UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Propuesta e implementación de mejoras al proyecto GEMA para su incorporación al mercado de IT.

INFORME DE ACTIVIDADES PROFESIONALES

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

López Rojas Dan Edgardo

ASESORA DE INFORME

MC. María Jaquelina López Barrientos

Ciudad Universitaria, Cd. Mx., 2017

INDICE

INDICE DE FIGURAS

Capítulo I.

Presentación de CPC y

Mi ingreso al campo laboral.

1.1 Sobre la empresa.

El Corporativo de Cobranza *(CPC)* nació en 1999 para apoyar y orientar la decisión de las empresas de lograr sus objetivos y asesorarlas para cubrir expectativas de sus accionistas en un mercado cada vez más complejo. El moderno Concepto de consultoría, administración y gestión de cobranza que *CPC* proporciona, logró en menos de cuatro años que CPC se convirtiera en una de las más grandes empresas administradoras de cobranza a nivel nacional. De 1999 hasta la actualidad, la empresa ha logrado crecer y consolidarse de manera eficaz, proporcionando hoy por hoy, asistencia a través de sus oficinas distribuidas en todos los estados de la república mexicana.

CPC ofrece servicios tales como:

- Negociación de Créditos.
- Recuperación de Créditos.
- Seguimiento y Control de Créditos y Procesos Legales.
- Asesoría Financiera.
- Venta de Inmuebles.

CPC está conformada por 5 departamentos los cuales son: Jurídico, Contraloría, Recursos humanos, Contabilidad y Sistemas. Cada uno con sus respectivas áreas, en el departamento de Sistemas encontramos 3 áreas que son: Soporte Técnico, Soporte a Proyectos y Desarrollo. El Departamento de Sistemas (véase figura 1.1) es capaz de atender las necesidades de todo el personal que labora en los diferentes estados de la República Mexicana, desde soporte técnico a sus equipos, multifuncionales e impresoras, así como soporte y asesoría a los usuarios de los diversos sistemas que forman parte fundamental de la operación de la empresa. También atiende las necesidades que van surgiendo con el paso del tiempo ya sea con el desarrollo de un nuevo sistema, o nuevas funcionalidades dentro de los ya existentes, de tal manera que satisface todos los requerimientos que la operación demanda, y así contribuir con el crecimiento continuo de la empresa.

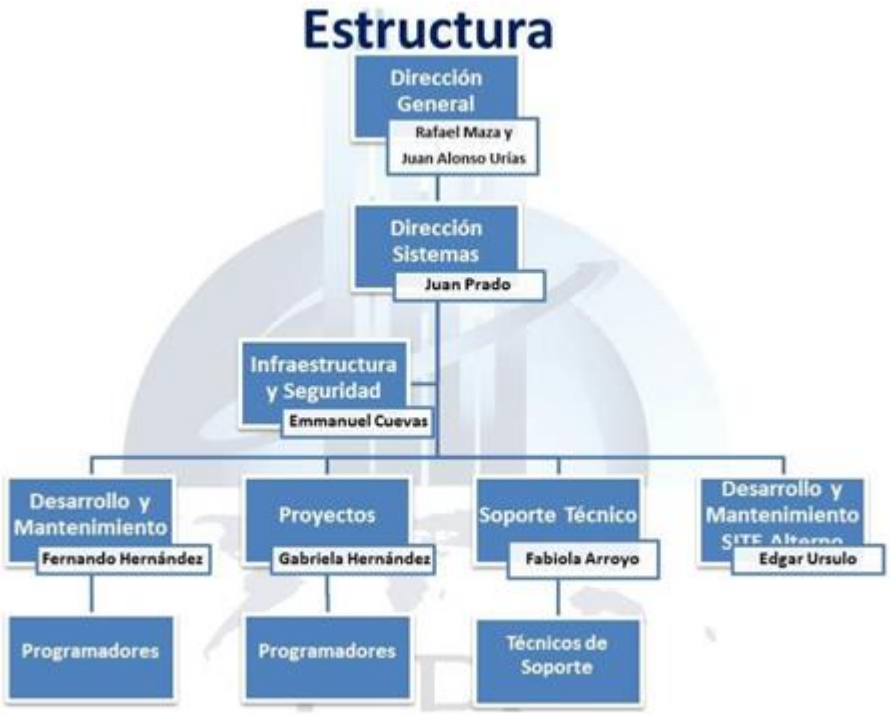


Figura 1.1. Organigrama del departamento de sistemas

1.2 Mi ingreso al campo laboral.

A finales del año 2014, me encontré en la necesidad de buscar un empleo para poder sostener mis estudios, cursaba 6° semestre de la carrera, así que decidí buscar un trabajo relacionado con mis estudios, ya que la FI me había brindado las habilidades necesarias para ello. Encontré en internet vacantes en CPC para el área de Mantenimiento de Sistemas, el puesto era de medio tiempo con conocimientos en programación *C#, asp .net* y *sql server*. Hasta el momento no contaba con conocimientos muy elevados sobre *sql* y base de datos, pero conocía lo esencial dado que había tomado unos cursos intersemestrales en la facultad; así que decidí postularme para la vacante.

Cabe señalar, que siempre me interesó el desarrollo de software y era muy bueno en ello cada vez que tenía la oportunidad de hacer algún proyecto en la FI, me convencía acerca de mis capacidades en la programación.

Me llamaron para hacer las pruebas necesarias y competir por el puesto, después de los exámenes psicométricos, realicé el examen del área, el cual consistía en realizar un sistema web en asp.net con C# y fuera capaz de realizar consultas a la base de datos a través de *stores procedures* en *SQL* SERVER, básicamente era realizar un *CRUD* con las tecnologías ya mencionadas. Terminé mi examen con éxito y entré a trabajar el 5 de enero del 2015 en el área de soporte a los sistemas. Fue así como conseguí mi primer empleo correspondiente a mi carrera, la perspectiva que yo tenía en ese momento era muy pequeña, pues si bien contaba con las bases teóricas necesarias para ser considerado en el puesto, carecía completamente de experiencia en el rubro y no hace falta señalar que, es muy diferente el ambiente académico de la Universidad que un ámbito laboral específico, aplicar todos los conocimientos adquiridos al enfoque de la empresa.

Me desempeñé durante casi dos años en la empresa, hasta diciembre del 2016, en este tiempo tuve la oportunidad de participar en diversos proyectos y a través de los cuales fui creciendo profesionalmente, también logré ganarme la confianza de mis superiores para ser considerado en proyectos más importantes, hasta que llegué a ser responsable de una aplicación móvil en Android, de la cual hablaré más adelante.

Capítulo II.

Mi participación en diversos Proyectos de la empresa.

2.1 Proyecto SIAC

Del 8 de enero del 2015 al 22 de enero del 2015.

2.1.1 Descripción.

El sistema SIAC, es un sistema para el control interno de recursos que los empleados necesiten, el cual, ayuda a registrar sus gastos para que el área de tesorería pueda reembolsar las cantidades necesarias, existen 3 tipos:

- Anticipos: todos los asuntos que requieran recursos, sin contar con la documentación que respalde dicho requerimiento al momento de la solicitud.
- Reembolsos: Se refiere a todo aquel pago de un gasto u honorario ya erogado, el cual cuente con un documento que lo ampare.
- Viáticos: Al igual que el reembolso cuentan con documentos que avalen el gasto exclusivamente en transporte.

SIAC, sigue un flujo de autorizaciones, el cual empieza desde la creación de la solicitud, pasa a revisión por el área de contabilidad, continua hacia tesorería, vuelve al área de contabilidad y se realiza el reembolso.

SIAC es un sistema desarrollado en Visual Studio 2012, su base de datos está en SQL SERVER 2012 y está programado en *C#*, *asp.net*, utiliza *JQUERY* para los *JQGRID*.

2.1.2 Problemática

Dentro de la mayor parte de SIAC, se encuentran módulos de consulta de solicitudes (véase figura 2.1), los cuales se representan a través de un JQGRID en el *frontend*, el sistema hace la consulta a la base de datos a través de LINQ desde el *backend* programado en *C#*, dicha consulta tarda más de 5 minutos para traer solamente 80 registros, esto se ve afectado cuando los registros son miles, una consulta puede llegar a tardarse más de 15 minutos, lo cual impacta en el tiempo que les lleva a los empleados para realizar búsquedas o a los usuarios con permisos para autorizar solicitudes. Dicho proceso puede observarse en la figura 2.2.

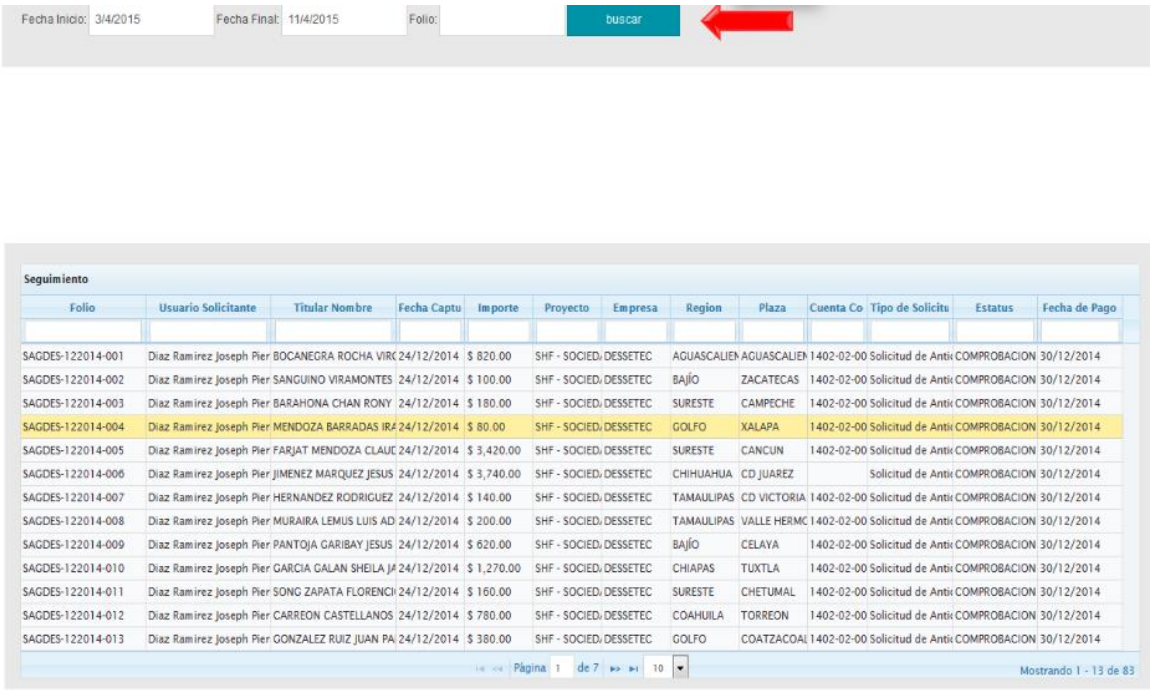


Figura 2.1 Vista del sistema SIAC, para la consulta de solicitudes.

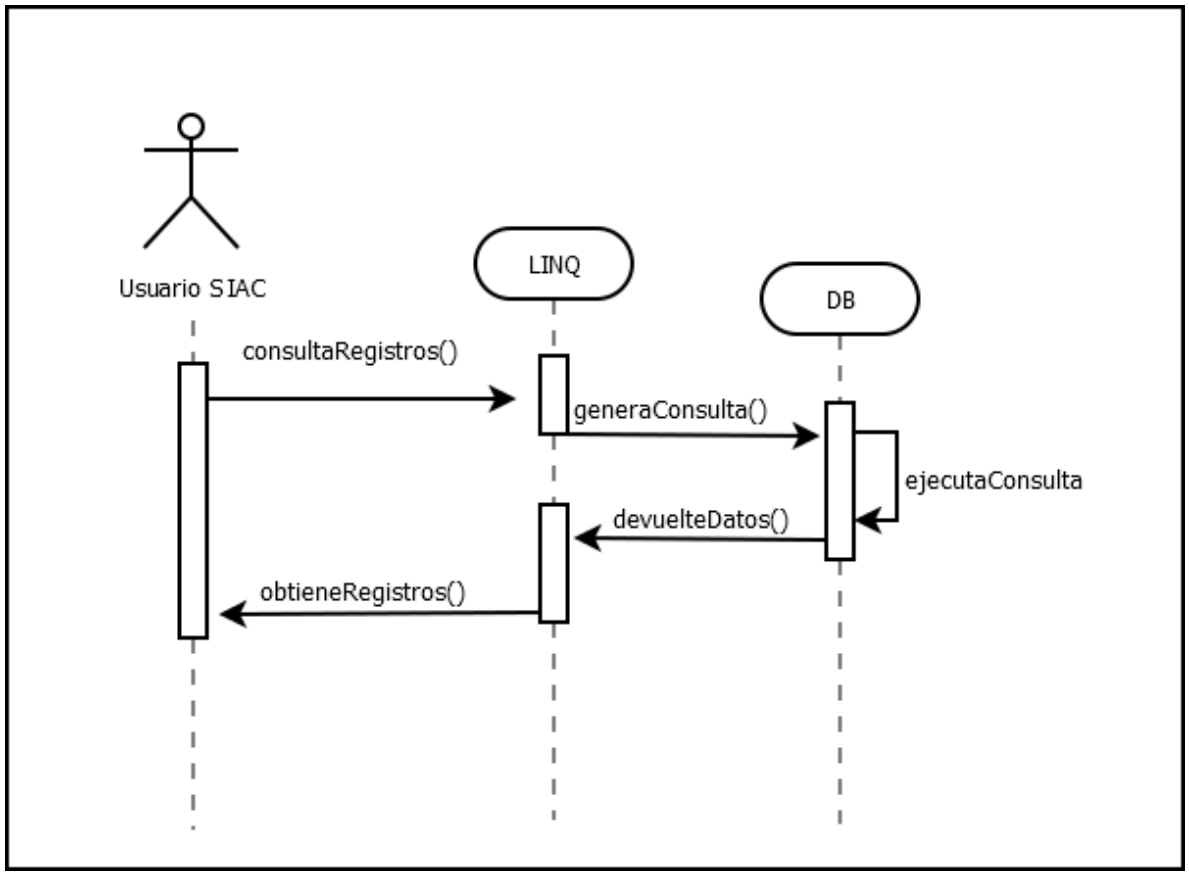


Figura 2.2 Diagrama del proceso de consulta de registros en SIAC

2.1.3 Objetivo.

El objetivo es claro, reducir el tiempo de búsqueda de los registros, para optimizar el funcionamiento del sistema, alcanzar el tiempo mínimo posible en las consultas a la base de datos.

2.1.4 Actividades.

Hasta ese momento, ya se habían presentado diferentes problemas con *LINQ* en otros sistemas, por eso ya no se le consideraba para la realización de proyectos en la empresa y se estaba modificando la manera de hacer las consultas necesarias a la base de datos, por ello decidí implementar una estructura de consultas a través de *stored procedures,* ya que al modificar las consultas en un SP, se reflejarán de manera instantánea en los GridViews con la propiedad AutoGenerateColumns, sin necesidad de compilar de nuevo el sistema y así evitar interrumpir la disponibilidad del mismo. Mi primer paso fue la creación de un *SP* en la base de datos en el cual implementé una búsqueda completa entre las diversas tablas para la obtención de los datos.

Dicho *SP* recibe de parámetros las fechas de búsqueda y a partir de ellas se realiza la consulta, una vez obtenida, se manda al *backend* en el cual metí el resultado de la búsqueda en un objeto *DataSet.*

Ya que las tablas en el *frontend* eran de JQGRID, estos reciben como parámetros objetos JSON para el llenado de los registros, entonces mi búsqueda la cual se encontraba en un *DataSet*, fue serializada para convertirla en un *JSON* y así poder mandarla al *JQGRID*.

2.1.5 Resultados.

Los resultados fueron favorables, con el nuevo proceso que implementé (véase figura 2.3), logré que las búsquedas del sistema quedaran en un promedio de 5 segundos, las cuales fueron medidas a través de la consola de debug del Visual Studio , optimizándolas y ofreciendo un mejor servicio a los usuarios, mi jefe reconoció mi trabajo y esto me valió para que me cambiaran de área, pasé de estar en soporte de aplicaciones a desarrollo en menos de 1 mes, de igual manera me cambiaron de proyecto, para ahora comenzar a desarrollar módulos nuevos. Hasta el momento ya comprendía un poco más sobre el lenguaje y acumulé en poco tiempo mucho conocimiento sobre *TSQL*, también comprendí el concepto del *Handler,* el cual utilizan los sistemas de CPC para el envió de peticiones y respuestas entre el cliente y el servidor (véase figura 2.4). En ese momento me sentía preparado para comenzar a desarrollar.

