figura 3.6 Diagrama de actividad: Subsistema de transmisión – Aplicación móvil decodificación de mensaje de configuración

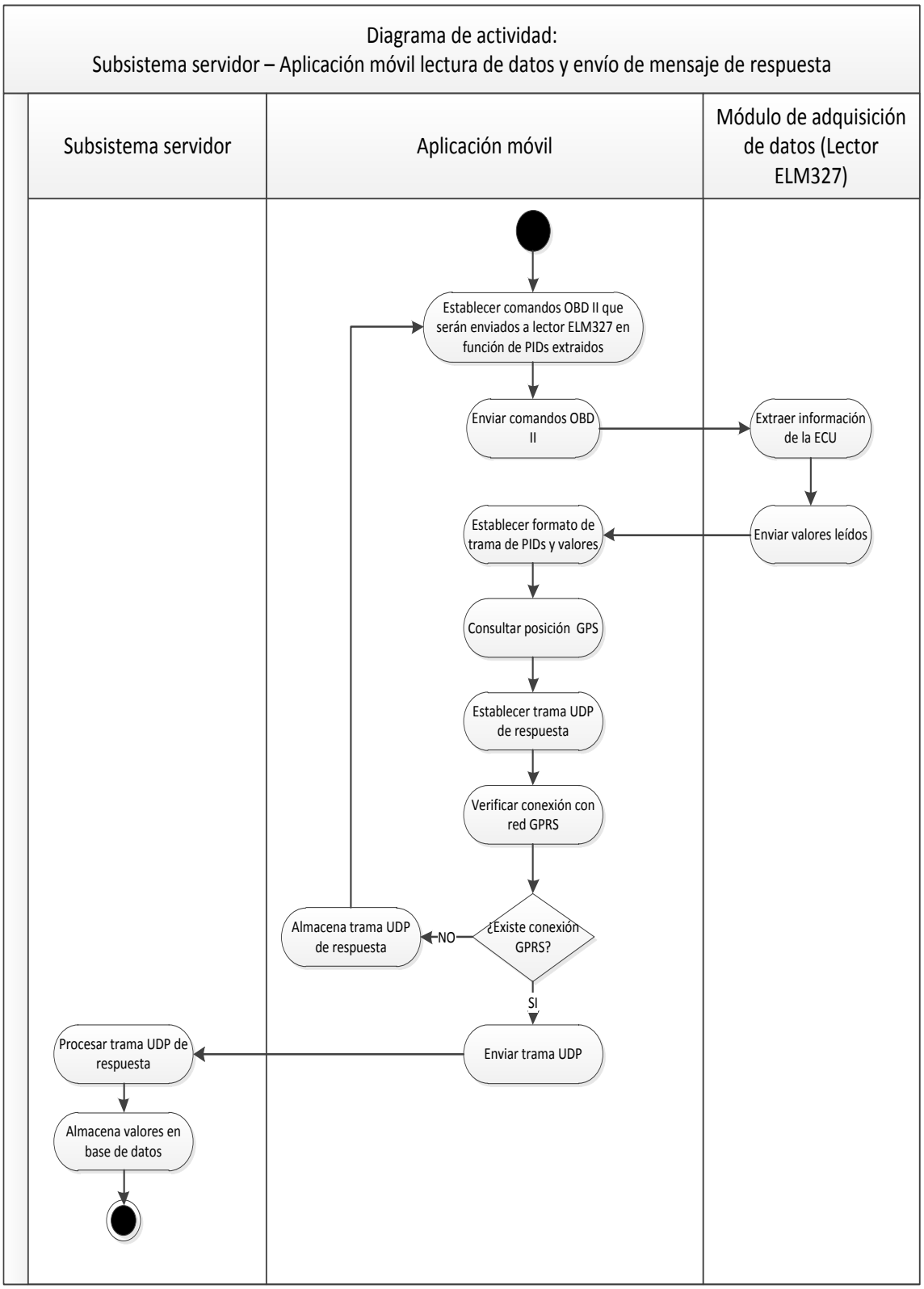

Figura 3.7 Diagrama de actividad: Subsistema servidor – Aplicación móvil lectura de datos y envío de mensaje de respuesta

3.2.3 DIAGRAMA DE CLASES

La Figura 3.8 muestra las clases con las que se desarrollará la aplicación móvil.