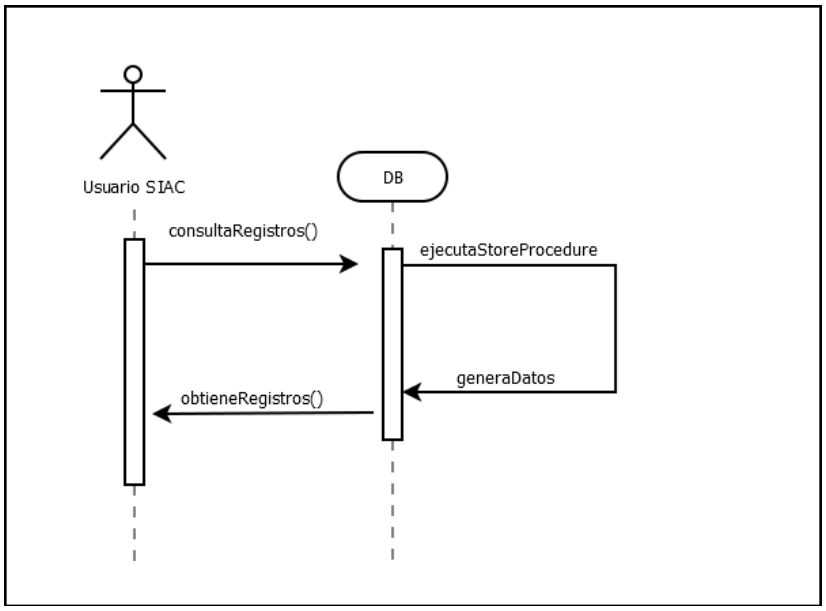


Figura 2.3. Nuevo proceso de consulta de solicitudes en SIAC.

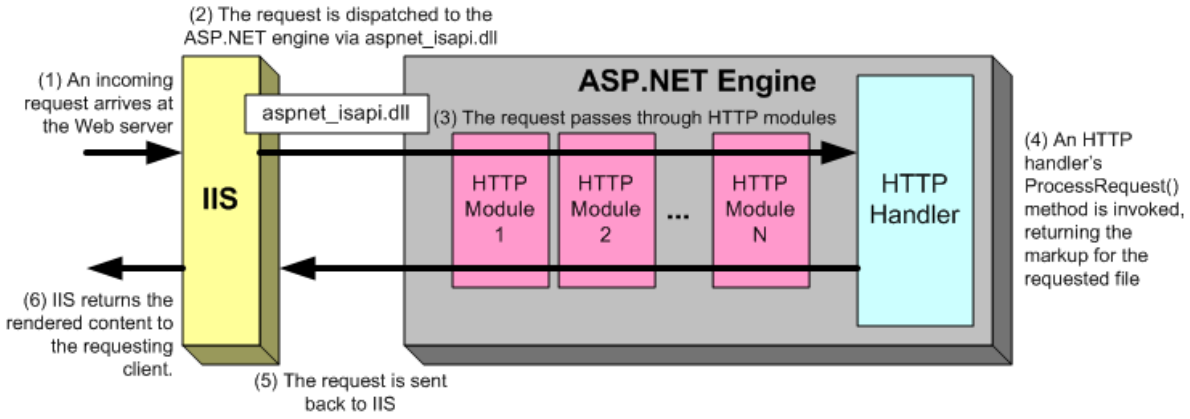


Figura 2.4. Diagrama del funcionamiento de un handler.

2.2 Proyecto SAB

Del 22 de enero del 2015 a octubre del 2015

2.2.1 Descripción.

SAB, fue el siguiente proyecto al que fui asignado, el objetivo del sistema es controlar, visualizar y almacenar la información de la cartera de créditos CARTERA-X y CARTERA-Y y así poder administrar el avance de los créditos en sus procesos y gestiones. Consta de 5 módulos principales:

- Control Documental: Permite al usuario la digitalización de documentos con una generación de carátula y un código de barras que asocia el documento con el expediente en la base de datos.
- Toma de posesión: El usuario lleva un control de las asignaciones y gestiones, así como reportes del estatus de la toma de posesión de las viviendas.
- Avalúos: Este módulo es para llevar un control sobre los avalúos de las viviendas.
- Expediente digital: Muestra toda la información de los documentos digitalizados en sus diversas etapas procesales, bitácoras y detalle de gestión de gastos.
- Reportes: En esta sección el usuario es capaz de visualizar información específica de los bienes dividida por área.

Durante mi estancia en este proyecto, tuve la oportunidad de participar en diferentes desarrollos y diversos módulos, pero en particular me basaré en el módulo Avalúos que fue el que yo elaboré completamente. SAB es un sistema que está en *C#* con *asp.net* y se ayuda de la *suit* de componentes *UI Dev Express* la cual nos permite agregar funcionalidad a los *GridViews*. Su base de datos está en *SQL SERVER* 2012.

2.2.2 Problemática.

El módulo de avalúos, consistía en bajar un formato en Excel, llenarlo y subirlo para cargar la información de los avalúos, una vez el sistema validaba la información del Excel, se guardaba en la base de datos y se continuaba con la edición si era necesario. Esta operación (véase figura 2.5) era incómoda y tardada para los usuarios, puesto que no se tenía control de los datos erróneos y solamente se rechazaba el archivo, también se llevaba tiempo en pasar su información al formato.

Durante el desarrollo surgió otro problema, los registros a guardar eran aproximadamente 8 mil, lo que representaba un tamaño en bytes que excedía el tamaño máximo de transferencia permitido por un *handler*.

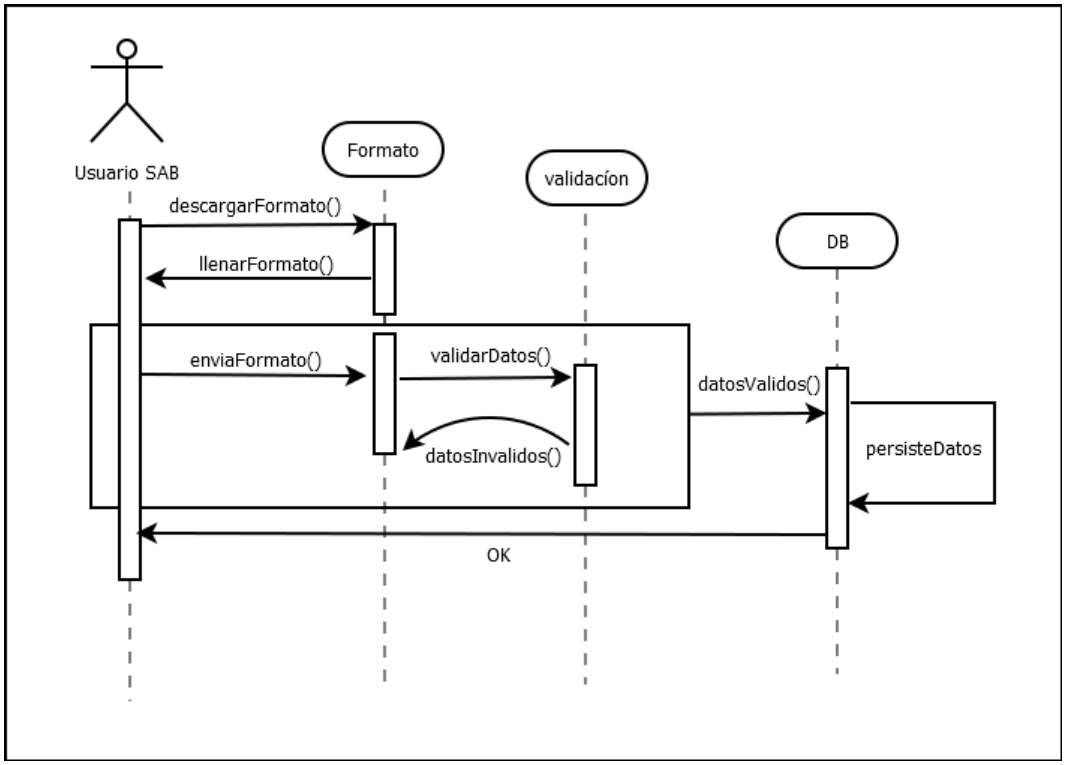


Figura 2.5. Proceso de carga de información en SAB.

2.2.3 Objetivo.

Idear una manera con la cual se pudiera hacer un copy-paste de la información del usuario directamente a un *grid* en el sistema al mismo tiempo validar la fila y celda de los datos que fueran erróneos. Poder cargar y validar hasta 8 mil registros al mismo tiempo.

2.2.4 Actividades.

Para empezar, me dediqué a investigar sobre tecnologías que satisficieran la necesidad de simular una hoja de Excel en un sistema web, entre las que encontré decidí utilizar *Handsontable*, la cual es una tecnología que utiliza *JavaScript* y de licencia abierta, para poder implementar un *grid* en el cual podemos editar, borrar y cargar datos en las celdas (véase figura 2.6). Posteriormente aprendí a utilizar esta herramienta, puesto que nunca se había utilizado en la empresa, aprender sus métodos

y la manera de obtener los datos y enviarlos al servidor. Proseguí con la implementación e integración de *Hansontable* al sistema, para ello descargué la herramienta desde su página oficial, como es *JavaScript*, se integra al sistema haciendo referencia en la vista con la sentencia <script src=””></script> donde src es la ruta relativa del archivo descargado.

Una vez integrada la herramienta al proyecto, se crearon las tablas y las reglas necesarias para la carga de la información, con columnas definidas, estaban listas para ser llenadas, hasta el momento ya se podían copiar y pegar datos desde Excel al *Handsontable*, de igual manera por medio de *JavaScript* se podía localizar el número de celda que no correspondiera a un dato válido, de tal manera que se tenía un control de los errores resultantes al momento de la carga. Surgió otro problema el cual era enviar la información al servidor, el tamaño máximo del *request* era aproximadamente 4 mb, y mandar más de 8 mil registros por medio del *hanlder* era imposible ya que excedía este límite. Así que decidí implementar una manera más actual para enviar la información al servidor, utilizar *WebMethod*, método que hereda de la clase *WebService*, este a diferencia del *handler* que recibe toda la información en una solo cadena serializada, manda pequeños paquetes hasta completar la transferencia total de información, de esta manera no importa el tamaño que se desee enviar ya que viajará de manera separada hasta el servidor.

	ID	Country	Code	Currency	Level	Units	Date	Change
1	1		EUR	Euro	0.9033	EUR / USD	08/19/2015	0.26%
2	2		JPY	Japanese Yen	124.3870	JPY / USD	08/19/2015	0.01%
3	3		GBP	Pound Sterling	0.6396	GBP / USD	08/19/2015	0.00%
4	4				0.9775	CHF / USD	08/19/2015	0.08%
5	5				1.3097	CAD / USD	08/19/2015	-0.05%
6	6				1.3589	AUD / USD	08/19/2015	0.20%
7	7				1.5218	NZD / USD	08/19/2015	-0.36%
8	8		SEK	Swedish Krona	8.5280	SEK / USD	08/19/2015	0.16%
9	9		NOK	Norwegian Krone	8.2433	NOK / USD	08/19/2015	0.08%
10	10		BRL	Brazilian Real	3.4806	BRL / USD	08/19/2015	-0.09%
11	11		CNY	Chinese Yuan	6.3961	CNY / USD	08/19/2015	0.04%
12	12		RUB	Russian Rouble	65.5980	RUB / USD	08/19/2015	0.59%
13	13		INR	Indian Rupee	65.3724	INR / USD	08/19/2015	0.26%
14	14		TRY	New Turkish Lira	2.8689	TRY / USD	08/19/2015	0.92%
15	15		THB	Thai Baht	35.5029	THB / USD	08/19/2015	0.44%
16	16		IDR	Indonesian Rupiah	13.8300	IDR / USD	08/19/2015	-0.09%
17	17		MYR	Malaysian Ringgit	4.0949	MYR / USD	08/19/2015	0.10%
18	18		MXN	Mexican New Peso	16.4309	MXN / USD	08/19/2015	0.17%

Figura 2.6, obtenida del sitio https://handsontable.com/examples.html?headers

2.2.5 Resultados.

El nuevo proceso (véase figura 2.7) implementado fue exitoso, el usuario era capaz de cargar su información simplemente pegándola y validándola al mismo tiempo, a su vez, esta era editable al momento sin tener que hacer más cargas. También podía cargar miles de registros sin problema alguno, este módulo gustó a los usuarios de modo que se replicó la herramienta en otros sistemas, de uno de ellos hablaré más adelante, finalizando este desarrollo de nuevo fui cambiado de proyecto, acercándome cada vez más al proyecto definitivo en el que quedé como responsable.

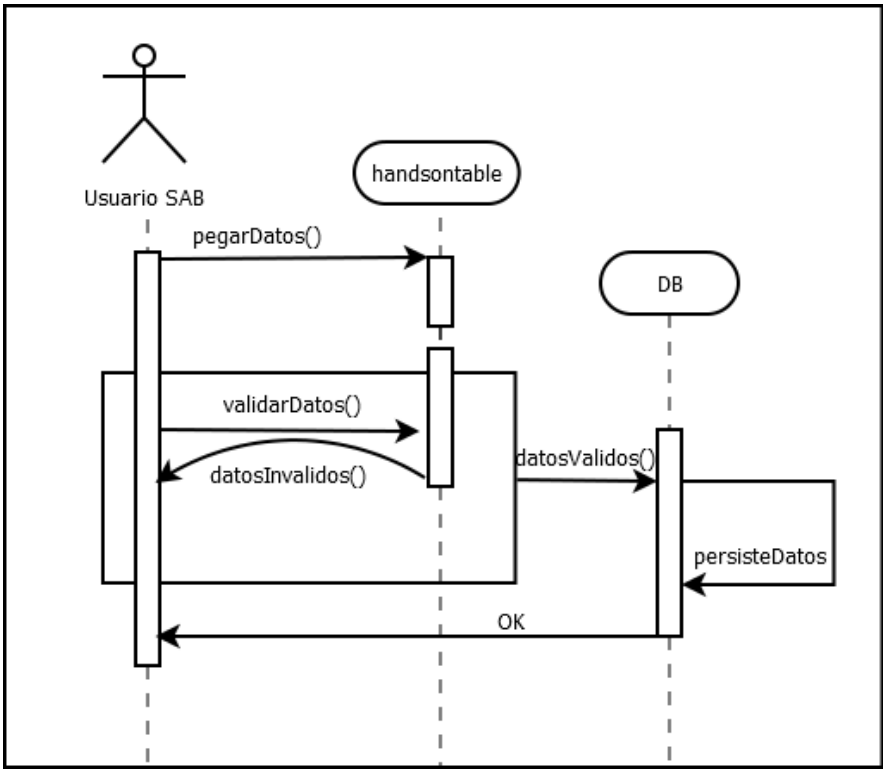


Figura 2.7. Proceso nuevo de carga de información en sistema SAB

2.3 Gestión Móvil.

De octubre del 2015 a diciembre del 2016

2.3.1 Descripción.

Gestión Móvil, es un sistema programado en C# con ASP.NET en el cual se incorporó la herramienta *handsontable* que utilicé en SAB, este sistema está directamente ligado con la aplicación móvil, ya que en él se ve reflejada toda la información obtenida a través de GEMA, Gestión Móvil consta de dos secciones:

- Asignación: Los créditos (crédito perteneciente a un deudor) disponibles son asignados a los gestores, y se mantienen en la base de datos, listos para ser descargados con la aplicación GEMA.
- Reporte de Resultados: Es esta sección, se pueden visualizar todas las gestiones realizadas, las cuales fueron cargadas al sistema a través de la aplicación GEMA, se muestran las gestiones en Grids y son exportables a Excel.

La asignación utiliza *handsontable,* se copian los créditos y se pegan en el sistema, de esta manera se pueden asignar múltiples créditos en poco tiempo.

2.3.2 Problemática.

Gestión Móvil, se utilizaba solamente para un cliente, se asignaban créditos de dicha cartera y se podían solamente gestionar estos créditos, ya que era una potente herramienta en conjunto con GEMA, se decidió transformar al sistema para ser “Multiproyecto”, es decir, poder gestionar todos los créditos de cualquier cliente, siguiendo sus reglas específicas de cada uno de ellos sin afectar el funcionamiento del sistema para las demás carteras de créditos.

2.3.3 Objetivo.

Hasta el momento CPC contaba con 4 carteras las cuales se identificaban como: CARTERA1, CARTERA2, CARTERA3 y CARTERA4, el objetivo era incorporar las cuatro carteras al sistema para ser gestionadas y, además, poder seguir incrementando carteras que se adquirieran en un futuro.

2.3.4 Actividades.

En esta ocasión, no me tocó programar nada, se me asignó la tarea de diseñar la estructura y hacer todo el análisis para el sistema de manera que cumpliera el objetivo, así que decidí mover la lógica del negocio a la base de datos y crear una serie de relación entre proyectos con reglas definidas en nuevas tablas (véase figura 2.9), de esta manera se pueden relacionar el proyecto con sus reglas desde la base de datos y consultarlas desde el sistema; por ejemplo, el sistema reconocerá el proyecto al cual se le asignarán créditos, irá a la base de datos a consultar las reglas definidas para dicho proyecto y las aplicará para hacer las validaciones pertinentes. Si se desea agregar un nuevo proyecto, basta con agregarlo a la base de datos, así conseguimos un sistema web dinámico y escalable, listo para satisfacer las necesidades de GEMA, este fue el inicio del último proyecto al que estuve asignado, del cual hablaré en el siguiente capítulo. Como se muestra en la figura 2.8, se puede apreciar el uso de *handsontable* y la elección de proyecto del sistema.

Figura 2.8 Pantalla del módulo Asignación del sistema Gestión Móvil.

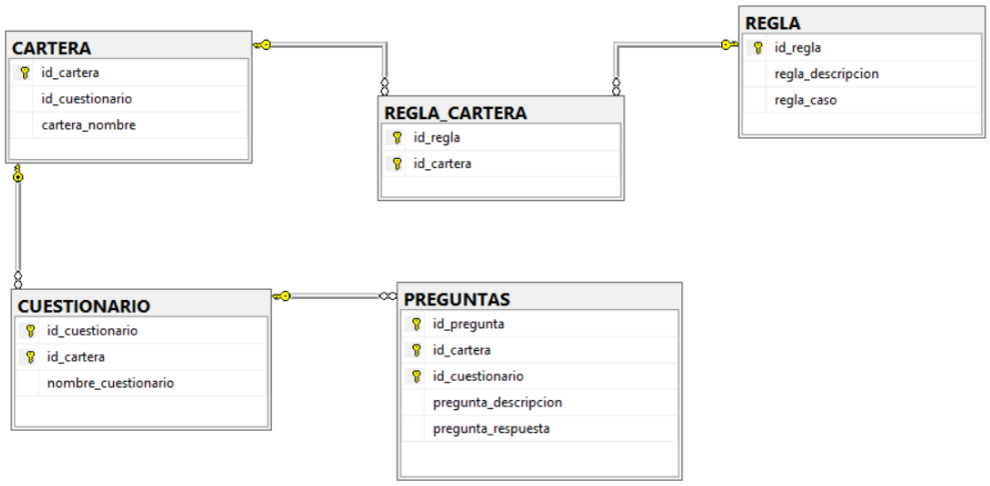


Figura 2.9. Diagrama de la relación de proyectos y reglas en Gestión Móvil

2.2.5 Resultados.

Gestión Móvil se convirtió en el sistema de gestión universal, para todas las carteras que CPC iba adquiriendo, el sistema era capaz de satisfacer las necesidades de cada proyecto y la incorporación de estos al sistema era relativamente sencilla, de manera que los directores de cada plaza podían usar el mismo sistema para asignar créditos de diferentes proyectos para su gestión, el desarrollo del sistema fue a la par con GEMA, ya que los dos son fundamentales para la operación.

Capítulo III.

GEMA

Gestión Móvil Avanzada.

3.1 Antecedentes.

3.1.1 Inicios de Gema.

La aplicación GEMA, antes de todos los cambios que le apliqué, era un proyecto desarrollado en Android bajo el esquema *ANT* del IDE *eclipse*, funcionaba solamente para una cartera de clientes al igual que Gestión Móvil y tenía muchos errores dentro de la aplicación los cuales provocaban el cierre inesperado de esta. Su función era solamente guardar la información que los gestores recopilaban a través de cuestionarios que les aplicaban a los deudores, también, guardaba fotografías de la vivienda al final del cuestionario y se tomaba la ubicación GPS del equipo móvil. GEMA era una aplicación con bastante potencial, pero aún no se había explotado, pues CPC utilizaba otra aplicación para sus demás carteras, la aplicación es bastante conocida en el mercado, su nombre es *BlueMessaging*.

3.1.2 *BlueMessaging*.

Es una aplicación muy conocida en el ámbito de soluciones IT, una de las carteras principales de CPC, lo incorporó como sistema de cobranza y obligaba a todas las consultoras de cobranza a usarlo para gestionar su cartera de deudores. Esta aplicación tiene muchas funciones, pero limitándola al ámbito de cobranza, su función es contestar un cuestionario que sigue cierto camino definido y además puede responder automáticamente y de manera transparente preguntas del cuestionario con datos precargados, esta función es publicitada en su página de internet como una inteligencia artificial capaz de seguir diferentes caminos o secuencias de preguntas a realizar y facilitar la gestión. Esta aplicación inspiró el nuevo funcionamiento de GEMA, del cual hablaremos más a delante.

3.2 Análisis.

3.2.1 Problemática.

Como mencioné anteriormente, GEMA utilizaba el esquema ANT de eclipse, en verano del 2015 Google anunció que dejaría de dar soporte a eclipse, ya que 6 meses atrás, Google lanzó su propio *IDE,* llamado Android Studio, esto significaba un problema para la aplicación, pues para futuros desarrollos, era importante mantenerse actualizado. Hasta el momento, no existe un motivo funcional, es decir, alguna función de la API de Android que nos limite usarla en eclipse, sin embargo google nos recomienda usar Android Studio y enlista algunas de las ventajas, las cuales son:

- Es puramente Android, creado para programar en Android.
- Se actualiza constantemente.
- El rendimiento es mejor en Android Studio.
- Más rápido que Eclipse.
- Más intuitivo, más fácil de usar.
- Código más ordenado, estructurado y mejores sugerencias.
- Es mejor para diseñar Interfaces.
- Compilador Gradle.
- Las mejores plantillas para empezar proyectos.
- Exporta .APK más fácil.
- Firma aplicaciones.

Para encaminar a GEMA al mercado de IT, era necesario hacer mejoras para que se convirtiera en una aplicación competente, es por eso que se necesitaban implementar algunos cambios, uno de ellos era la instalación de la aplicación y sus actualizaciones de manera automática, además, proporcionaba seguridad en cuanto a gestiones realizadas con la versión correcta de GEMA, lo cual hacía que se evitaran errores por versiones desfasadas.

Como mencioné anteriormente, *BlueMessaging* contaba con un cuestionario que era capaz de tener preguntas precargadas y seguir un camino dependiendo de los datos precargados en la aplicación, esta funcionalidad tenía que ser replicada en GEMA para fortalecer la aplicación.

Otro de los principales problemas de GEMA, era su inestabilidad, pues la aplicación se detenía constantemente impidiendo a los gestores realizar su trabajo de manera eficaz.

El soporte a los usuarios de GEMA, se realizaba vía telefónica, los gestores debían tener un equipo de cómputo a la mano y su equipo móvil para poder copiar los archivos de la aplicación y enviarlas por correo al equipo de sistemas que atendía casos de GEMA, esta operación era demasiado tardada, aproximadamente un soporte vía telefónica a GEMA duraba entre 40 y 60 minutos, lo que representaba una pérdida de tiempo y dinero para la empresa.

El último problema estaba asociado al área de soporte técnico de CPC, pues ellos tenían un inventarío de equipos telefónicos que se enviaban a las plazas, ellos llevaban un reporte de fallas y equipos perdidos, dañados o robados, pero era muy difícil localizar todos los equipos enviados a diferentes plazas de la República Mexicana y como entre usuarios se prestaban los equipos, era difícil detectar la última persona que tuvo acceso al equipo, es por ello que se decidió registrar datos del equipo y la versión de GEMA en la base de datos y así facilitar la tarea del rastreo de equipos.

3.2.2 Objetivos.

1. Migrar el proyecto GEMA de eclipse a Android Studio.
2. Convertir GEMA en una aplicación multiproyecto como Gestión Móvil, de manera que sea capaz de satisfacer las necesidades de cada cartera, y a su vez, sea escalable.
3. Solucionar la inestabilidad de la aplicación, No debe detenerse ni cerrarse inesperadamente.
4. Implementar cuestionarios con preguntas automáticas, y GEMA sea capaz de determinar el camino a seguir, a partir de estas respuestas.
5. Incluir un soporte remoto, para facilitar la atención a usuarios y reducir el tiempo en la solución de problemas. (Iniciativa propia).
6. Registro de datos del equipo utilizado para gestionar, en la base de datos, tales como *IMEI,* modelo, compañía telefónica, versión de sistema operativo.
7. Actualizaciones automáticas y obligatorias de GEMA.

3.2 Implementación.

3.3.1 Migrar GEMA a Android Studio.

Ya que las estructuras de los archivos de proyectos entre Eclipse que usa *ANT* y Android Studio que usa *GRADLE* son incompatibles entre sí, es necesario migrar toda la aplicación, para esto, *Google* proporciona una herramienta integrada en el IDE Android Studio con la cual se puede llevar a cabo la migración.

Primero abrí el proyecto GEMA en Eclipse e instale el Plugin ADT de eclipse (los pasos para la instalación del plugin se pueden consultar en el apartado 1.1), este plugin ayuda a exportar el proyecto generando el archivo Gradle. Una vez dentro del IDE Eclipse, me dirigí a la parte superior izquierda en el menú *File -> Export -> Android -> Generate Gradle Build Files.* Y seleccione el proyecto GEMA. Esto me generó el archivo build.gradle, necesario para la migración, la estructura de los archivos se muestran en la figura 3.1 a manera de ejemplo.

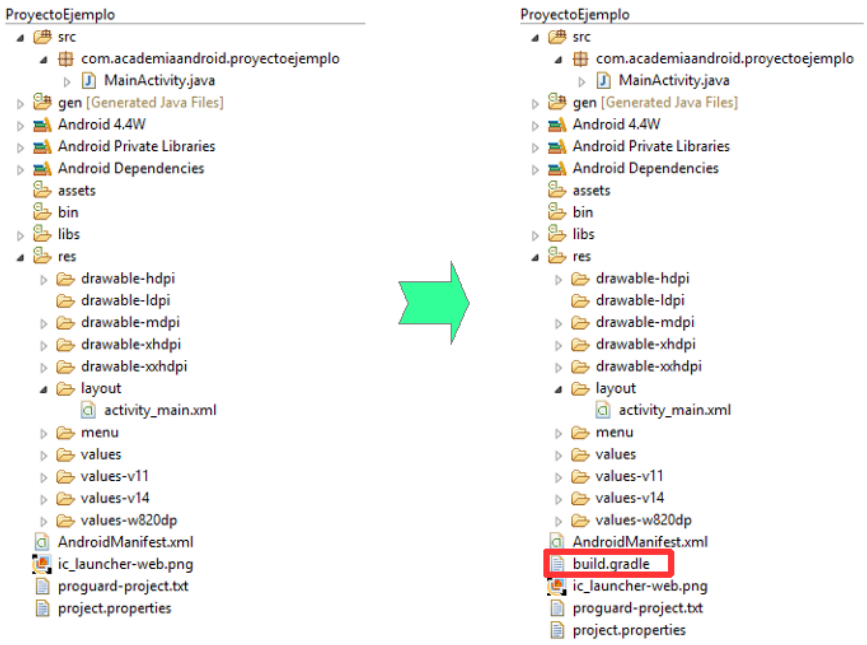


Figura 3.1 Obtenida del sitio https://academiaandroid.com/android-studio-importacion-proyectos-eclipse-otras-caracteristicas-destacadas/

Hasta este momento ya tenía listo mi proyecto GEMA con el archivo gradle necesario, así que proseguí a iniciar Android Studio, se elige la opción Import-Non Android Studio Project como se muestra en la figura 3.2

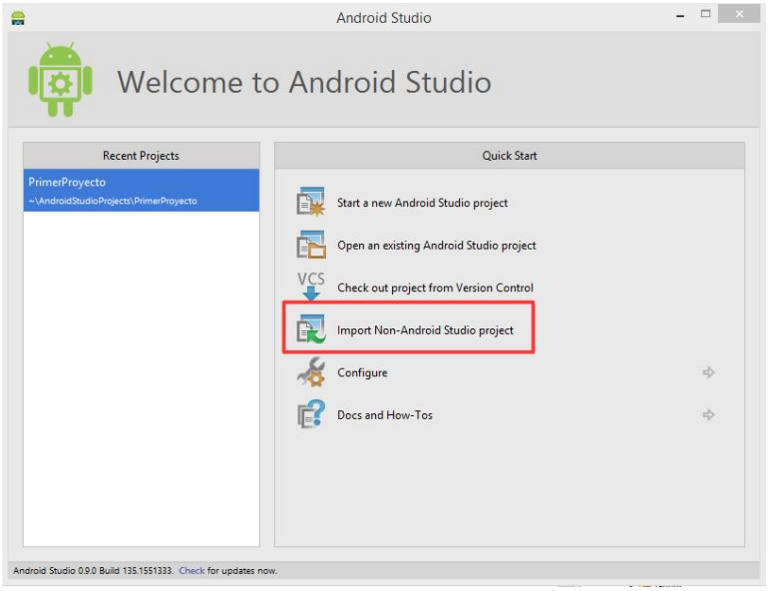


Figura 3.2. Importar proyecto a Android Studio.

Continúe seleccionando el archivo "Build.Gradle" que generé anteriormente y el sistema comienzo el proceso de reconstrucción del proyecto GEMA. Una vez que terminó el proceso, el proyecto estaba listo para ser ejecutado sobre Android Studio.

Comencé la fase de pruebas, para esto fue necesario instalar un driver de Motorola y así poder ejecutar GEMA en modo *Debug* sobre los equipos disponibles en la empresa para efecto de realizar las pruebas pertinentes (la instalación de dicho driver puede consultarla en el apartado 1.2). Conecté el equipo móvil a la computadora, habilité el modo "Depuración por USB" en el dispositivo móvil y ejecute el proyecto GEMA desde Android Studio, los resultados fueron favorables, pues la aplicación se instaló correctamente en el dispositivo y estaba lista para ser utilizada. El proyecto ya estaba en una versión de Android Studio listo para empezar los nuevos desarrollos.

3.3.2 Convertir GEMA en Multiproyecto y a su vez, sea escalable.

GEMA utiliza *SQL LITE* como base de datos, para poder diferenciar los créditos descargados a la aplicación por proyecto realicé un cambio en las tablas de la aplicación, creé una tabla que relacionara los proyectos con el usuario la cual se muestran en la figura 3.3. De esta manera, al recibir la descarga de gestiones, tenía ligada cada gestión a su proyecto por usuario, entonces comencé la programación pertinente para aprovechar esta información que nos brinda la base de datos.