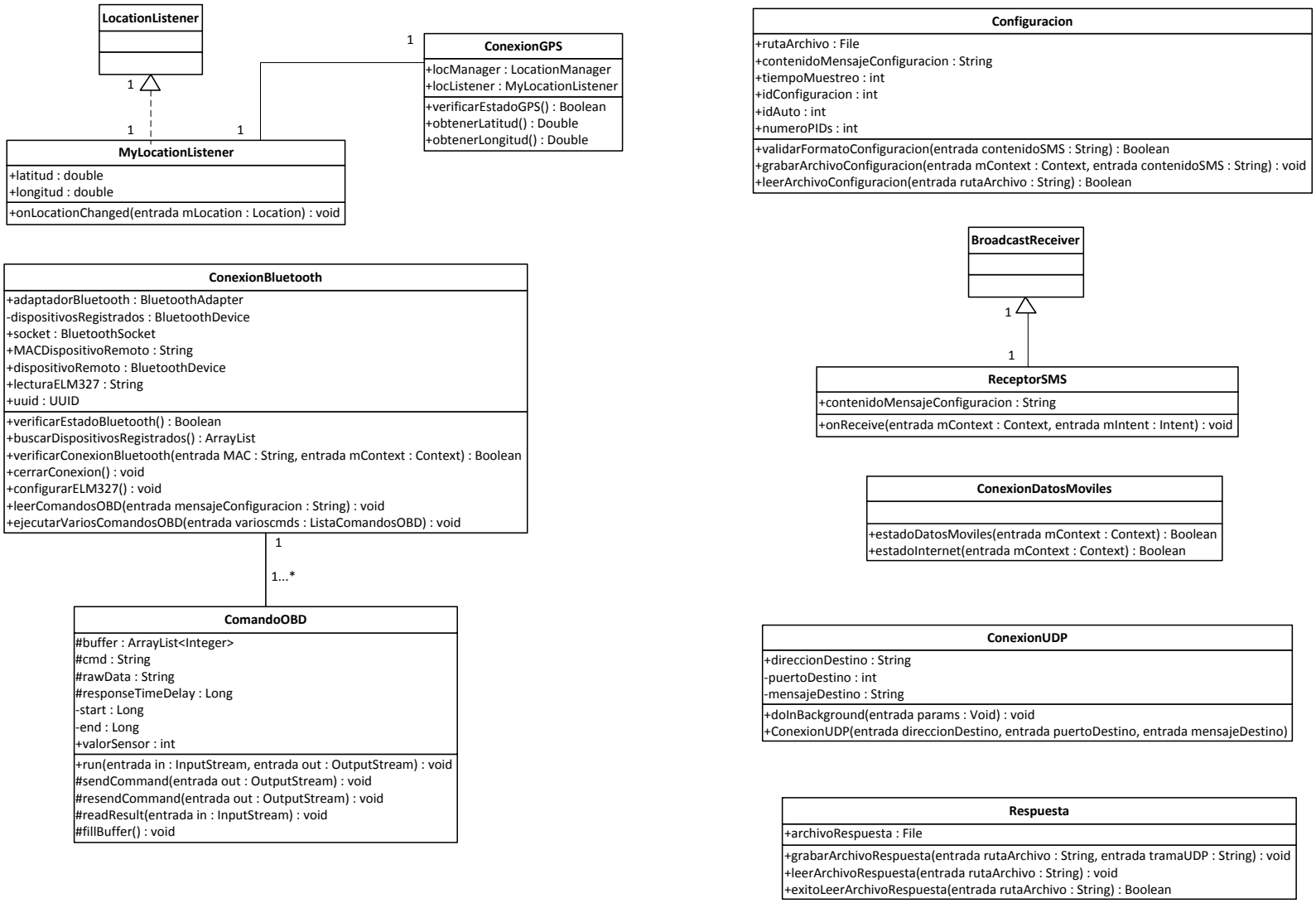

Figura 3.8 Diagrama de clases aplicación móvil

3.2.3.1 DEFINICIÓN DE CLASES

- *Clase MyLocationListener:* implementa la interfaz *LocationListener* que tiene Android y permite obtener la latitud y longitud cuando se detecta un cambio de posición luego de establecer una conexión con un satélite GPS.
- *Clase ConexionGPS:* representa el módulo GPS del Smartphone y mediante el uso de la *Clase MyLocationListener*, se obtiene la latitud y longitud; adicionalmente permite verificar si el GPS está habilitado en el Smartphone.
- *Clase ConexionBluetooth:* representa el módulo Bluetooth del Smartphone , permite establecer, configurar y cerrar conexión Bluetooth con el módulo de adquisición de datos. Una vez establecida la conexión, permite crear una lista de comandos OBD II y enviarlos módulo de adquisición de datos, luego de este proceso; se encarga de recibir los valores de los sensores.
- *Clase ComandoOBD:* representa el comando OBD II que será enviado al módulo adquisición de datos y permite ejecutar uno a uno los comandos OBD II de la lista de comandos OBD II. Además en esta clase se capturan los errores que pueden generarse al momento de leer la información de los sensores.
- *Clase ListaComandosOBD:* representa una lista de comandos y permite establecer una primera trama con los PIDs y los valores de los sensores leídos desde el módulo de adquisición de datos.
- *Clase ConexiondatosMoviles:* representa el módulo GPRS del Smartphone, permite verificar si la opción de datos móviles del Smartphone se encuentra habilitada y si existe conexión GPRS.
- *Clase Configuración:* representa la configuración de lectura que se va a procesar, permite verificar si los mensajes de configuración cumplen el formato establecido con la finalidad de grabarlos en la unidad de almacenamiento interna, adicionalmente lee los archivos de configuración desde la unidad de almacenamiento interna.
- *Clase BroadcastReceiver:* permite detectar un evento SMS y tomar una acción inmediatamente. Mediante el uso de esta clase se notifica a la *Clase ReceptorSMS* y *MainActivity* la llegada de un SMS.

- *Clase ReceptorSMS:* extiende de la clase BroadcastReceiver e implementa el método onReceive con la finalidad de ejecutar inmediatamente la grabación del mensaje de configuración una vez recibido el SMS.
- *Clase ConexionUDP:* representa la conexión entre subsistemas, permite enviar la trama de respuesta al subsistema servidor mediante una conexión UDP.
- *Clase Respuesta:* representa el mensaje de respuesta que será enviado al subsistema servidor, permite capturar la información que será enviada al subsistema servidor mediante una conexión UDP.

  En caso de no existir conexión GPRS esta clase es la encargada de grabar el archivo de respuesta que será leído al recuperar la conexión GPRS.
- Finalmente la clase principal que usa las instancias de las anteriores clases definidas, es la *Clase MainActivity* que muestra al usuario en pantalla el botón para verificar parámetros de configuración y otro para cerrar la aplicación móvil.

  Mediante el primer botón se puede verificar que GPS, Bluetooth y Datos Móviles se encuentren habilitados correctamente. Mientras no se encuentren habilitados correctamente, al aplicación no mostrará la MAC address del lector ELM327.

3.3 CODIFICACIÓN

3.3.1 MIDDLE GAME

Una vez definidas las clases que serán implementadas en la aplicación móvil, se procede a codificar todas las clases y métodos de la aplicación móvil.

A continuación se mencionan los elementos usados en esta fase:

- Entorno de programación: Android Studio versión1.4.1
- Lenguaje de programación: Java
- Versión mínima del sistema operativo en el Smartphone: Android 4.0.3

La Figura 3.9 muestra la conformación de equipos para el proceso middle game.

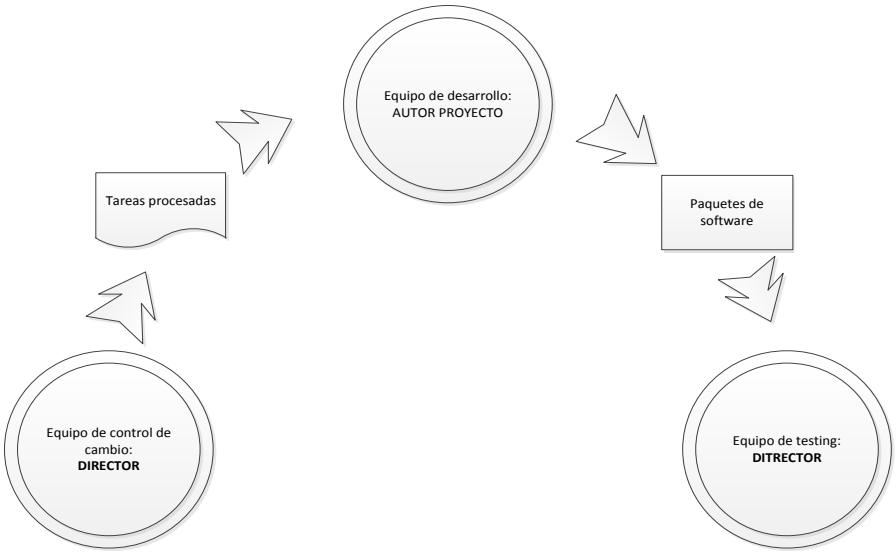

Figura 3.9 Proceso middle game – conformación de equipos

En esta fase se mantuvo reuniones semanales con el director de tesis para la revisión de las clases y su correcto funcionamiento. En el Anexo 1 se encuentra el código fuente completo de la aplicación móvil desarrollada. Sin embargo a continuación se revisa el código fuente de Clase ReceptorSMS ya que con ella se logró implementar la recepción del mensaje SMS y su procesamiento de manera asincrónica.

En la Tabla 3.26 y Tabla 3.27 se muestra el código fuente que permite detectar un evento SMS, ejecutar inmediatamente la grabación del archivo de configuración y notificar a la clase MainAtivity el evento. Asimismo, se revisa la codificación de la comprobación de conexión GPRS y el almacenamiento del archivo de respuesta en la unidad de almacenamiento interna del Smartphone debido a que es importante porque esta información deberá ser enviada al subsistema servidor al recuperar conexión GPRS.

```
public class ReceptorSMS extends BroadcastReceiver {
    //Objeto que permitirá grabar mensaje de configuración
    Configuracion mConfiguracion;
    //String que guarda el contenido de configuracion recibido en SMS
entrante
    String contenidoMensajeConfiguracion = "";

    //Método que permite detectar la llegada de un SMS
    @Override
    public void onReceive(Context mContext, Intent mIntent) {
```

Tabla 3.26 Clase Receptor SMS – detección de evento SMS, grabación de mensaje de configuración y notificación de evento a clase MainActivity

```
        //Instanciar mConfiguracion
mConfiguracion = new Configuracion();
        //Permite pasar contenidoMensajeConfiguracion entre actividades
        Bundle bundle = mIntent.getExtras();
        //Almacenar el contenido del SMS entrante
        SmsMessage[] smsEntrante = null;
        //String contenidoMensajeConfiguracion = "";
        if(bundle != null){
            // Obtener el array de estructuras pdus
            Object[] pdus = (Object[])bundle.get("pdus");
            smsEntrante = new SmsMessage[pdus.length];
            for(int i = 0; i<smsEntrante.length; i++){
                // Generar mensajes smsEntrante
                smsEntrante[i] =
SmsMessage.createFromPdu((byte[])pdus[i]);
                contenidoMensajeConfiguracion +=
smsEntrante[i].getMessageBody().toString();
            }
//Verificar el formato del mensaje de configuración
        boolean exitoFormatoConfiguracion;
        exitoFormatoConfiguracion =
mConfiguracion.validarFormatoConfiguracon(contenidoMensajeConfiguracion
);
        if(exitoFormatoConfiguracion == true){
            try {
                //Grabar contenido del SMS en archivo de configuración
                mConfiguracion.grabarArchivoConfiguracion(mContext,
contenidoMensajeConfiguracion);
                //Instanciamos Intent que servirá para enviar el
contenido del SMS a clase principal
                Intent broadcastIntent = new Intent();
                //Configurar una palabra clave para el Intent, en este
caso "CONFIGURACION_GRABADA"
                //esta palabra clave deberá ser configurada en clase
principal
                broadcastIntent.setAction("CONFIGURACION_GRABADA");
                //Enviar contenido de configuración a clase principal
                broadcastIntent.putExtra("config_grabada",
contenidoMensajeConfiguracion);
                //Enviar el Broadcast a clase principal
                mContext.sendBroadcast(broadcastIntent);
            } catch (IOException e) {
                //En caso de no ejecutarse el Broadcastreceiver
capturamos la excepción
                Log.d("tag","Error al grabar configuracion:
"+e.toString());
                e.printStackTrace();
            }
        }else{
            //Si mensaje de configuración no cumple el formato
establecido
            Log.d("tag","Mensaje de configuración no cumple
formato:"+exitoFormatoConfiguracion);
        }
    }
}
```

Tabla 3.27 Clase Receptor SMS – detección de evento SMS, grabación de mensaje de configuración y notificación de evento a clase MainActivity (continuación)

En la Tabla 3.28 se muestra un fragmento de la clase MainActivity en el que se procede a leer inmediatamente el archivo de configuración, obtener el tiempo de muestreo y ejecutar iterativamente el proceso de lectura de valores de los sensores de la ECU.

```
private BroadcastReceiver mBroadcastReceiver = new BroadcastReceiver()
{
    @Override
    public void onReceive(Context mContext, Intent mIntent) {

        //Mensaje de configuración es almacenado en
contenidoMensajeConfiguracion
        contenidoConfiguracion =
mIntent.getExtras().getString("config_grabada");
        Log.d("tag", "Contenido configuracion en main: "+
contenidoConfiguracion);
        boolean exitoLeerArchivoConfiguracion;

        exitoLeerArchivoConfiguracion =
mConfig.leerArchivoConfiguracion("/storage/emulated/0/obd2/configuraci
on.txt");
        //Si la conexión con ELM327 ya se encuentra establecida y
obteniendo información
        //es necesario detener el timmer para evitar duplicidad de
información
        //si mTimer es null significa que es la primera ejecución del
timer
        if (mTimer == null){

            if(exitoLeerArchivoConfiguracion == true){
                int valorMuestreo = mConfig.getTiempoMuestreo();
                int tiempoRepeticion =
calcularTiempoRepeticion(valorMuestreo);
                //Leer iterativamente los valores de PIDs desde el
ELM327
                leerIterativamente(tiempoRepeticion);
            }

        }else{
            //Si mTimer es diferente de null significa que existe un
timer ejecútandose y es necesario cancelarlo
            //esto ayuda a no duplicar lectura de valores PIDs
            mTimer.cancel();
            if(exitoLeerArchivoConfiguracion == true){
                int valorMuestreo = mConfig.getTiempoMuestreo();
                int tiempoRepeticion =
calcularTiempoRepeticion(valorMuestreo);
                //Leer iterativamente los valores de PIDs desde el
ELM327
                leerIterativamente(tiempoRepeticion);
            }
        }
    }
};
```