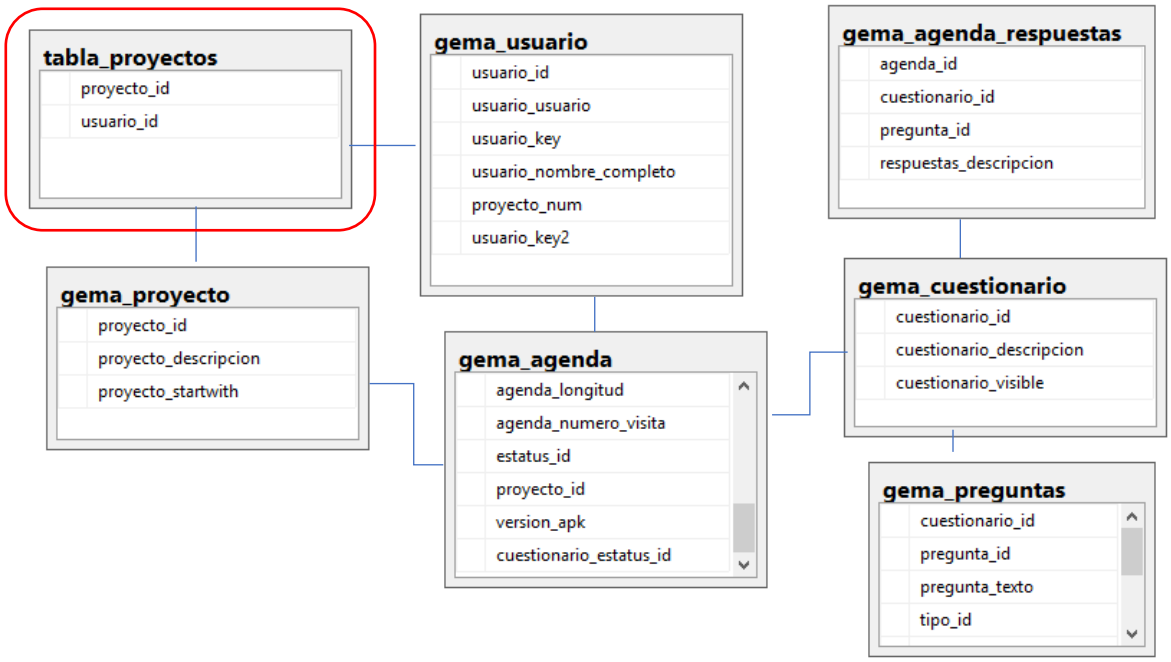


Figura 3.3. Ejemplo de la Estructura de la Base de Datos de GEMA

Comencé por diseñar la vista del *Activity* que contenía la elección del proyecto por parte del usuario, para ello, creé un archivo .xml con el nombre “activity_proyecto.xml” en el cual inserté el siguiente código:

En este fragmento de código, se crea un botón con el cual es posible salir de la pantalla, se describen las dimensiones del icono, la ruta de la imagen que contendrá y un ID que identifica a dicho botón.

```
<com.ridle.views.ButtonFloat
android:id="@+id/buttonFloatSalir"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_alignParentRight="true"
android:background="#1E88E5"
ridle:animate="false"
ridle:iconDrawable="@drawable/ic_logout">
</com.ridle.views.ButtonFloat>
```

También agregué un *ListView* para cargar la información de los proyectos y que el usuario pudiera elegir de la lista, el proyecto para gestionar. En el código se puede observar que el *ListView* contiene un ID para ser identificado y a su vez contiene el ID del botón, con el fin de mantenerlo arriba del *ListView* y fuera siempre visible.

```
<ListView android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:id="@+id/mainListView"
android:isIndicator="true"
android:textColor="@color/Black"
android:layout_above="@+id/buttonFloatSalir"
android:layout_alignParentLeft="true">
</ListView>
```

Posteriormente, programé el *Backend* para llenar esta vista, el código es el siguiente:

```
public class obtenerProyectos extends AsyncTask<Void, Void, Boolean> {
    private ArrayList proyectos;
    private int user;
    private ProgressDialog mDialog;
  protected void onPreExecute() {
      mDialog = ProgressDialog.show(proyecto.this,"Please wait...",
"descargando Proyectos ...", true);
  }
    LoginTask_(int usuario_id) {
        proyectos = null;
        user = usuario_id;
    }
 protected void onProgressUpdate(Void... values) {
    }
    protected Boolean doInBackground(Void... params) {

        proyectos = proyectos(user);
        return true;
    }
    protected void onPostExecute(final Boolean success) {

        ArrayList proyectos = new ArrayList();

        proyectos = gm.getList_Proyecto();
        mainListView = (ListView) findViewById(R.id.mainListView);
        listAdapter = new ArrayAdapter<String>(getApplicationContext(),
R.layout.simplerow, pList);
        mainListView.setAdapter(listAdapter);
            }
        };
        mDialog.dismiss();
    }
    protected void onCancelled() {
        proyectos = null;

    }
}
```

Se crea una clase que hereda los atributos de la clase definida como tarea asíncrona, esto es necesario para todo aquel procedimiento que requiera hacer Android, ya que si no se controla de esta manera, el sistema puede llegar a fallar, siendo que necesite ciertos datos que aún no los conoce, generalmente el error es un *NullPointerException.* Una tarea asíncrona en Android se divide en 4 partes principalmente, las cuales son:

- Metodo onPreExecute(): Se preparan los datos necesarios para realizar la tarea asíncrona, en mi caso, inicie un *Dialog,* para indicarle al usuario una acción.
- Metodo doInbackground(): Se ejecutan todos los procesos deseados en segundo plano, en mi caso, ejecuté el método que me regresaba la lista de proyectos.

- Metodo onPostexecute(): Que se va a hacer después de terminada la acción realizada en segundo plano, en mi caso, guarde mi lista de proyecto y la mande a la vista, al terminar desaparecí el *Dialog,* de esta manera el usuario sabe que ya termino.
- Metodo onCancelled(): Procesos a realizar en caso de que la acción haya sido cancelada.

Dentro del método *doInBackground,* explicado anteriormente, se encuentra este otro método que regresa un *ArrayList,* el cual invoca un método de la clase Parse y recibe como parámetro el id de usuario.

```
public ArrayList proyectos(int usuario_id){
    return new Parse(proyecto.this).proyectos(usuario_id);
}
```

El siguiente método es proyectosAsignados el cual hace la consulta a la base de datos del dispositivo móvil, en el cual se llena una lista con el cursor que regresa la consulta de la base de datos.

```
public ArrayList proyectosAsignados(int usuario_id){
   ArrayList proyectos = new ArrayList();
   //gema_proyecto gemaProyecto = null;
   try{
      Cursor c = dbAdapter.proyectosAsignados(usuario_id);
      if (c != null ) {
         if  (c.moveToFirst()) {
            do {
        proyectos.add(c.getString(c.getColumnIndex("proyecto_descripcion")));

            }while (c.moveToNext());
         }
      }
      //dbAdapter.closeDB();
   }catch(NullPointerException e){
      e.printStackTrace();
   } catch (Exception e) {
      e.printStackTrace();
   }
   return proyectos;
}
```

Aquí se encuentra la consulta a la base de datos, es un método que regresa un objeto de tipo Cursor, se logra observar la sintaxis de la consulta en el siguiente fragmento de código:

```
public  Cursor proyectosAsignados(int user){


   return database.rawQuery("select proyecto_descripcion from " +
UtilsConstantes.KEY_TABLE_TABLAPROYECTOS+
         " where usuario_id = '"+user+"';",null);


}
```

Finalmente obtuve el resultado deseado, en la figura 3.5 se puede observar el proceso creado para la obtención de los proyectos por usuario, los cuales se reflejan en la pantalla para la elección de proyectos con el fin de separar las gestiones, tal como se muestra en la figura 3.4.

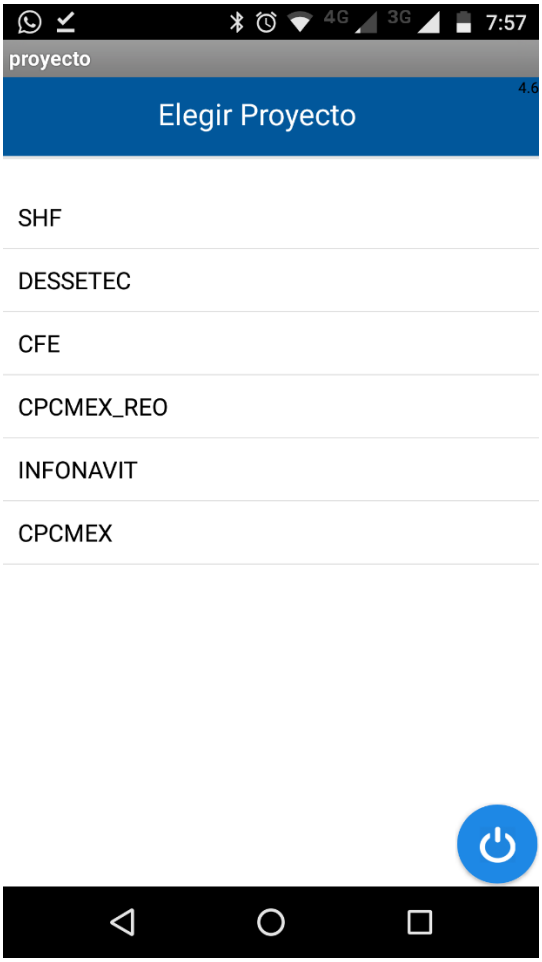


Figura 3.4. Pantalla para seleccionar proyecto.

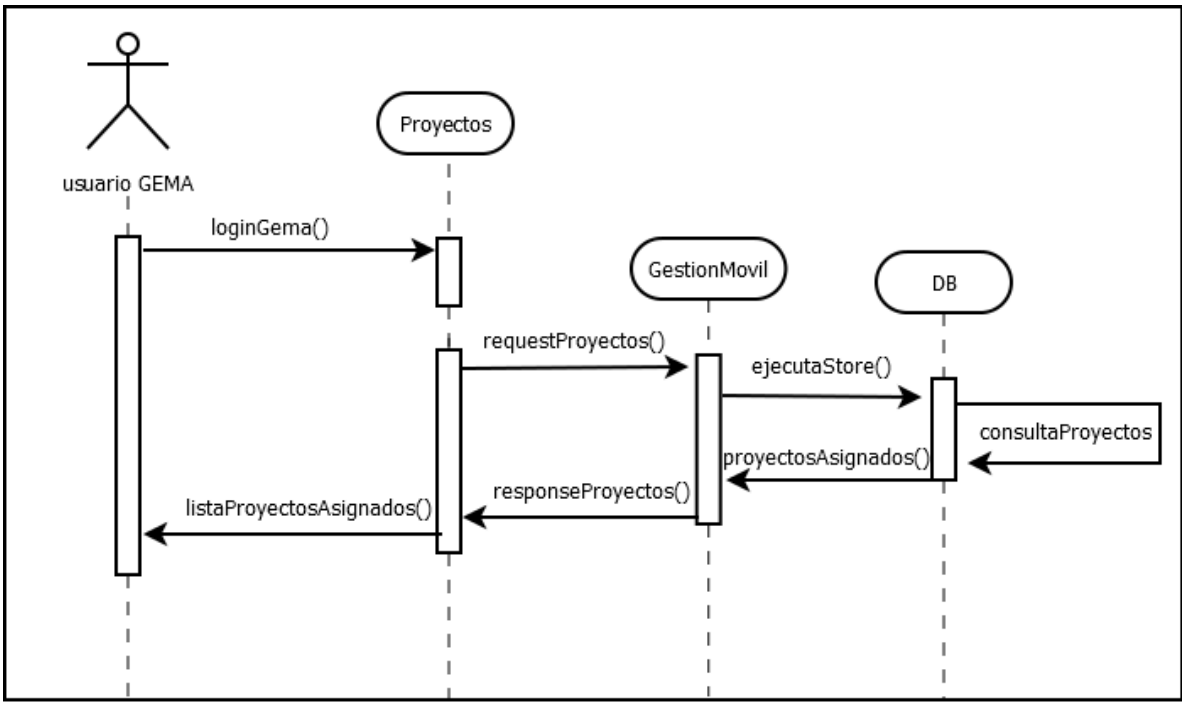
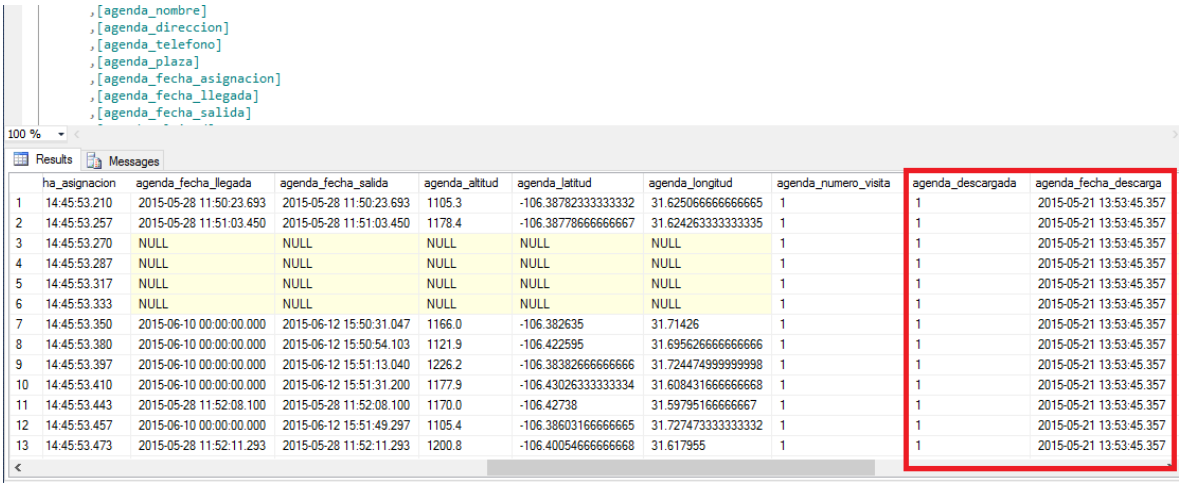


Figura 3.5. Diagrama del proceso de obtención de proyectos GEMA

3.3.3 Solucionar la inestabilidad de la aplicación.

Comencé por realizar una serie de pruebas en modo *Debug,* para identificar los momentos exactos en los que la aplicación se detenía, tras varias pruebas, logré localizar los métodos que hacían morir la aplicación, estos momentos sucedían en la descarga y la subida de las gestiones, es decir, al hacer la petición de la aplicación al servidor para poder descargar la información o al mandarla desde el equipo móvil al servidor, la aplicación se detenía.

El problema ocurría al descargar las gestiones, ya que se marcaban como descargadas antes de asegurar que verdaderamente se hubiesen descargado, como lo muestra la figura 3.6.



```
,[agenda_nombre]
,[agenda_direccion]
,[agenda_telefono]
,[agenda_plaza]
,[agenda_fecha_asignacion]
,[agenda_fecha_llegada]
,[agenda_fecha_salida]
```

100 %

Results | Messages

	ha_asignacion	agenda_fecha_llegada	agenda_fecha_salida	agenda_altitud	agenda_latitud	agenda_longitud	agenda_numero_visita	agenda_descargada	agenda_fecha_descarga
1	14:45:53.210	2015-05-28 11:50:23.693	2015-05-28 11:50:23.693	1105.3	-106.38782333333332	31.625066666666665	1	1	2015-05-21 13:53:45.357
2	14:45:53.257	2015-05-28 11:51:03.450	2015-05-28 11:51:03.450	1178.4	-106.38778666666667	31.624263333333335	1	1	2015-05-21 13:53:45.357
3	14:45:53.270	NULL	NULL	NULL	NULL	NULL	1	1	2015-05-21 13:53:45.357
4	14:45:53.287	NULL	NULL	NULL	NULL	NULL	1	1	2015-05-21 13:53:45.357
5	14:45:53.317	NULL	NULL	NULL	NULL	NULL	1	1	2015-05-21 13:53:45.357
6	14:45:53.333	NULL	NULL	NULL	NULL	NULL	1	1	2015-05-21 13:53:45.357
7	14:45:53.350	2015-06-10 00:00:00.000	2015-06-12 15:50:31.047	1166.0	-106.382635	31.71426	1	1	2015-05-21 13:53:45.357
8	14:45:53.380	2015-06-10 00:00:00.000	2015-06-12 15:50:54.103	1121.9	-106.422595	31.695626666666666	1	1	2015-05-21 13:53:45.357
9	14:45:53.397	2015-06-10 00:00:00.000	2015-06-12 15:51:13.040	1226.2	-106.38382666666666	31.724474999999998	1	1	2015-05-21 13:53:45.357
10	14:45:53.410	2015-06-10 00:00:00.000	2015-06-12 15:51:31.200	1177.9	-106.43026333333334	31.608431666666668	1	1	2015-05-21 13:53:45.357
11	14:45:53.443	2015-05-28 11:52:08.100	2015-05-28 11:52:08.100	1170.0	-106.42738	31.59795166666667	1	1	2015-05-21 13:53:45.357
12	14:45:53.457	2015-06-10 00:00:00.000	2015-06-12 15:51:49.297	1105.4	-106.38603166666665	31.727473333333332	1	1	2015-05-21 13:53:45.357
13	14:45:53.473	2015-05-28 11:52:11.293	2015-05-28 11:52:11.293	1200.8	-106.40054666666668	31.617955	1	1	2015-05-21 13:53:45.357

Figura 3.6. Consulta a la base de datos, donde se muestran los campos afectados al descargar gestiones.