Tabla 3.28 Fragmento de código Clase MainActivity – implementación de evento SMS

En la Tabla 3.29 se muestra el fragmento de código de la clase MainActivity que permite detectar si existe conexión GPRS, en caso de no existir, graba el archivo de respuesta y al recuperar conexión lee el archivo para enviarlo al subsistema servidor mediante conexión UDP.

```
//Verificar si existe Internet Móvil
boolean exitoConexionInternet =
cxDatosMoviles.estadoInternet(getApplicationContext());
//Obtener la posición del PID 0C que permite comprobar si el auto esta
apagado
int posicion = cxBluetooth.getLecturaELM().indexOf("0C");
//Obtener el primer dígito del valor del PID 0C. Es necesario recorrer
3 posiciones para obtener dicho valor
String temp = cxBluetooth.getLecturaELM().substring(posicion + 3,
posicion + 4);
int rpm = Integer.parseInt(temp);
Log.d("tag","lecturaelm valorSensor:"+temp);
if (exitoConexionInternet == true && rpm > 0) {
    //Validar si existe archivo respuesta.txt
    boolean exitoLeerArchivoRespuesta =
mRespuesta.exitoLeerArchivoRespuesta("/storage/emulated/0/obd2/respuest
a.txt");
    Log.d("tag", "exitoLeerArchivoRespuesta:" +
exitoLeerArchivoRespuesta);
    if (exitoLeerArchivoRespuesta == true) {

mRespuesta.leerArchivoRespuesta("/storage/emulated/0/obd2/respuesta.txt
");
    } else {
        Log.d("tag", "lecturaELM" + cxBluetooth.getLecturaELM());
        String tramaUDP =
inicioTrama+cxBluetooth.getLecturaELM()+finTrama;
        Log.d("tag","lecturaelm trama con internet:"+tramaUDP);
        cxUDP = new ConexionUDP(IPDestino, puertoDestino, tramaUDP);
        cxUDP.execute();
    }
} else if (rpm == 0){
    Log.d("tag","lecturaelm auto detenido:"+rpm);
}else{
    String tramaUDP = inicioTrama+cxBluetooth.getLecturaELM()+finTrama;

//mConfig.grabarArchivoRespuesta("/storage/emulated/0/obd2/respuesta.tx
t", tramaUDP + "\n");

mRespuesta.grabarArchivoRespuesta("/storage/emulated/0/obd2/respuesta.t
xt", tramaUDP + "\n");
    Log.d("tag", "lecturaELM trama sin internet:" + tramaUDP);
}
```

Tabla 3.29 Fragmento de código Clase MainActivity – detectar conexión GPRS y grabar y leer archivo de respuesta

CAPÍTULO 4

4 PRUEBAS DE FUNCIONAMIENTO

Como se mencionó en el capítulo 1, la fase de pruebas se la realizará mediante el proceso *late game* de la metodología ágil hibrida para desarrollar software en dispositivos móviles [22].

4.1 PROCESO LATE GAME

La Figura 4.1 muestra la conformación de equipos para el proceso late game.

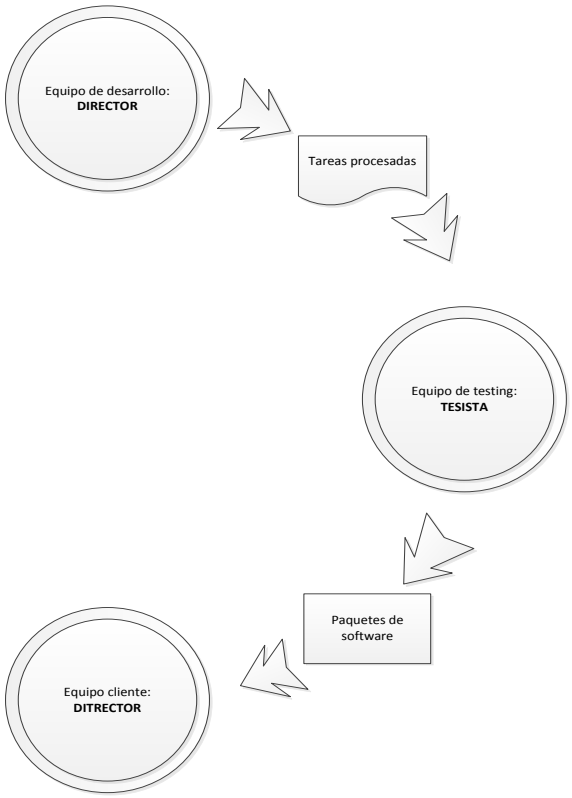

Figura 4.1 Proceso late game – conformación de equipos

Una vez definidos los equipos para el proceso late game, se describe la infraestructura desplegada para los escenarios de pruebas, los escenarios que permiten probar la funcionalidad de la aplicación móvil y los requisitos de usuario que fueron satisfechos.

La Figura 4.2 muestra la infraestructura desplegada para las pruebas.

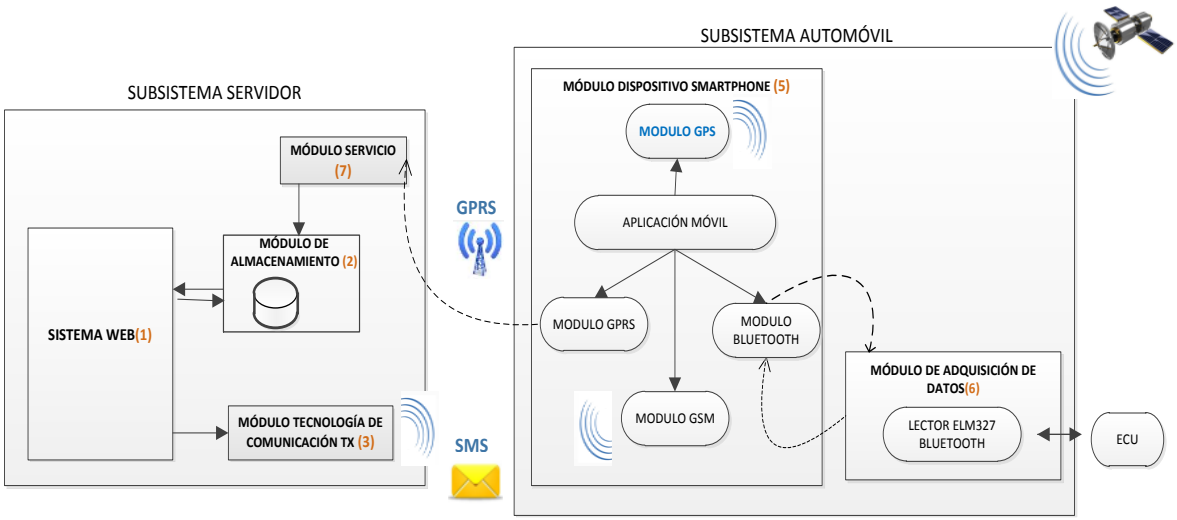

Figura 4.2 Infraestructura desplegada para las pruebas

Características Subsistema servidor

- Marca: Dell
- Procesador: Core i7
- Memoria RAM:16 GB
- Sistema Operativo: Centos versión 6
- Apache versión 2.2.31
- Celular Nokia 300 con paquete SMS habilitado

Características Subsistema automóvil

- Smartphone Samsung Ace 3, características detalladas en la sección 2.3.5.3
- SIM card con plan de datos habilitado
- Lector ELM327 Bluetooth
- Aplicación móvil instalada en Smartphone
- Automóvil Chevrolet Aveo Emotion

4.2 PRUEBAS PRELIMINARES, VERIFICAR ESTADO DE CONFIGURACIONES

Una vez desplegada la infraestructura de la Figura 4.2 y con el fin de verificar los mensajes de notificación que muestra la aplicación móvil. La Figura 4.3 permite

observar los mensajes de notificación cuando GPS, Bluetooth y Datos Móviles se encuentran habilitados correctamente.