Decidí implementar un método de descarga de gestiones y otro para confirmar la descarga, en el siguiente código se observa que, dentro de una tarea asíncrona, creé el método verificaGestiones el cual se encarga de hacer la llamada al servidor para descargar gestiones pendientes. En dado caso en el que la conexión sea fallida o se interrumpa, elaboré un mensaje para el usuario, de tal manera que sepa que la conexión está fallando.

```
public class GestionDownloadTask extends AsyncTask<Void, Void, Boolean>
{
     protected Boolean doInBackground(Void... params) {
         mResult = verificaGestiones();
         return true;
      }
protected void onPostExecute(final Boolean success) {
mDownloadGestion = null;
showProgress(false);
if (UtilsConstantes.GOOD_SAVE_USER.equals(mResult)) {
   loadList(loaderGestiones());
}
else if (UtilsConstantes.ERROR_NOT_SITE_RESPONSE.equals(mResult))
{
   final Dialog dialog = new Dialog(GestionesPendientesActivity.this,
         getString(R.string.title_dialog_rety),"Compruebe que su equipo
este conectado a internet"
         + "\npresione aceptar para reintentar");
   dialog.setOnAcceptButtonClickListener(new OnClickListener() {
      public void onClick(View v) {
         mDownloadGestion = new GestionDownloadTask();
         mDownloadGestion.execute();
         dialog.dismiss();
       }
   });

   });
```

Dentro del método para verificar gestiones pendientes, se pueden observar dos métodos, gestiones y gestiones_descarga, los cuales hacen una petición al servidor, la primera es para descargar gestiones pendientes, la segunda se ejecuta si el mensaje de la primera no tiene errores, de esta manera garanticé que las gestiones se descargaran correctamente y solo restaba hacer la nueva petición al servidor para marcarlas como descargadas.

```
public String parseToGestiones(int usuario_id, Context context){
if(accessServer()){
      String result =
UtilsMake.refactorStringHtmlToJson(net.gestiones(usuario_id + ""));
      JSONParser jParser = new JSONParser();
        try {
             JSONObject json = (JSONObject) jParser.parse(result);
             String error = (String) json.get("error");
             if(error.equals("0")) {
              String descarga_completa =
UtilsMake.refactorStringHtmlToJson(net.gestiones_descarga(usuario_id +
"",UtilsMake.makeDataPhone(context)));
              if (descarga_completa.equals(UtilsConstantes.UPDATE_SUCCESSFUL))
{
//Inserta el resultado en la base de datos del teléfono.

        return UtilsConstantes.GOOD_SAVE_USER;
              }
         return UtilsConstantes.NOT_NEW_ELEMENTS;
           }
             }  else {
                 return error;
             }
         } catch (ParseException e) {
         e.printStackTrace();
      }
   }
   return UtilsConstantes.ERROR_NOT_SITE_RESPONSE;
}
```

El código genera la siguiente consulta:

```
URL url = new URL(UtilsConstantes.URL_CONTROLLER_+"?p=gestiones &id="+this.user);
```

http://www.cpc.com.mx/GestionMovil/controller/android/HAndroid.ashx?p=gestiones &id=###

La cual manda la instrucción al servidor para descargar gestiones y el id del usuario que las está requiriendo.

El siguiente código es un fragmento del sistema Gestión Móvil, en el cual se observan los métodos userAgendaCuestionarioPreguntas, userAgendaCuestionarios, userProyectos y userAgenda. Estos métodos los elabore para descargar las gestiones, se llena una Lista con los datos necesarios para mandar al equipo móvil, esto es ejecutado por la primera petición explicada anteriormente.

```
      else if (proceso.Equals("gestiones"))
            {
                try
                {
                    List<GemaProyecto_> lista_proyectos =
mIUsuarioService.userProyectos(int.Parse(context.Request.Params["id"]));
                    List<GemaCuestionario_> lista_cuestionario =
mIUsuarioService.userAgendaCuestionarios(int.Parse(context.Request.Params["id"]))
;
                    List<GemaAgenda_> lista_agenda =
mIUsuarioService.userAgenda(int.Parse(context.Request.Params["id"]));
                    List<GemaPreguntas_> lista_preguntas =
mIUsuarioService.userAgendaCuestionariosPreguntas(int.Parse(context.Request.Param
s["id"]));
                    List<GemaPrecargadas> lista_precargadas =
mIUsuarioService.userPreguntasPrecargadas(int.Parse(context.Request.Params["id"])
);
                    result = JsonConvert.SerializeObject(new
                    {
                        error = "0",
                        lista_proyectos = lista_proyectos,
                        lista_cuestionario = lista_cuestionario,
                        lista_agenda = lista_agenda,
                        lista_preguntas = lista_preguntas,
                        lista_precargadas = lista_precargadas
                    });
                    //
mIUsuarioService.actualizaAgendasDownloads(int.Parse(context.Request.Params["id"]
));
                }
                catch (Exception)
                {
                    result = JsonConvert.SerializeObject(new { error = "1" });
                }

            }
```

La segunda petición que programe, es para confirmar las descargas, esta petición se muestra en el siguiente fragmento de código:

```
     else if (proceso.Equals("gestiones_descarga"))
            {
                try
                {


mIUsuarioService.actualizaAgendasDownloads(int.Parse(context.Request.Params["id"]
), context.Request.Params["imei"], context.Request.Params["modelo"],
context.Request.Params["so"]);
                    result = "ok";
                }
                catch (Exception)
                {
                    result = JsonConvert.SerializeObject(new { error = "1" });
                }

            }

//llama al store

        public void actualizaAgendasDownloads(int usuarioId,String imei,String
modelo,String so)
        {
            using (var db = new DataGemaDataContext())
            {


                DataTable dt = new DataTable();
                try
                {
                    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["GestionMovilConnectionStrin
g"].ConnectionString))
                    {
                        SqlCommand sqlComm = new
SqlCommand("sp_actualiza_descargas", connection);
                        sqlComm.Parameters.AddWithValue("@usuario_id",
usuarioId);
                        sqlComm.Parameters.AddWithValue("@imei", imei);
                        sqlComm.Parameters.AddWithValue("@modelo", modelo);
                        sqlComm.Parameters.AddWithValue("@version", so);
                        sqlComm.CommandType = CommandType.StoredProcedure;
                        SqlDataAdapter da = new SqlDataAdapter();
                        da.SelectCommand = sqlComm;
                        da.Fill(dt);
                    }
                }
                catch (Exception) { }

            }
        }
```

Se puede observar que se ejecuta el método actualizaAgendasDownloads, el cual manda a llamar un *Store Procedure,* como se muestra en la figura 3.7, el store que programé recibe ciertos parámetros, en este caso solo usamos el id de usuario, los otros parámetros se agregaron para la funcionalidad que posteriormente explicaré; este procedimiento almacenado, actualiza el estado de las gestiones que ya están descargadas, de esta manera se garantiza que efectivamente, las gestiones estén en el equipo móvil y se evita que la aplicación se detenga, puesto que ya cuenta con los datos necesarios para operar.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:        <Author,,Name>
-- Create date: <Create Date,,>
-- Description:    <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[sp_actualiza_descargas]
    @usuario_id int,
    @imei varchar(50) =null,
    @modelo varchar(50) = null,
    @version varchar(50) = null

    |

AS
BEGIN
    SET NOCOUNT ON;

    UPDATE gema_agenda SET agenda_descargada = 1,
    agenda_fecha_descarga = GETDATE(),imei=@imei,modelo=@modelo,sistema_operativo=@version
    WHERE usuario_id = @usuario_id AND agenda_descargada = 0 AND  estatus_id = 1

    UPDATE [GestionMovil].[dbo].[gema_respuestas_precargadas] SET descargada=1,fecha_descarga=GETDATE()
    where usuario_id=@usuario_id and descargada = 0

    select 'exitoso' as 'exitoso'
END
```

Figura 3.7. Codigo del Store Procedure para actualizar las gestiones descargadas.

El resultado fue exitoso, como vemos en las figuras 3.8 y 3.9, las pantallas para el usuario al momento de descargar sus gestiones, en dado caso de que falle, es posible repetir el procedimiento “n” veces, hasta que la transacción sea satisfactoria, el proceso implementado se puede observar en la figura 3.10.

Figura 3.8. Screenshot de la aplicación Gema al descargar Gestiones

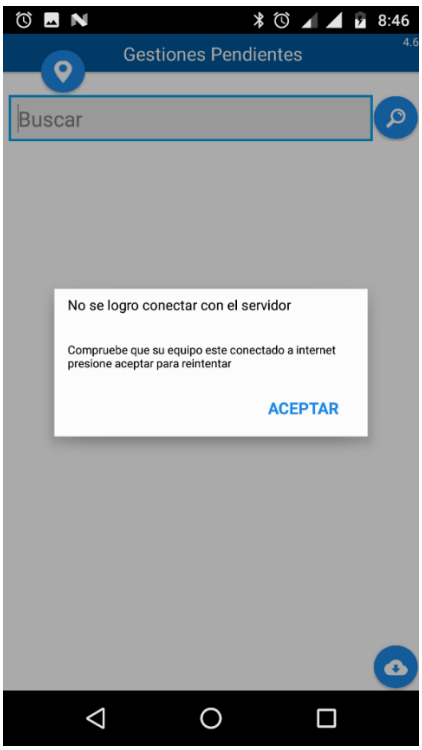


Figura 3.9. Screenshot de Gema al no poder descargar las Gestiones..

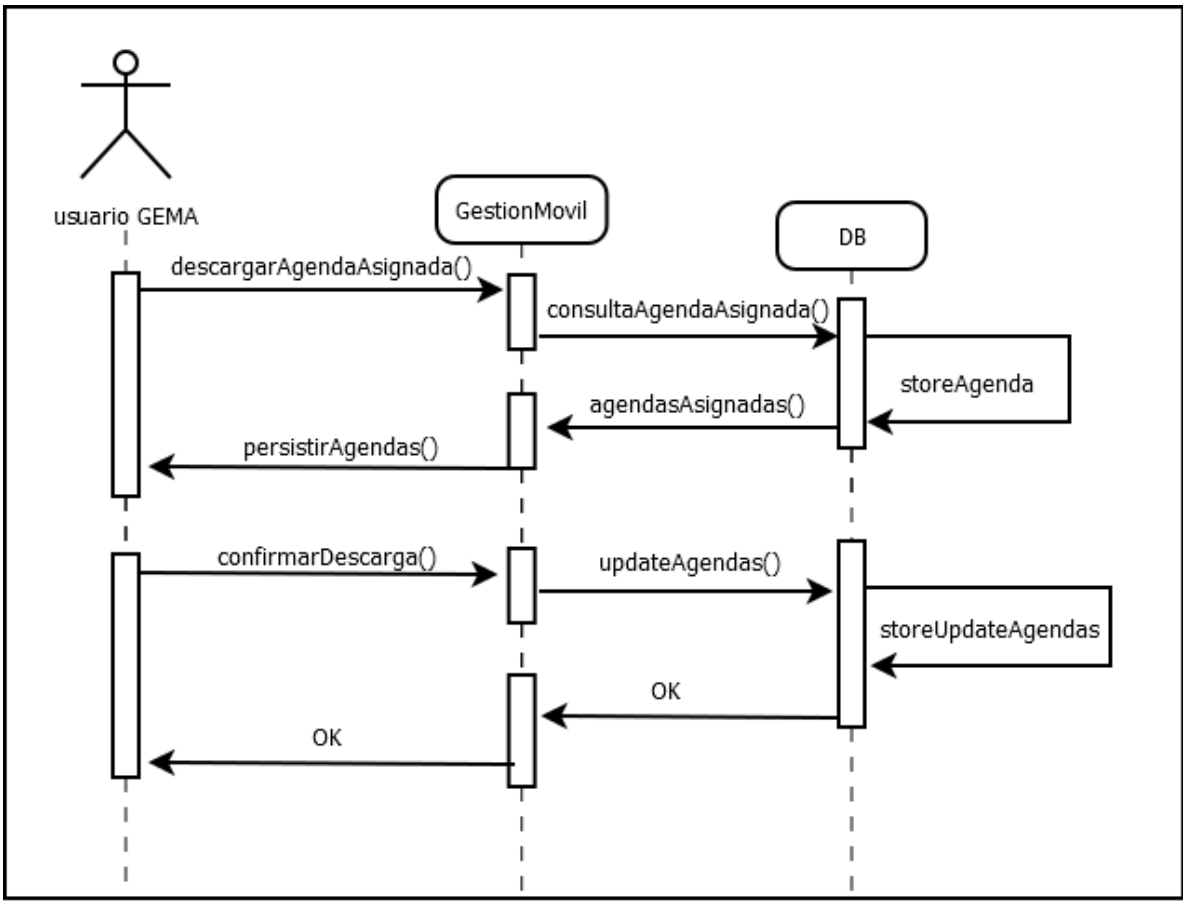


Figura 3.10. Diagrama del proceso de descarga de gestiones.

3.3.4 Implementar cuestionarios con preguntas automáticas, y GEMA sea capaz de determinar el flujo, a partir de estas respuestas.

En el código expuesto anteriormente, se encuentra el método userPreguntasPrecargadas, el cual lleva al código que se encuentra más abajo, en donde mando ejecutar un *store procedure* con los parámetros de id de usuario y caso. Un fragmento del store se encuentra en la figura 3.11.

```
     public List<GemaPrecargadas> getPrecargadas(String Usuario_id)
        {

            using (var db = new DataGemaDataContext())
            {


                DataTable dt = new DataTable();
                try
                {
                    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["GestionMovilConnectionStrin
g"].ConnectionString))
                    {
                        SqlCommand sqlComm = new
SqlCommand("sp_preguntas_precargadas", connection);
                        sqlComm.Parameters.AddWithValue("@Usuario_id",
Usuario_id);
                        sqlComm.Parameters.AddWithValue("@caso", 1);
                        sqlComm.CommandType = CommandType.StoredProcedure;
                        SqlDataAdapter da = new SqlDataAdapter();
                        da.SelectCommand = sqlComm;
                        da.Fill(dt);
                    }
                    try
        {
            List<GemaPrecargadas> list = new List<GemaPrecargadas>();

            foreach (DataRow row in dt.Rows)
            {
                GemaPrecargadas g = new GemaPrecargadas();

            g.agenda_id   = int.Parse(row["agenda_id"].ToString());
            g.agenda_numero_unico = row["agenda_numero_unico"].ToString();
            g.proyecto_id = int.Parse(row["proyecto_id"].ToString());
            g.cuestionario_id= int.Parse(row["cuestionario_id"].ToString());
            g.pregunta_id =   int.Parse(row["pregunta_id"].ToString());
            g.pregunta_texto = row["pregunta_texto"].ToString();
            g.pregunta_respuestas = row["pregunta_respuestas"].ToString();
                g.usuario_id = int.Parse(row["usuario_id"].ToString());

                list.Add(g);
            }

            return list;
        }
        catch
        {
         return  null;
        }
                }
                catch (Exception) { return null; }
            }
        }
```

Este *SP*, es un poco complejo y dado a lo extenso, no es posible capturar todo el código, dicho procedimiento lo elabore para que, a partir del id de usuario que está haciendo la petición para descargar sus gestiones, obtenga los "agenda_id" de todas sus gestiones pendientes y busque en diferentes bases de datos, las respuestas de las preguntas que tienen la bandera de auto encendida, es una consulta bastante compleja dado que involucra diferentes bases de datos, el resultado de dicha consulta es una tabla con las preguntas y respuestas que deben estar lista para descargarse, cabe señalar que este procedimiento debe realizarse en un tiempo relativamente corto, pues se generan en el instante en el que el usuario intenta descargar gestiones pendientes desde la aplicación móvil.

```
SELECT pregunta_id,case  when pregunta_id in (5) then (select case when ARC_DESC_ANEC like '%CON ESCRITURA%' then 'Si' else 'No' end from [sv-cpc-datos
                                    when pregunta_id in (35,37,39,16,20,24) then (select case when agrupador like '%PREJURIDICO%' then 'Si' else '
                                    when pregunta_id in (32) then (select case when Producto_ARC = 'REA' then 'Si' else 'No' end  from [sv-cpc-dat
                                    when pregunta_id in (25) then (select case when Omisos > 12 then 'Si' else 'No' end  from [sv-cpc-datos].[sicv
                                    when pregunta_id in (26) then (select case when REPLACE ( ARC_POR_TXT,'%','')   > 50 then 'Si' else 'No' end
                                    when pregunta_id in (21,40,27) then (select case when @TOTAL >= 2 then 'Si' else 'No' end )
                  /*Analizar*/      when pregunta_id in (14) then (select case when @total3 >= 2 then 'Si' else 'No' end )
                  /*Analizar******/ when pregunta_id in (15) then (select case when @total2 >= 2 then 'Si' else 'No' end)
                  /*Analizar*****/  when pregunta_id in (11,38,36) then (select case when count(Id_Proceso) >= 1 then 'Si' else 'No' end from  [sv
                  /*Analizar*****/  when pregunta_id in (33) then (select case when count(agenda_id) =2 then 'B' when count(agenda_id) =3 then 'C'
                                    else ''  end
  FROM [GestionMovil].[dbo].[gema_preguntas]
  where cuestionario_id = 1008 and pregunta_automata = 1


insert into gema_respuestas_precargadas
select @agenda_id,@agenda_numero_unico,@proyecto_id,@cuestionario_id,GP.pregunta_id,pregunta_texto,VT.respuesta,@Usuario_id,0,null from gema_preguntas
inner join @var_tabla VT on VT.pregunta_id=GP.pregunta_id
where cuestionario_id=1008 and pregunta_automata=1
```

Figura 3.11. Fragmento de Código del Store Procedure creado para generar las respuestas Automaticas.

En la figura 3.12 se puede apreciar el resultado del *SP* anterior, en el cual se cuenta con una relación entre el "agenda_id" que identifica el crédito, las preguntas y respuestas que ya deben estar precargadas y listas para descargarse en el equipo móvil, de tal manera que, la base de datos del equipo móvil cuente con las respuestas necesarias para determinar el camino a seguir, sin necesidad de que el usuario ingrese una respuestas, y todo esto es información que se procesó y que se tiene en las bases de datos de los sistemas de CPC.

	agenda_id	agenda_numero_unico	proyecto_id	cuestionario_id	pregunta_id	pregunta_texto	pregunta_respuestas	usuario_id	descargada	fecha_descarga
1	456317	100009136	1004	1008	5	¿La vivienda tiene escritura?	Si	2	1	2017-02-27 09:42:16.2
2	456317	100009136	1004	1008	11	¿Ya paso por proceso de mediación?	No	2	1	2017-02-27 09:42:16.2
3	456317	100009136	1004	1008	14	¿Tiene Convenios privados incumplidos?	No	2	1	2017-02-27 09:42:16.2
4	456317	100009136	1004	1008	15	¿Tiene Convenios judiciales incumplidos?	No	2	1	2017-02-27 09:42:16.2
5	456317	100009136	1004	1008	16	¿Esta en etapa pre-jurídica?	No	2	1	2017-02-27 09:42:16.2
6	456317	100009136	1004	1008	20	¿Esta en etapa pre-jurídica?	No	2	1	2017-02-27 09:42:16.2
7	456317	100009136	1004	1008	21	¿Tiene dos o más convenios privados incumplidos?	No	2	1	2017-02-27 09:42:16.2
8	456317	100009136	1004	1008	24	¿Esta en etapa pre-jurídica?	No	2	1	2017-02-27 09:42:16.2
9	456317	100009136	1004	1008	25	¿Tiene más de 12 omisos?	Si	2	1	2017-02-27 09:42:16.2
10	456317	100009136	1004	1008	26	¿Tiene más de 50 % de pagos efectivos?	Si	2	1	2017-02-27 09:42:16.2

Figura 3.12. Tabla que muestra las respuestas precargadas y generadas por el Store Procedure, ligadas a una Gestión.

La estructura del cuestionario a contestar, guarda dentro de él, el camino que debe seguir dependiendo de la respuesta que se le otorgue, es decir, dependiendo de la respuesta puede continuar a la siguiente pregunta o saltar a un número de pregunta cualquiera, la lógica está en la columna “pregunta_condicion”, donde se separa mediante una “,” el número de pregunta al cual seguirá, por ejemplo: “v_NO, 50”, cuando se detecta que en esa pregunta la respuesta es “NO“, saltara a la pregunta con número 50, si está vacía la condición se infiere que seguirá a la pregunta consecutiva, el proceso completo se logra observar en la figura 3.14. Además de esto, también implemente un tipo de pregunta, las columnas se pueden observar en la figura 3.13, hay tres tipos:

- Pregunta requerida: El usuario no pueden seguir contestando el cuestionario si no se obtiene una respuesta a la pregunta con esa bandera activada.
- Pregunta auto: Es una pregunta que se conoce y se le muestra al usuario con la respuesta ya definida, puede o no ser editada.
- Pregunta autómata: Es un tipo de pregunta cuya respuesta esta precargada y es totalmente transparente para el usuario, la cual a partir de su respuesta determina el camino que debe seguir, es decir, la siguiente pregunta que le aparecerá al usuario, cabe señalar que pueden ser desde 1 hasta n preguntas autómatas que se contesten antes de que el usuario intervenga.

100 %

Results | Messages

		tipo_id	pregunta_respuestas	pregunta_condiciones	pregunta_requerida	pregunta_auto	pregunta_automata
1		1	Si|No	v_No, 50|	0	0	0
2		1	Casa Habitación|Departamento|Otro	v_Casa Habitación, 4|v_Departamento,4|	0	0	0
3	ble	2			0	0	0
4	de estar en una privada?	1	Si|No	v_No, 50|	0	0	0
5	localización	2			0	0	0
6		1	Si|No	v_No, 7|v_Si, 8|	0	0	0
7		1	Abarzonada|Vandalizada|Abandonada|Destruida|Tapi...	v_Abarzonada,9|v_Vandalizada,9|v_Abandonada,9|v_...	0	0	0
8		1	Abarzonada|Prestada|Rentada|Invadida	v_Abarzonada,38|v_Prestada,38|v_Rentada,38|v_Inva...	0	0	0
9	da?	1	Si|No		0	0	0
10	o?	1	Si|No		0	0	0

Figura 3.13. Tabla que muestra el tipo de pregunta por cuestionario.

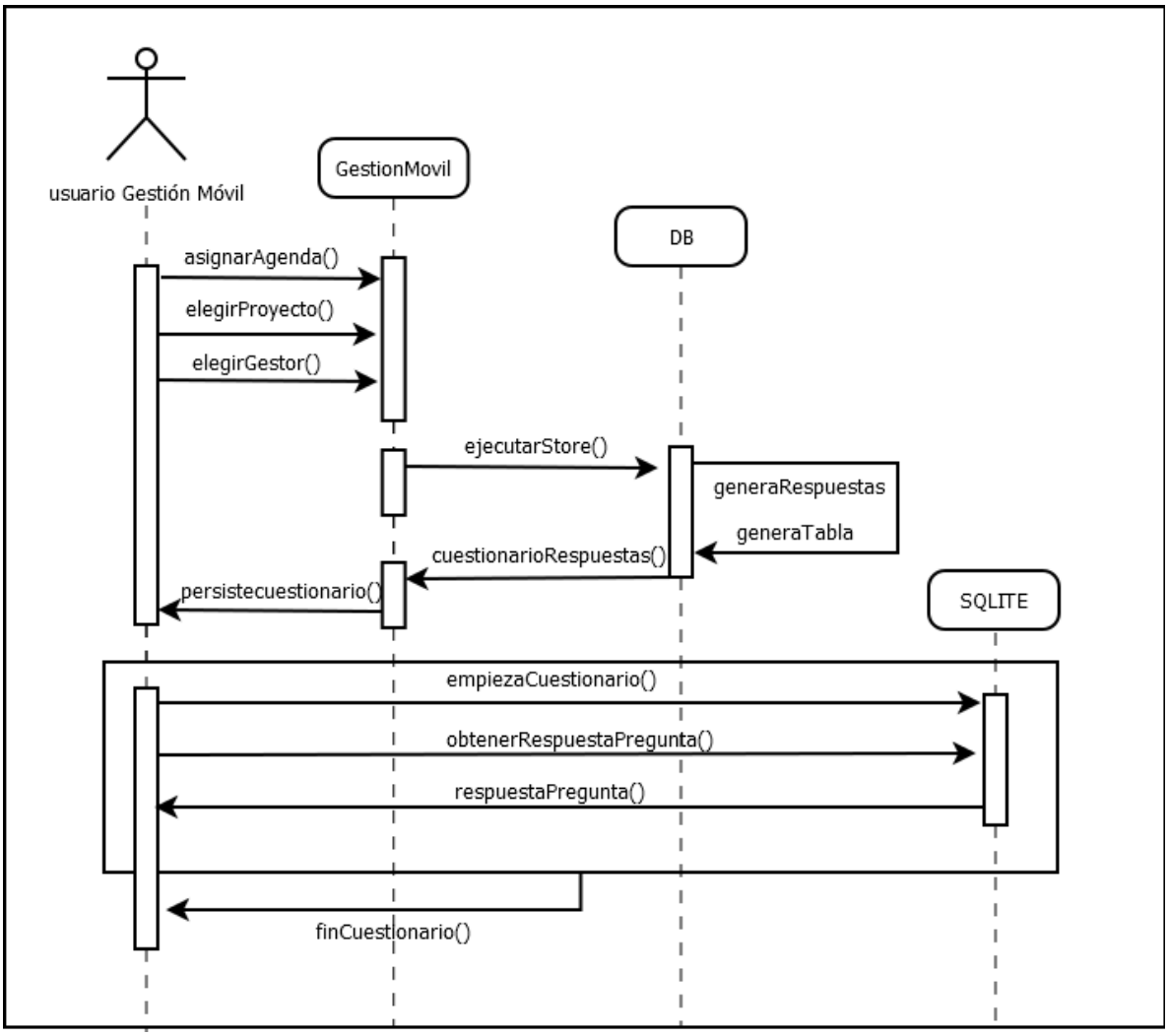


Figura 3.14. Proceso implementado para el flujo de preguntas en GEMA

3.3.5 Incluir un soporte remoto, para facilitar la atención a usuarios y reducir el tiempo en la solución de problemas. (Iniciativa propia).

Este desarrollo, fue una iniciativa propia, ya que tenía la intención de aprender a usar los servicios web, comencé por programar el servicio, en el siguiente código podemos observar la sentencia [SoapHeader("Authentication", Required = true)], la cual indica que mi *WS* será de tipo SOAP, lo elegí de esta manera por la seguridad, *REST* usa *HTTPS,* mientas que *SOAP* usa *WS SECURITY,* en ese momento los servidores de CPC no contaban con certificado de seguridad. Primeramente recibo los parámetros que se envían desde el cliente, los cuales son el usuario, la contraseña y una cadena que indica el procedimiento a realizar. Una vez se validan los datos, se obtiene el nombre del

archivo, y se envía a una ruta establecida en el servidor, hasta este momento ya se tenían todos los datos del equipo móvil en el servidor.

```
[WebService(Namespace = "http://www.cpc.com.mx/")]
public class WebServiceSupport : System.Web.Services.WebService
{
    public AuthHeader Authentication;
    [WebMethod]
    [SoapHeader("Authentication", Required = true)]
public String SensitiveData(String upload, String file, String fileName)
    {
if (Authentication.Username == "AppMovil@14" &&
                Authentication.Password == "Soporte@41")
        {
          if (upload.Equals("upload"))
          {
            String path = WebConfigurationManager.AppSettings["PATH_SUPP"] + fileName + ".zip";
if (File.Exists(path))
              {
              byte[] buffer = System.Convert.FromBase64String(file);
              File.WriteAllBytes(path, buffer);
            }
            send(path);
            return "correcto";
          }
          return "error";
        }
        else
        }
          return "Error";

        }
    }
}
```

El siguiente proceso que desarrollé del cual el código se muestra más abajo, fue, al momento de subir el archivo, notificar a los programadores correspondientes mediante un correo electrónico, que se necesitaba un soporte, y se les envía la ruta del archivo, es importante notar que, en el método smtpClient.SendAsync, indico que se enviará de manera asíncrona, de forma tal que, el usuario termine el proceso de carga, se desbloquee su equipo para que lo pueda seguir utilizando y el resto de la tarea se la deje al servidor, si no se hiciera esto, ocurriría el caso en el que los correos están lentos o encolados y los usuarios no recibirían respuesta del *WS,* entonces se detiene su aplicación.

```
   public bool send(String filename)
      {
            SmtpClient smtpClient = new SmtpClient();

            NetworkCredential basicCredential = new
NetworkCredential(WebConfigurationManager.AppSettings["MAILEMISOR"], "l@passw0rd");

            MailMessage message = new MailMessage();

            MailAddress fromAddress = new MailAddress(WebConfigurationManager.AppSettings["MAILEMISOR"],
"GEMA SUPPORT");

            smtpClient.Host = WebConfigurationManager.AppSettings["MAILHOST"];

            smtpClient.Credentials = basicCredential;

            smtpClient.Port = 25;

            smtpClient.EnableSsl = false;

            message.From = fromAddress;
            message.Subject = "Asunto";

            message.IsBodyHtml = true;

            message.Body = "! Necesitan un soporte ¡, con ubicación de archivo:  \n\n" + filename;


            message.To.Add(WebConfigurationManager.AppSettings["MAILRECEPTOR1"]);
            message.To.Add(WebConfigurationManager.AppSettings["MAILRECEPTOR2"]);


            smtpClient.SendAsync(message,"");

         }
         return false;
      }
```

Continué desarrollando el cliente que consumirá el *WS*, en el código de abajo se pueden observar los parámetros necesarios para que el cliente pueda alcanzar el servicio web, estos datos pueden consultarse en el *WSDL* (Web Service Description Lenguage) y los descompuse en 4 variables:

- Método: Contiene el nombre del método que va a consumir.
- namespace: Contiene el espacio de nombres, el cual se encuentra al inicio del Servicio.
- accionSoap: se conforma de Namespace + Método.
- url: Contiene la dirección en donde se encuentra alojado el servicio Web.

```
  // Empieza Soap Services parametros
    // Metodo que queremos ejecutar en el servicio web
    private static final String Metodo = "SensitiveData";
    // Namespace definido en el servicio web
    private static final String namespace = "http://www.cpc.com.mx/";
    // namespace + metodo
    private static final String accionSoap =
"http://www.cpc.com.mx/SensitiveData";
    // Fichero de definicion del servcio web
    private static final String url =
UtilsConstantes.URL_SITE+"controller/android/WebServiceSupport.asmx";

    //Termina Soap Services parámetros
```

Otro proceso importante que elaboré, es cifrar la información en Base64, es importante que la información que viajará a través de un canal que no es seguro, viaje cifrada, esto con el fin de evitar que los datos sensibles estén expuestos y visibles a entes no autorizados, el motivo por el cual utilicé Base64, además de su sencillez para implementarlo, es para simular la autenticación básica del protocolo http, la cual también usa Base64. En el siguiente fragmente de código, se muestra el método Base64.*encode*, el cual recibe en bytes el archivo a cifrar, y regresa la cadena en Base64.

```
private String encodeFileToBase64Binary(File file){
     String encodedfile = null;
         byte[] bytes = new byte[(int)file.length()];
fileInputStreamReader.read(bytes);
         encodedfile = com.gema.app.utils.security.Base64.encode(bytes);
     return encodedfile;
 }
```

Finalmente, para enviar la información al servicio web, desarrolle el método soapService, el cual recibe el archive ya cifrado y el nombre con el cual se va a guardar, creé un objeto SoapObject, el cual recibe como parámetros el espacio de nombres y el método, funciona como un *request,* le añadí las propiedades que necesitaba, una vez elaborado el objeto a mandar, también configure la seguridad en el *header* se le añade el usuario y contraseña para poder consumir el servicio web, finalmente envolví todo en un sobre, ya que los *WS* de tipo Soap, reciben objetos de tipo SoapSerializationEnvelope, al cual le indiqué mediante el método .doNet, que el servicio web estaba programado en .Net.