Cuando hay alguna interfaz no habilitada o hay algún problema, la aplicación móvil solicita al usuario la activación de GPS, Datos Móviles y Bluetooth. La Figura 4.4 muestra los menús para habilitar las interfaces que así lo requieran.

Procedimiento de usuario:

1. Instalar aplicación móvil en Smartphone Samsung Ace 3.
2. Abrir aplicación móvil.
3. Presionar botón Verificar parámetros configuración.

Resultados obtenidos:

Figura 4.3 Mensajes de notificación - parámetros de configuración habilitados

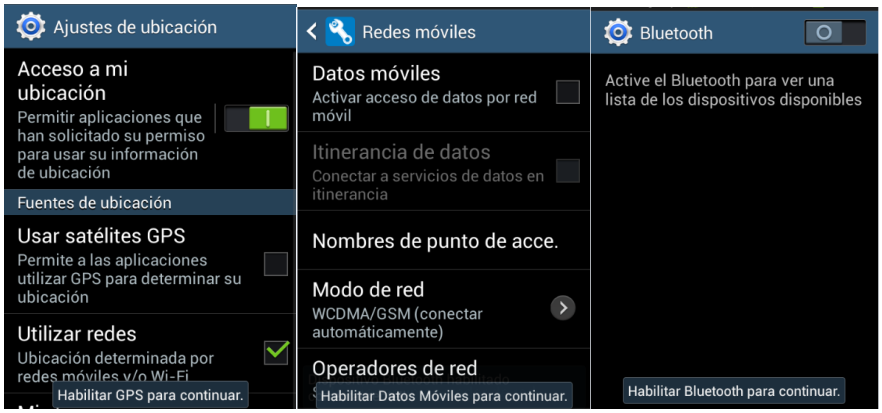

Figura 4.4 Mensajes de notificación - parámetros de configuración deshabilitados

Requisitos de usuario cumplidos:

- RUC-01: Verificar parámetros de configuración.

4.3 PRUEBA DE RECEPCIÓN DE MENSAJE DE CONFIGURACIÓN Y EJECUCIÓN INMEDIATA

En este caso se considera que existe un archivo de configuración grabado en el Smartphone, el tiempo de muestreo es 1 minuto y existe cobertura GPRS. En este escenario, la aplicación móvil está enviando valores de los sensores leídos desde la ECU cada minuto al subsistema servidor. En el momento de recibir el mensaje de configuración vía SMS, la aplicación móvil deberá interrumpir el proceso y ejecutar inmediatamente el envío de datos de acuerdo al nuevo mensaje de configuración.

Procedimiento de usuario:

- Abrir aplicación móvil.
- Establecer conexión con ELM327.
    - La aplicación comienza a enviar información al subsistema servidor, considerando la configuración inicial que está en un archivo.
- El usuario crea una nueva configuración y la envía desde el subsistema servidor.
    - La aplicación suspende el proceso de lectura en curso, guarda la nueva configuración recibida en un archivo y la usa en el proceso de lectura.

Resultados obtenidos:

La Figura 4.5 muestra el mensaje de configuración almacenado en el Smartphone y su contenido.

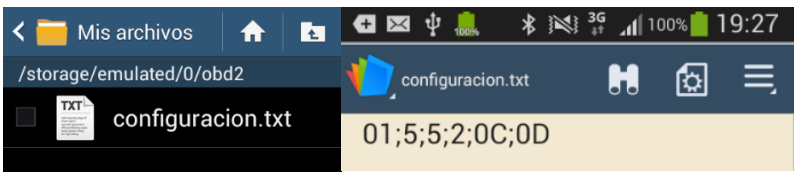

Figura 4.5 Mensaje de configuración almacenado en Smartphone y contenido

La Figura 4.6 muestra los datos que se leen desde la ECU con la configuración actual cada minuto, que son generados por logs de la aplicación móvil y capturados en Android Studio.

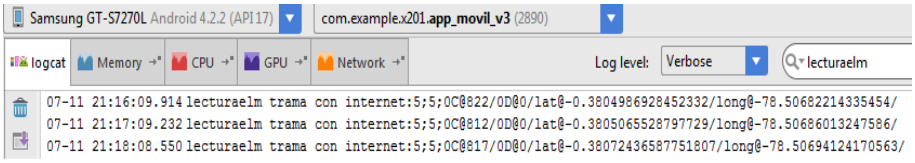

Figura 4.6 Datos leídos con configuración inicial - Logs generados por aplicación móvil

La Figura 4.7 muestra los datos que son recibidos en el subsistema servidor cada minuto en base a configuración inicial.

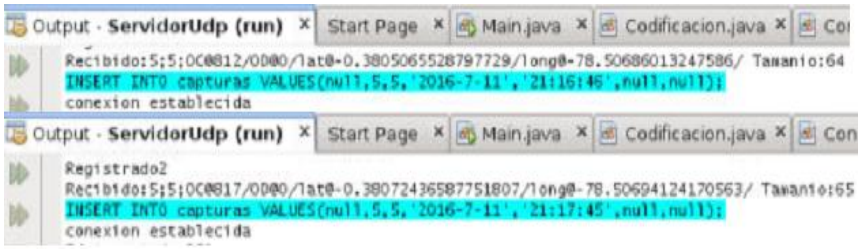

Figura 4.7 Datos recibidos en subsistema servidor con configuración inicial

En la Figura 4.8 muestra la llegada de un nuevo mensaje de configuración desde el subsistema servidor, suspensión del proceso de lectura en curso, ejecución inmediata y envío de información en base a nueva configuración.

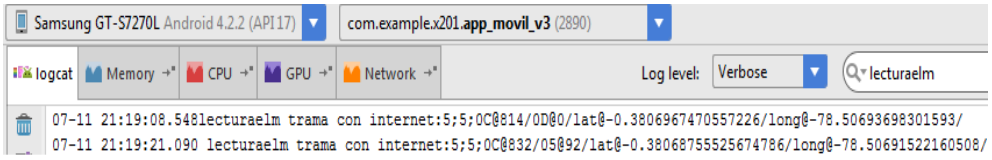

Figura 4.8 Suspensión de lectura en curso y ejecución inmediata de nueva configuración

La Figura 4.9 muestra el contenido del nuevo archivo de configuración.

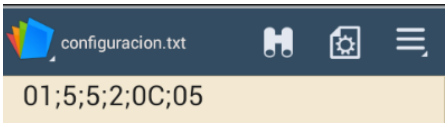

Figura 4.9 Contenido nuevo archivo de configuración

La Figura 4.10 muestra desde el lado del subsistema servidor la suspensión del actual proceso de medición y recepción inmediata de datos en función de nueva configuración.

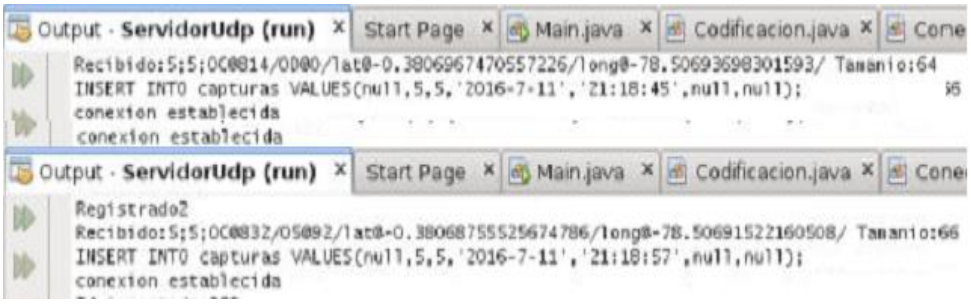

Figura 4.10 Ejecución inmediata de nueva configuración y recepción de datos en subsistema servidor

Requisitos de usuario cumplidos:

- RUR-01: Reemplazar módulo dispositivo por Smartphone.
- RUR-02: Tecnología de transmisión compatible.
- RUC-02: Ejecución inmediata de mensaje de configuración.
- RUR-03: Integración de subsistemas.

4.4 PRUEBA SIN COBERTURA GPRS

En este caso se considera que la aplicación está enviando los valores de los sensores leídos desde la ECU cada minuto al subsistema servidor. El subsistema servidor atraviesa por una zona sin cobertura GPRS, la aplicación detecta conexión GPRS fallida y graba el resultado en archivo de respuesta. Una vez que recupera conexión GPRS, la aplicación móvil envía el contenido del archivo de respuesta a subsistema servidor y continúa proceso de lectura.

Procedimiento de usuario:

- Abrir aplicación móvil.
- Establecer conexión con ELM327.
    - La aplicación comienza a enviar información al subsistema servidor, considerando la configuración inicial que está en un archivo.
- Se deshabilita los datos móviles manualmente del menú del Smartphone.
    - Los datos se almacenan en archivo de respuesta temporal.
- Se habilita datos móviles manualmente.
    - Los datos del archivo temporal se envían hacia el subsistema servidor.
    - El proceso de lectura y envío de información continúa normalmente.

Resultados obtenidos:

La Figura 4.11 muestra el envío de datos a subsistema servidor cada minuto considerando la configuración inicial.

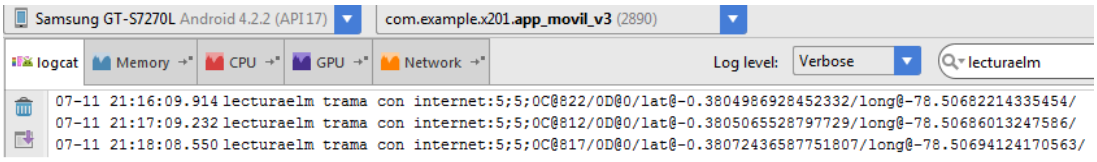

Figura 4.11 Envío de información a subsistema servidor considerando configuración inicial

La Figura 4.12 muestra el archivo de respuesta y su contenido al momento de deshabilitar datos móviles manualmente.

Figura 4.12 Archivo de respuesta y contenido – datos móviles deshabilitados

La Figura 4.13 muestra los logs generados por aplicación móvil mientras datos móviles están deshabilitados.

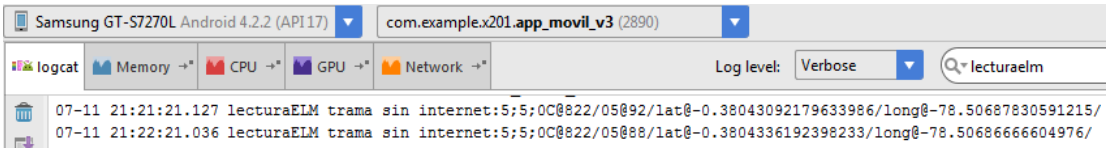

Figura 4.13 Logs de aplicación móvil en subsistema automóvil al deshabilitar datos móviles

La Fifura 4.14 muestra la recepción de los datos del archivo temporal de respuesta en el subsistema servidor y la ejecución normal del proceso de lectura.

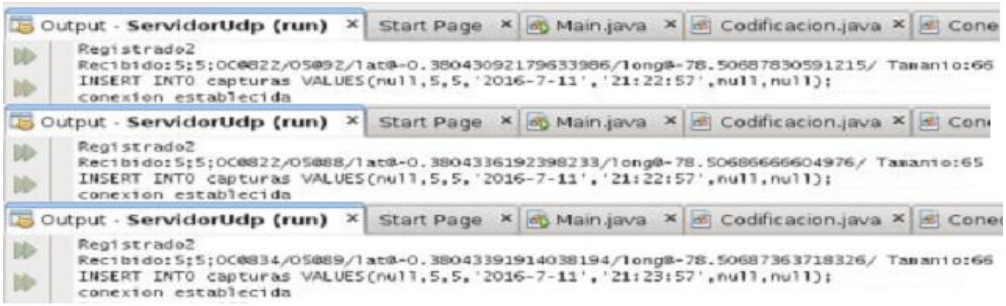

Figura 4.14 Recepción de datos de archivo temporal de respuesta en subsistema servidor

Requisitos de usuario cumplidos:

- RUC-03: Almacenamiento de información en zonas sin cobertura GPRS.

Con la finalidad de verificar el espacio máximo que ocuparía el archivo de respuesta cuando no existe cobertura GPRS, se toman las siguientes consideraciones:

- Tiempo que el automóvil estaría en marcha y sin cobertura GPRS: 24 horas
- Tiempo de muestreo mínimo: 1 minuto
- Tamaño del archivo de respuesta almacenado cuando no existe cobertura GPRS: 66 bytes por cada trama de respuesta
- Total de minutos: 1440 minutos
- Total de bytes en archivo de respuesta almacenado: 95040 bytes

Total de MBytes en archivo de respuesta almacenado: 92,8125 MBytes

CAPÍTULO 5

5 ANÁLISIS DE COSTOS

Debido a que el objetivo del presente proyecto de titulación fue reemplazar el subsistema automóvil del trabajo previo [4], en este capítulo se considera exclusivamente los costos de este subsistema.

5.1 COSTOS DE IMPLEMENTACIÓN DE LA SOLUCIÓN

A continuación se calculan todos los costos en los que se han incurrido en este proyecto.

5.1.1 COSTOS DIRECTOS

Los costos directos corresponden a la mano de obra y los materiales utilizados para la implementación de arquitectura del presente proyecto de titulación.

La Tabla 5.1 muestra el detalle de los costos generados.

Descripción	Cantidad	Costo Unitario	Costo Total
Smartphone Samsung Ace 3	1	$ 70,00	$ 70,00
Tarjeta SIM	1	$ 3,50	$ 3,50
Lector ELM327 Bluetooth	1	$ 25,00	$ 25,00
Plan de datos Smartphone Básico, incluye 500MB	1	$ 15,00	$ 15,00
Horas de trabajo	240	$ 5,00	$ 1.200,00
TOTAL			**$ 1.313,50**

Tabla 5.1 Costo total de la implementación del subsistema automóvil

5.1.2 COSTOS INDIRECTOS

El único costo identificado es el combustible usado durante las pruebas de funcionamiento, el valor es de USD 5 considerando la ruta habitual hogar-trabajo.

5.1.3 COSTO TOTAL DEL SUBSISTEMA AUTOMÓVIL

El costo total del subsistema automóvil se lo realiza en base a los costos directos e indirectos detallados anteriormente. La Tabla 5.2 muestra el costo total del subsistema automóvil.

Descripción	Costo Total
Costo Directos	$ 1.313,50
Costos Indirectos	$ 5,00
TOTAL	**$ 1.318,50**

Tabla 5.2 Costo total de subsistema automóvil

5.2 COMPARACIÓN DE COSTOS DE SUBSISTEMA AUTOMÓVIL

En esta sección se hará una comparación del costo del subsistema automóvil considerando la solución anterior, basada en Arduino, y la solución actual basada en un Smartphone Android.

DESCRIPCIÓN		Cantidad	Costo Unitario	Costo Total
Arduino Mega 2560		1	$ 55,00	$ 55,00
Lector ELM327		1	$ 37,00	$ 37,00
MÓDULO ZIGBEE	XBee serie 1	1	$ 70,00	$ 70,00
	XBee Regulated	1	$ 25,00	$ 25,00
	XBee Adapter	1	$ 17,00	$ 17,00
Shield SIM900 Shield		1	$ 90,00	$ 90,00
Shield GPS 2.0		1	$ 45,00	$ 45,00
Tarjeta SIM		1	$ 3,50	$ 3,50
Plan de datos para Módulo Shield SIM900		1	$ 15,00	$ 15,00
TOTAL				**$ 357,50**

Tabla 5.3 Costo del hardware subsistema automóvil del trabajo previo [4]

La Tabla 5.4 muestra el costo del hardware del actual proyecto.

Descripción	Cantidad	Costo Unitario	Costo Total
Smartphone Samsung Ace 3	1	$ 70,00	$ 70,00
Tarjeta SIM	1	$ 3,50	$ 3,50
Lector ELM327 Bluetooth	1	$ 25,00	$ 25,00
Plan de datos Smartphone Básico, incluye 500MB	1	$ 15,00	$ 15,00
TOTAL			**$ 113,50**

Tabla 5.4 Costo del hardware del subsistema automóvil actual proyecto

Del análisis de las anteriores tablas, se puede observar que el actual subsistema automóvil reduce el costo en $ 244 lo que representa un ahorro del 68,3%.

CAPÍTULO 6

6 CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presenta un conjunto de conclusiones y recomendaciones producto de la experiencia y conocimientos adquiridos en el desarrollo del presente proyecto.

6.1 CONCLUSIONES

Se comprobó que el Smartphone Android puede reemplazar la solución del trabajo previo [3] basado en Arduino, cumpliendo con las siguientes funcionalidades: comunicación inalámbrica mediante el uso de Bluetooth para establecer conexión con el lector ELM327 conectado en la interfaz DLC del automóvil, GPS para capturar las marcas de posición, conexión GSM/GPRS para recibir el mensaje de configuración vía SMS y para enviar los datos desde el subsistema automóvil respectivamente.

El presente proyectó reduce 68,3% el costo del subsistema automóvil respecto de los trabajos previos [2]–[4] con lo cual se pretende una mayor adopción de la misma.

Las prácticas de metodologías agiles usadas en el presente proyecto han permitido realizarlo de forma ordenada, rápida y simple.

Un Smartphone reciclado permite conseguir todas las funcionalidades que antes se tenían con lo cual se plantea una alternativa de reciclaje tecnológico.

En el presente proyecto se pudo detectar la llegada de un SMS y procesarlo de manera asincrónica al proceso de lectura de datos desde la ECU. Esto se consiguió a través del uso de la clase BroadcastReceiver mediante el método onReceive().

En el presente proyecto se pudo almacenar los datos del subsistema automóvil en zonas que no existe conexión GPRS y enviarlos al subsistema servidor al

recuperar conexión mediante la clase ConexiónDatosMoviles y el método estadoInternet().

6.2 RECOMENDACIONES

Al momento de poner en funcionamiento el proyecto se debe tomar en cuenta que el mensaje de configuración generado por el subsistema servidor siempre debe incluir el PID 0C ya que con ello el subsistema automóvil verifica que el auto está encendido.

El Smartphone deberá tener deshabilitada la interfaz WLAN con la finalidad de que la aplicación móvil detecte correctamente si datos móviles están habilitados o deshabilitados.

Verificar que Smartphone usado en subsistema automóvil tenga al menos 10MB disponibles para el almacenamiento del mensaje de configuración y archivo de respuesta temporal en caso de no existir conexión GPRS.

Se debe tomar en cuenta que el subsistema automóvil siempre estará funcionando y solicitando información a la ECU a pesar de que el automóvil se encuentre apagado, debido a que el lector ELM327 se mantendrá encendido y consumiendo energía ya que está alimentado por el PIN 16 del conector DLC. Por tal motivo se debe desconectar el subsistema automóvil si el auto permanecerá apagado por un largo tiempo con la finalidad de evitar el agotamiento de la batería del auto.

REFERENCIAS BIBLIOGRÁFICAS

[1] Pedro Mauricio González Castillo, “Diseño de una interfaz gráfica en Labview para el diagnóstico de vehículos por medio de OBD2”, Universidad Pontificia Bolivariana, Bucaramanga, 2010.

[2] Wilson Andrés Simbaña Coyago, “Diseño e implementación de una solución telemática basada en OBD-II (ON-BOARD DIAGNOSTIC) que permita obtener y procesar la información de los sensores del motor de un automóvil”, Escuela Politécnica Nacional, Quito, 2015.

[3] Miriam Lucía Loachamín Loachamín, “Implementación de un sistema de administración remota para el proceso de obtención de datos del sistema OBD-II de un automóvil”, Escuela Politécnica Nacional, Quito, 2015.

[4] Eliana Lorena Montero Revelo, “Diseño e implementación de un sistema web para el monitoreo del estado de un motor”, Escuela Politécnica Nacional, Quito, 2016.

[5] Israel Cervantes Alonso y Saúl Osborn Espinosa Solis, “Escáner automotriz de pantalla táctil”, Instituto Politécnico Nacional, México, 2010.

[6] Christian Arlén Morales Landín y Ulises Yosafat Valverde Jiménez, “SCANNER AUTOMOTRIZ INTERFAZ PC”, ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA UNIDAD PROFESIONAL “ADOLFO LÓPEZ MATEOS”, México, 2010.

[7] Pedro González Melis, “ELECTRÓNICA DEL AUTOMÓVIL OBD II”, ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL DE BARCELONA, Barcelona, 2008.

[8] Vicente Blasco, “Artículo OBDII”, *Sistema de diagnóstico OBD II*. [En línea]. Disponible en: http://www.electronicar.net/IMG/Articulo_OBDII.pdf. [Consultado: 20-jun-2016].

[9] Ángel Martín Velázquez, “Android OBD-II”, Universidad de Valladolid, Segovia.

[10] “Scanner Automotriz Profesional Tech2 Para Suzuki, Isuzu, Gm - U$S 1.499,00”. [En línea]. Disponible en: http://articulo.mercadolibre.com.ec/MEC-409065745-scanner-automotriz-profesional-tech2-para-suzuki-isuzu-gm-_JM. [Consultado: 20-jun-2016].

[11] “Escaner Automotriz Livianos 12v Y 24v Carecar C68 Obd1 Obd2 - U$S 1.500,00”, *Escaner automotriz 1500*. [En línea]. Disponible en: http://articulo.mercadolibre.com.ec/MEC-409053358-escaner-automotriz-livianos-12v-y-24v-carecar-c68-obd1-obd2-_JM. [Consultado: 20-jun-2016].

[12] “Escanner Automotriz Vetronix Bosch Tech 2 +2 Tarjetas Pcmcia - U$S 1.799,00”, *Escaner automotriz 1799*. [En línea]. Disponible en: http://articulo.mercadolibre.com.ec/MEC-409105309-escanner-automotriz-vetronix-bosch-tech-2-2-tarjetas-pcmcia-_JM. [Consultado: 20-jun-2016].

[13] “PC-based Scan Tools”, *OBD Software*. [En línea]. Disponible en: https://www.scantool.net/scan-tools/pc-based/. [Consultado: 20-jun-2016].

[14] “Buy OBD-II Software for Windows, Mac and Linux | OBD Auto Doctor”. [En línea]. Disponible en: http://www.obdautodoctor.com/pricing-and-purchase? [Consultado: 20-jun-2016].

[15] “Interface Elm327 en MercadoLibre Ecuador”, *Escaneres Bluetooth WiFi*. [En línea]. Disponible en: http://listado.mercadolibre.com.ec/interface-elm327. [Consultado: 20-jun-2016].

[16] Paco Blanco, Julio Camarero, Antonio Fumero, Adam Werterski, y Pedro Rodríguez, “Metodología de desarrollo ágil para sistemas móviles Introducción al desarrollo con Android y el iPhone”, Universidad Politécnica de Madrid, Madrid, 2009.

[17] Irene Amador Román, “nfcTicketing: Aplicación para uso de tiques de transporte p_ublico”, Universidad Carlos II de Madrid, Madrid, 2013.

[18] Giseth Johana Grimaldo Botero, “Desarrollo de aplicación móvil de apoyo a la plataforma Web del observatorio ‘Monitoréo de variables físicas y fisiológicas en niños y adolescentes el edad escolar de Risaralda’”, Universidad Tecnológica de Pereira, Pereira, 2013.

[19] Santiago Ceria, “Ingeniería de Software I Casos de uso”. Universidad de Buenos Aires, 1998.

[20] Departamento de Computacion, “Ingeniería de Software I Casos de uso”. Universidad de Buenos Aires, abril-2006.

[21] Santiago Ceria, “Ingeniería de Software I Diagramas de Actividad”. Universidad de Buenos Aires, 1998.

[22] Ignacio Leiva Mundaca y Marco Villalobos Abarca, “Método ágil híbrido para desarrollar software en dispositivos móviles”. Revista chilena de ingeniería, abril-2015.

[23] André Felippa, “Tendencias de Movilidad para 2016 en América Latina”, *Revista Estrategia & Negocios*, 23-dic-2015. [En línea]. Disponible en: http://www.estrategiaynegocios.net/tecnologia/913814-330/tendencias-de-movilidad-para-2016-en-américa-latina. [Consultado: 27-jun-2016].

[24] Romain Huljack, <romain.huljack@telefonica.com>, “Catálogo de equipos celulares 2016”, [Correo electrónico], jul-2016.

[25] Juan Pablo Reinoso, <jreinosz@claro.com.ec>, “Catálogo de equipos celulares 2016”, [Correo electrónico], jul-2016.

[26] Ejecutivo Movistar y Ejecutivo Claro, <romain.huljack@telefonica.com>, <jreinosz@claro.com.ec>, “Catálogo de precios Smartphones 2014”, [Correo electrónico], feb-2014.

ANEXOS

ANEXO 1: CÓDIGO FUENTE DE LA APLICACIÓN MÓVIL DESARROLLADA

El anexo se encuentra en el CD adjunto.