```
public boolean  soapService(File zip,String name)
{
       String archivo=encodeFileToBase64Binary(zip);

   try {
      SoapObject request = new SoapObject(namespace, Metodo);
      request.addProperty("upload", "upload"); // Paso parametros al WS
      request.addProperty("file", archivo);
      request.addProperty("fileName", name);

      /*autentification header*/
      Element[] header = new Element[1];
      header[0] = new Element().createElement(namespace,"AuthHeader");
      Element username = new Element().createElement(namespace, "Username");
      username.addChild(Node.TEXT, "AppMovil@14");
      header[0].addChild(Node.ELEMENT, username);
      Element password = new Element().createElement(namespace, "Password");
      password.addChild(Node.TEXT, "Soporte@41");
      header[0].addChild(Node.ELEMENT, password);
      /*autentification header*/
      // Modelo el Sobre
      SoapSerializationEnvelope sobre = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
      sobre.dotNet = true;
      sobre.setOutputSoapObject(request);
      sobre.headerOut = header;

      HttpTransportSE transporte = new HttpTransportSE(url);

      transporte.call(accionSoap, sobre);

      SoapPrimitive resultado = (SoapPrimitive) sobre.getResponse();

      if (resultado.toString().equals("correcto"))
         return true;
         return false;
   } catch (Exception e) {
   return false;
   }
}
```

Se logra observar en las figuras 3.15, 3.16 y 3.17, el resultado del desarrollo que implemente, de esta manera logré reducir drásticamente el tiempo que consumía realizar un Soporte y también, actualice la manera en la cual la aplicación se comunicaba con el servidor, ya que los servicios web, son herramientas muy utilizadas, y más recientes a comparación del *hanlder* utilizado en la aplicación del cual ya presenté varios ejemplos. De igual manera en la figura 3.18 se describe el proceso que se realiza para pedir un soporte.

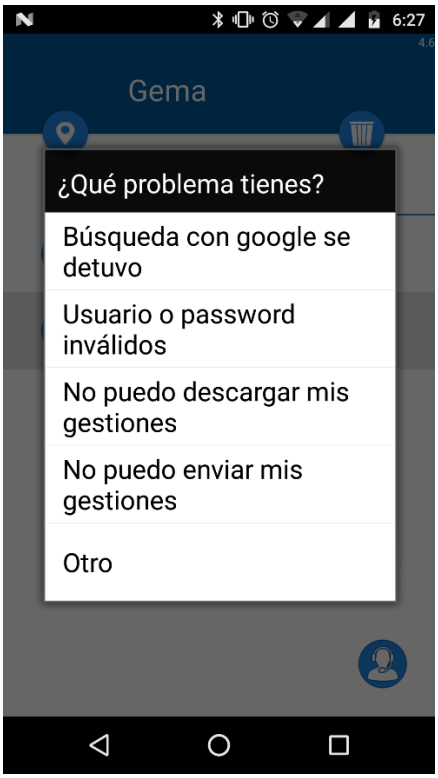


Figura 3.15. Soporte Técnico Remoto en la Aplicación Gema. Parte 1

Figura 3.16. Soporte técnico Remoto en Aplicación Gema. Parte 2

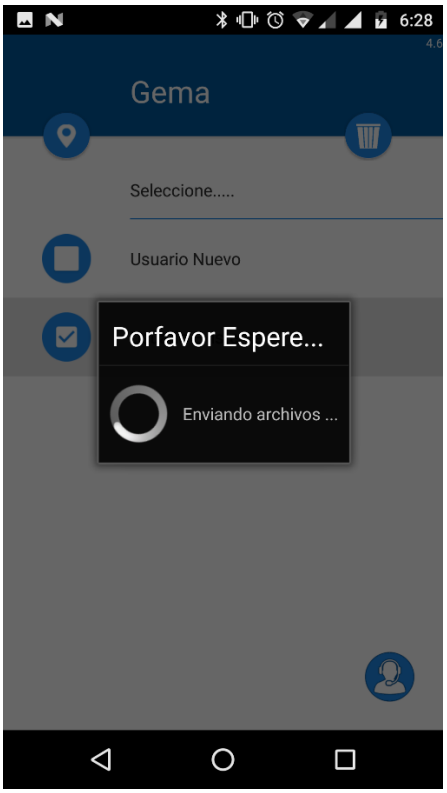


Figura 3.17. Soporte técnico Remoto en Aplicación Gema. Parte 3.

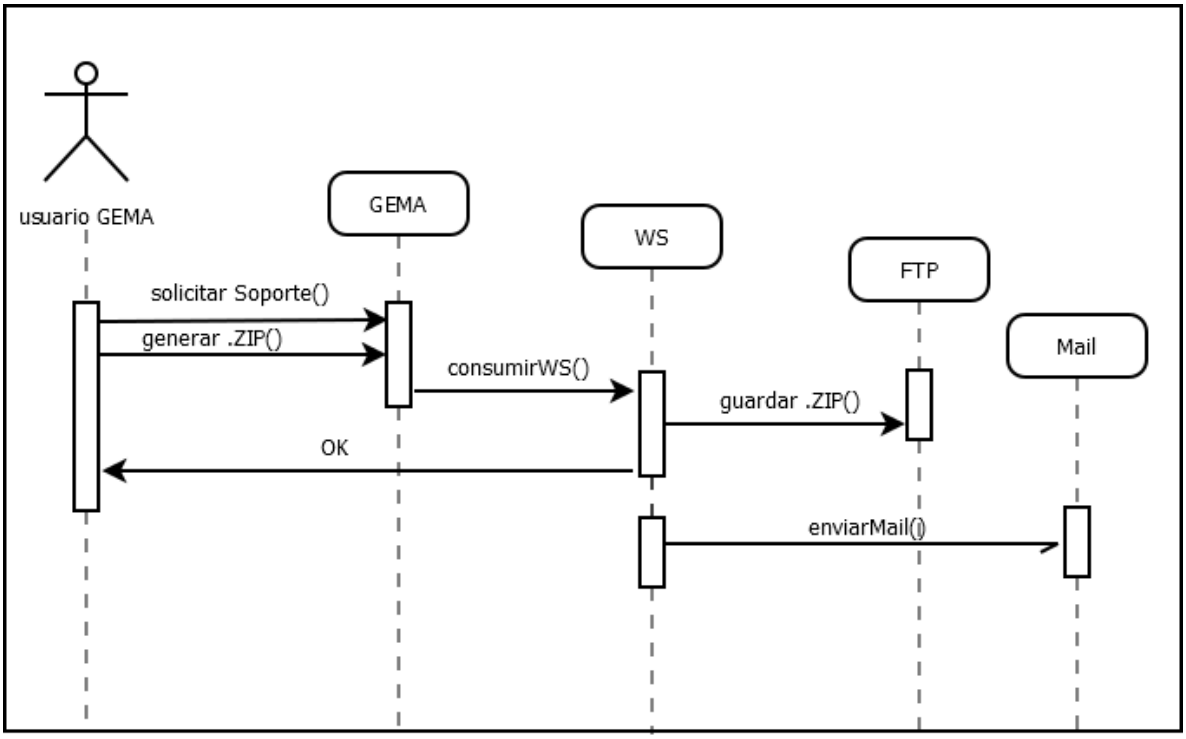


Figura 3.18. Proceso implementado para el envío de datos de GEMA al servidor

3.3.6 Registro de datos del equipo utilizado para gestionar.

Para cumplir con este objetivo, lo dividí en dos procesos, en el primero agregue en la petición de "login" el método mIUsuarioService.saveMovil, el cual recibe de parámetros, el modelo, el *imei* y el sistema operativo del dispositivo que se está autenticando en la aplicación.

```
if (proceso.Equals("login"))
            {
                try
                {
                    var gemaUsuario = new GemaUsuario_();
                    gemaUsuario.usuario_usuario = context.Request.Params["user"];
                    gemaUsuario.usuario_key = context.Request.Params["key"];
                    var user = mIUsuarioService.authenticate(gemaUsuario);
                    if (user != null)
                    {
                            if
(user.usuario_key.Equals(context.Request.Params["key"].Trim()))
                            {
                                result = JsonConvert.SerializeObject(new { error
= "0", usuario = user });
                            }
                        }

                        mIUsuarioService.saveMovil(user,
context.Request.Params["imei"], context.Request.Params["modelo"],
context.Request.Params["so"]);


                    }
                    else
                    {
                        result = JsonConvert.SerializeObject(new { error = "1"
});
                    }
                }
                catch (Exception)
                {
                    result = JsonConvert.SerializeObject(new { error = "2" });
                }
            }
```

Dentro del método saveMovil, inserte dichos datos en la tabla gema_movil, lo cual se muestra en el fragmento de código siguiente:

```
        internal bool saveMovil(int usuario_id, string imei, string modelo, string
so)
        {
            using (var db = new DataGemaDataContext())
            {
                try
                {
                    var movil = new gema_movil();
                    movil.usuario_id = usuario_id;
                    movil.movil_imei = imei;
                    movil.movil_modelo = modelo;
                    movil.movil_so = so;
                    movil.movil_fecha = DateTime.Now;
                    db.gema_movil.InsertOnSubmit(movil);
                    db.SubmitChanges();
                    return true;
                }
                catch (Exception)
                {
                }
            }
            return false;
        }
```

Esta parte del código es diferente a lo antes visto, ya que se están persistiendo los datos con ayuda de *LINQ,* la tabla está dentro de un modelo que LINQ ya conoce, le asigné los datos que necesitaba persistir y basta con seguir la sintaxis específica del *framework*. El resultado se muestra en la figura 3.19, donde se observa que ya se cuenta con un registro de datos del equipo móvil que está siendo utilizado para la aplicación GEMA, de manera que resulte un mayor control sobre los dispositivos.

	id_registro	version_apk_instalada	imei	sistema_operativo	modelo	numero_linea	compañia	fecha
1	1	4.6	358966066590084	6.0.1-23	Motorola XT1563		TELCEL	2016-10-17 19:04:53.027
2	3	4.5	353205055316820	4.1.2-16	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
3	4	4.5	352222073252227	5.0.2-21	Samsung SM-G530H		TELCEL	2016-10-17 19:04:53.027
4	5	4.5	352691072606627	5.1-22	LANIX Ilium LT500		TELCEL	2016-10-17 19:04:53.027
5	6	4.5	353205055311896	4.4.2-19	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
6	1002	4.6	353205056231549	4.4.2-19	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
7	1003	4.5	353205053997688	4.1.2-16	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
8	1004	4.6	353205056250234	4.1.2-16	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
9	1005	4.5	353205055308652	4.4.2-19	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
10	1006	4.5	353205056249855	4.1.2-16	Motorola XT914		TELCEL	2016-10-17 19:04:53.027
11	1007	4.5	353205056223793	4.1.2-16	Motorola XT914		TELCEL GSM	2016-10-17 19:04:53.027

Figura 3.19.Tabla que guarda registros de los dispositivos Moviles.

La segunda parte del proceso, forma parte del *SP* de descarga de gestiones, que se explicó anteriormente, en el cual, al momento de hacer la petición para descargar las gestiones pendientes, inserté para cada gestión, el *imei* y el modelo del dispositivo que las descargó, de esta manera logré ligar el último usuario que utilizó el dispositivo para gestionar, las gestiones que se han realizado desde un equipo móvil especifico y todos los detalles del mismo, teniendo en cuenta que el *IMEI,* es único. La figura 3.20 muestra parte de la tabla de gestiones, en donde se aprecian las columnas de IMEI, usuario y agenda, los tres datos importantes para llevar un control de dispositivos y gestiones. Así el equipo de soporte técnico, que se encarga de administrar los dispositivos, tiene información suficiente para contactar con los usuarios responsables de los dispositivos móviles.

	agenda_id	usuario_id_asig	usuario_id	cuestionario_id	agenda_numero_unico	imei
1	11032	99	91	1	221001000035	353205056231663
2	348355	2166	2311	1005	603227370	359587073776372
3	348358	2166	2311	1005	603209052	359587073776372
4	348398	2166	2311	1005	60368067	359587073776372
5	348432	2166	2311	1005	603206001	359587073776372
6	357307	2166	2311	1005	603110763	359587073776372
7	357311	2166	2311	1005	603112939	359587073776372
8	358840	2166	2311	1005	603228792	359587073776372
9	362744	2166	2311	1005	603228792	359587073776372
10	362811	2166	2311	1005	603228792	359587073776372
11	362890	1	2	1005	611268736	358966066590084

Figura 3.20. Tabla que muestra los campos ligados entre Gestión y dispositivo Móvil.

3.3.7 Actualizaciones automáticas y obligatorias de GEMA.

El último de los objetivos que programé, fue la descarga automática de actualizaciones, este proceso redujo importante tiempo en la instalación de nuevas versiones para los usuarios. Comencé por crear el método newVersion dentro de una tarea asíncrona, la cual se muestra a continuación:

```
public class downloadVersion extends AsyncTask<Void, Void, Boolean> {
    private gema_usuario gemaUsuario;
    downloadVersion () {
       gemaUsuario = null;
    }
    protected Boolean doInBackground(Void... params) {
       if(!newVersion()){
                          isNewVersion = true;
       }else{
          gemaUsuario = UtilsLock.getLock(SplashScreenActivity.this);
       }
        return true;
    }
    protected void onPostExecute(final Boolean success) {

       UtilsLogger.e(getApplicationContext(), "LockTask",
"newVersion..."+isNewVersion);
       if(!isNewVersion){
         if (gemaUsuario != null) {
           switchInActivity(gemaUsuario);
         } else{
           switchInActivityUsers();
         }
       }else{
           mDownTask = new download();
          mDownTask.execute();
       }
        mLockTask = null;
    }
    protected void onCancelled() {
    }
}
```

Este método, realiza una petición al servidor, el cual regresa true o false, en dado caso la versión instalada sea menor a la versión registrada y disponible en el sistema, el código de esta petición se muestra a continuación:

```
  private boolean newVersion(){
   UtilsLogger.e(getApplicationContext(), "newVersion", "newVersion...");
   if(UtilsBegin.isConnection(getApplicationContext())){
       String version_ = UtilsBegin.newVersion(getApplicationContext());
   if(!version_.equals(UtilsConstantes.ERROR_NULL)){
      urlDownload = version_ ;
      return false;
   }
   }
return true;
  }
```

Si la versión es menor a la del sistema, la tarea asíncrona antes mostrada, termina e inicia una nueva, la cual se muestra en el siguiente fragmento de código:

```
//Metodo Download
public class download extends AsyncTask<Void, Void, Boolean> {
        private gema_usuario gemaUsuario;
        private ProgressDialog mDialog;
        download() {
            gemaUsuario = null;
        }
protected  void onPreExecute(){
    mDialog = ProgressDialog.show(SplashScreenActivity.this,"Descargando...",
"Se ha detectado una nueva versión, Espere ...", true);
   }
        protected Boolean doInBackground(Void... params) {
                new DownloadFile(getApplicationContext(), urlDownload);

            return true;
        }
        protected void onPostExecute(final Boolean success) {
            mDialog.dismiss();
            UtilsLogger.e(getApplicationContext(), "LockTask",
"newVersion..."+isNewVersion);
((TextView)findViewById(R.id.textView1)).setText(getString(R.string.error_need
_version));
                new SnackBar(SplashScreenActivity.this,
getString(R.string.error_yet_version));

            mDownTask = null;
        }
    }
```

En esta tarea se declara el método DownloadFile el cual inicia la descarga de las actualizaciones necesarias para la nueva versión de GEMA, mientras en primer plano, al usuario se le muestra un mensaje de progreso y descarga de versión, el cual se muestra en las figuras 3.22 y 3.23. Para realizar la descarga, se crea una petición al servidor como se muestra a continuación:

```
    URL url = new
URL(UtilsConstantes.URL_CONTROLLER_+"?p=download&v="+this.version);
```

Ese fragmento de código, genera la siguiente petición: http://www.cpc.com.mx/GestionMovil/controller/android/HAndroid.ashx?p=download&v=4.5 . La cual si nosotros la probamos en el explorador, nos regresa el archivo .apk de la nueva versión de GEMA, tal como se muestra en la figura 3.21.

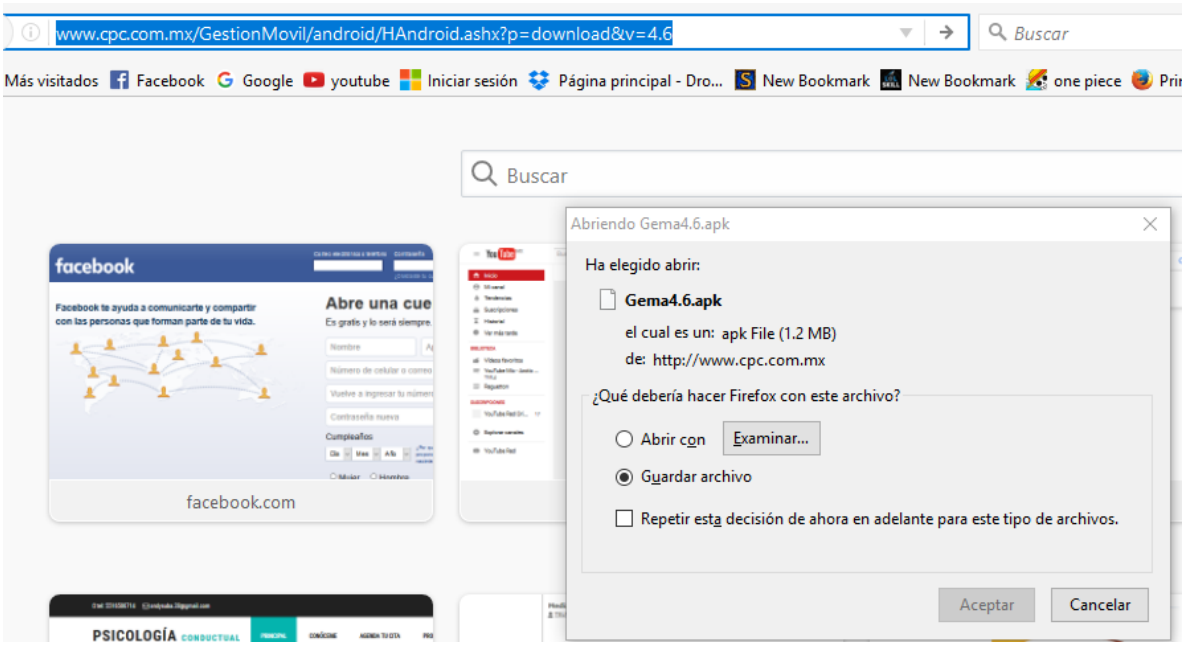


Figura 3.21. Petición al servidor para descargar la versión actual de Gema.

Figura 3.22. Screenshot del proceso de descarga de Gema

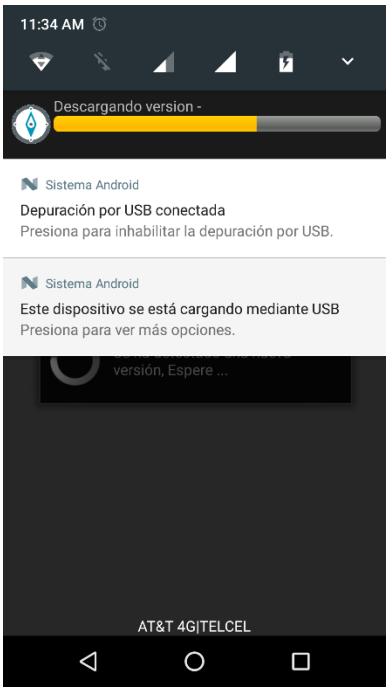


Figura 3.23. Screenshot del progreso de descarga de Gema.

El resultado de este proceso fue contar con la versión actualizada de GEMA, de manera que se contenga el índice de errores y se apliquen las correcciones pertinentes a la brevedad. Con este último desarrollo y en conjunto con los anteriores, logre después de 6 meses, dejar a GEMA, como una aplicación competitiva y estable, lista para salir al mercado.

Capítulo IV.

Resultados

4.1 Resultados.

4.1.1 Sobre Gema.

La aplicación le fue presentada a los dueños de CPC y directivos de cada proyecto, tras diversas sesiones para afinar los objetivos de la aplicación y poder ofrecerla como una solución IT en el mercado, GEMA fue aceptada con éxito y se decidió presentar la nueva aplicación móvil a uno de los clientes mayoritarios de CPC. Como explique anteriormente, dicho cliente obliga a las empresas que contrata para gestionar su cartera vencida, a usar la aplicación de bluemessaging, en un intento por reducir gastos y aumentar sus ganancias, CPC buscaba ofrecer una alternativa, así que se acordó una reunión con el cliente mayoritario para hacer la presentación de GEMA, y convencerlos de usar la aplicación para gestionar la nueva cartera. Debido a que mi estadía en CPC concluyó el 16 de Diciembre del 2016, aproximadamente 20 días después de concluir con todos los cambios y mejoras en la aplicación, desconozco el resultado de la reunión.

Actualmente GEMA cuenta con 6 proyectos a gestionar (CARTERA 1, CARTERA 2, CARTERA 3, CARTERA 4, CARTERA 5, CARTERA 6) los dos últimos se añadieron en fechas posteriores a mi salida de CPC, pero podemos observar la escalabilidad de la aplicación con dos nuevas carteras añadidas, cabe mencionar que una de ellas es el cliente mayoritario, así que existe la posibilidad de que la reunión con ellos haya sido exitosa.

CPC es una consultoría que ofrece servicios de cobranza, GEMA se convirtió en su carta de presentación, pues los clientes, buscan que la empresa que están contratando sea capaz de llevar a cabo su tarea de manera profesional y eficiente, actualmente no hay mejor manera de ofrecer un servicio que con ayuda de la tecnología e innovación. Así que a los nuevos clientes, se les hace mención de la aplicación GEMA para la recuperación de cartera vencida, una demostración del proceso de gestión y los datos que se logran recabar a través de esta tecnología de la información, cabe destacar que CPC es de las pocas empresas de cobranza, que cuenta con una aplicación de esta dimensión, y es un incentivo para atraer nuevos clientes.

Ese es el papel que GEMA juega para CPC, resulto ser una herramienta muy utilizada y eficiente para realizar las gestiones correspondientes, y una carta de presentación para atraer nuevos clientes, se espera que GEMA pueda ser vendida a otras empresas de cobranza, gestionada y distribuida por CPC.

4.1.2 Sobre mi desarrollo profesional.

A partir de la experiencia laboral adquirida, si bien la teoría es fundamental para cualquier profesionista, es muy importante traducir todos esos conocimientos adquiridos en la Universidad y plasmarlos en la vida real, tener la oportunidad de enfrentarte a un problema real y salir avante aplicando lo aprendido, conlleva adquirir experiencia y de esta manera fortalecer tus habilidades como profesionista. Comencé a trabajar aproximadamente cuando cursaba sexto semestre de la carrera, trabajar con un lenguaje de programación orientado a objetos como es C#, solidificó las lecciones aprendidas en las materias enfocadas a programación, esto me ayudó mucho para materias venideras, en las cuales yo ya contaba con conocimientos y práctica al respecto. Como estudiante de ingeniería, se incrementó mi habilidad de solucionar problemas, el proponer ideas o soluciones e implementarlas.

Me vi en la oportunidad de aprender nuevas tecnologías utilizadas en el trabajo, incluso tener la oportunidad de ponerme al día y llegar a programar dispositivos móviles, lo cual considero fundamental en mi formación como profesionista, mantenerme constantemente actualizado respecto a las nuevas tecnologías que van surgiendo y como estas impactan en la sociedad actual.

No todo fue conocimientos técnicos, gracias al trabajo, tuve la oportunidad de relacionarme con clientes y usuarios, aprender las reglas de negocio de cada proyecto o de la empresa en general, lo cual es una parte fundamental de un ingeniero en computación, que a mi consideración, transforma lo que se realiza en la vida real y lo plasma en un sistema capaz de satisfacer las necesidades del negocio, esa experiencia de abstracción es invaluable. Otra de las aptitudes en constante fortalecimiento a partir de la experiencia laboral, es el trabajo en equipo, desarrollar bajo una línea de mando y resaltar el liderazgo se puede llegar a tener y que la Facultad de Ingeniería me ha inculcado a lo largo de la carrera.

En la recta final de mi estadía en CPC, comencé a realizar mi servicio social en Banco de México, dado que contaba con bastante experiencia programando (aproximadamente 1 año y medio), me dieron la oportunidad de una vacante para trabajar en el Banco, considero que la oportunidad me fue ofrecida por la combinación de mis conocimientos adquiridos por parte de la facultad de ingeniería y la experiencia absorbida durante mi periodo como empleado en CPC, lo cual es un ejemplo de la importancia de fortalecer y crecer profesionalmente a partir del trabajo.

Apartado 1.

GUÍAS

A 1.1. Instalación del Plugin ADT en Eclipse.

I. Iniciamos nuestro IDE Eclipse.

II. Nos dirigimos a nuestra barra superior hasta la opción HELP.

III. Elegimos la opción “Instalar nuevo Software”.

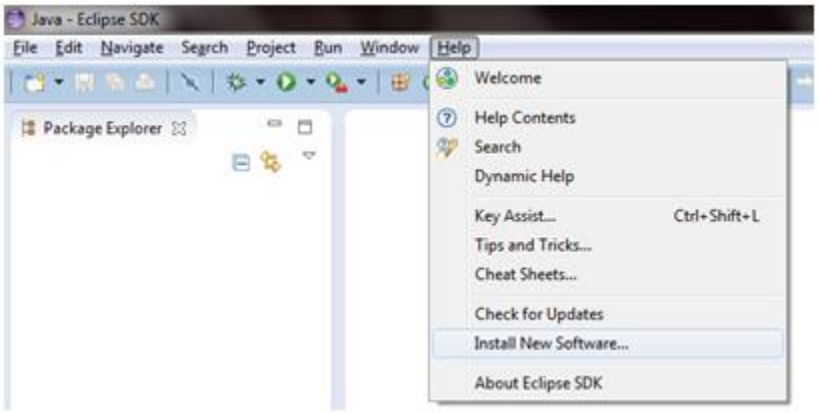
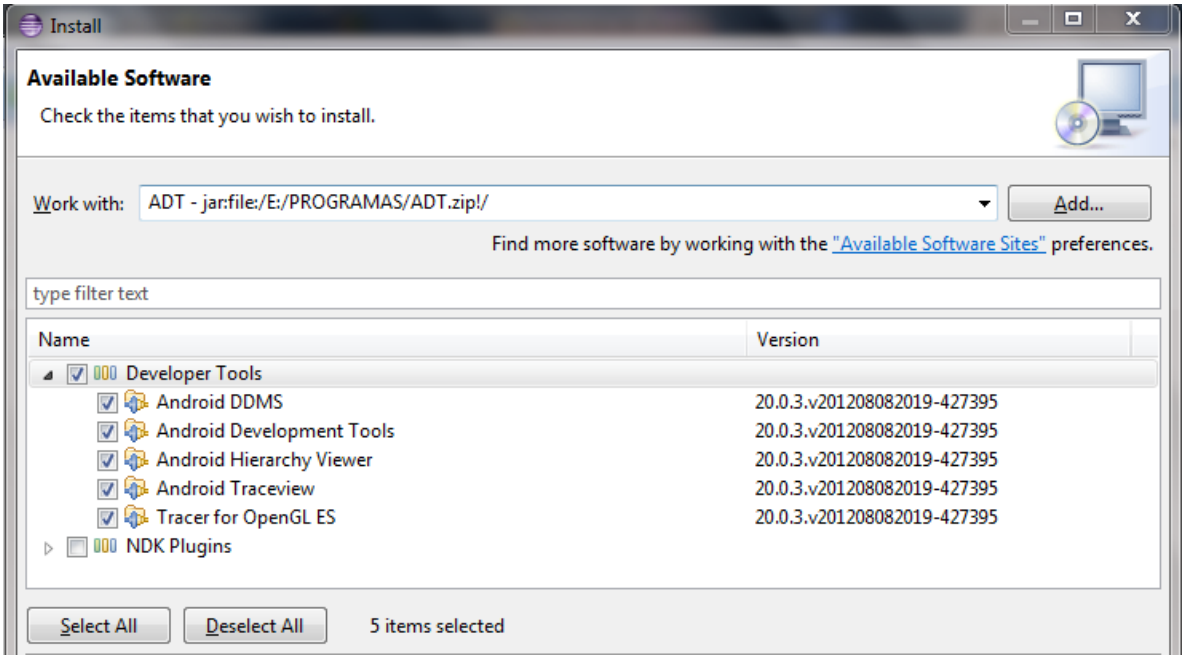


Ilustración 1 Imagen obtenida del sitio http://es.wikihow.com/instalar-Eclipse-y-configurar-ADT

IV. Se mostrara una pantalla con los Softwares disponibles para descargar, marcamos la opción “Developer Tools”



V. Una vez marcados, damos click en el botón ADD, comenzara a instalarse el plugin, damos click en siguiente → siguiente → Aceptar Licencias → finish.

VI. Reiniciamos Eclipse.

VII. Para confirmar que efectivamente, el plugin fue instalado, al reiniciar Eclipse deberán aparecer dos iconos nuevos.

A 1.2. Instalación del Driver de Motorola para utilizar el dispositivo en modo Debug.

I. Ingresamos a la página https://developer.android.com/studio/run/oem-usb.html?hl=es-419#InstallingDriver.

II. Se encuentra una lista de marcas de Dispositivos móviles entre otros, para efecto de esta guía elegiremos el de Motorola.

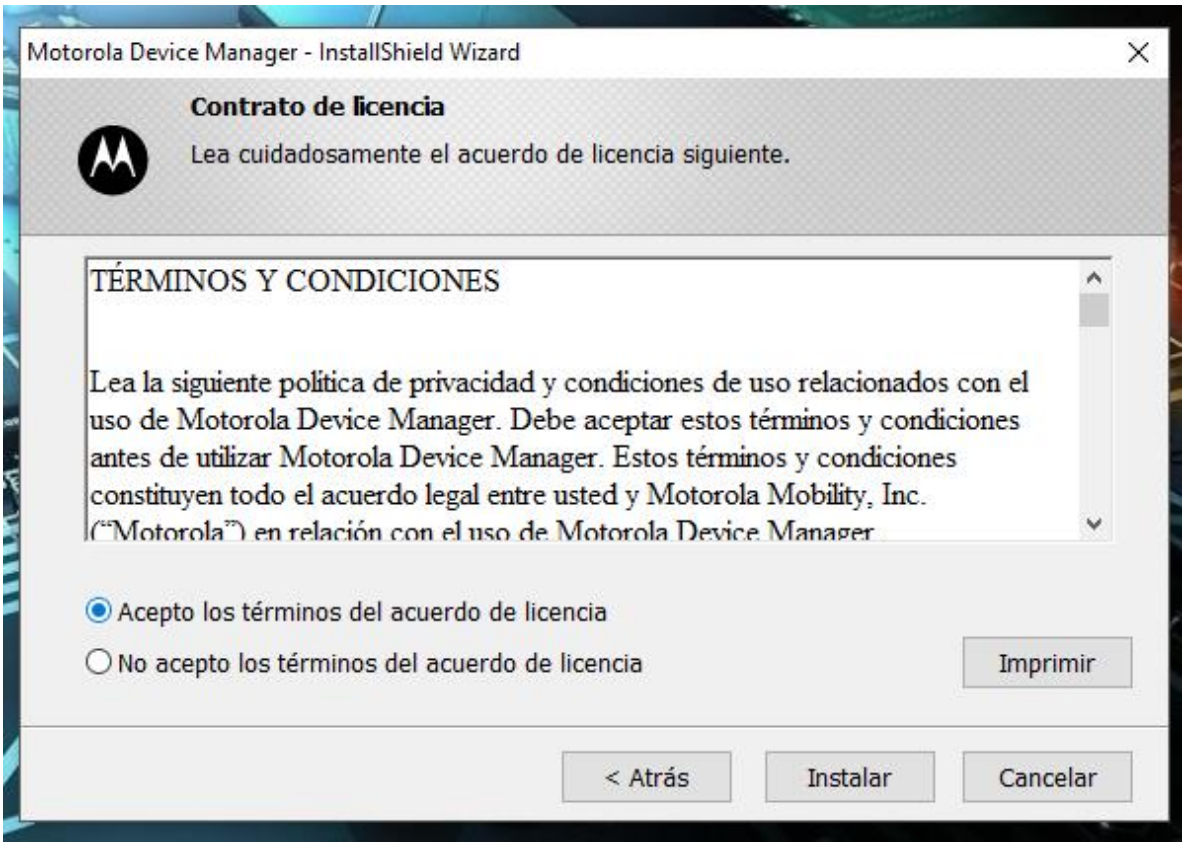
Dell	http://support.dell.com/support/downloads/index.aspx?c=us&cs=19&l=en&s=dhs&~ck=anavml
Fujitsu	http://www.fmworld.net/product/phone/sp/android/develop/
Hisense	http://app.hismarttv.com/dss/resourcecontent.do?method=viewResourceDetail&resourceId=16&type=5
HTC	http://www.htc.com Haz clic en la pestaña de soporte para seleccionar tus productos y tu dispositivo. Los vínculos variarán según la región.
Huawei	http://consumer.huawei.com/en/support/index.htm
Intel	http://www.intel.com/software/android
Kyocera	http://www.kyocera-wireless.com/support/phone_drivers.htm
Lenovo	http://support.lenovo.com/us/en/GlobalProductSelector
LGE	http://www.lg.com/us/support/software-firmware
Motorola	https://motorola-global-portal.custhelp.com/app/answers/detail/a_id/88481/
MTK	http://online.mediatek.com/Public%20Documents/MTK_Android_USB_Driver.zip (descarga de archivo ZIP)
Oppo	http://www.oppo.com/index.php?q=software/view&sw_id=631
Pegatron	http://www.pegatroncorp.com/download/New_Duke_PC_Driver_0705.zip (ZIP download)

III. Nos preguntará el sistema operativo para el cual se instalara el controlador, en esta guía se instalara sobre Windows.

System Requirements:

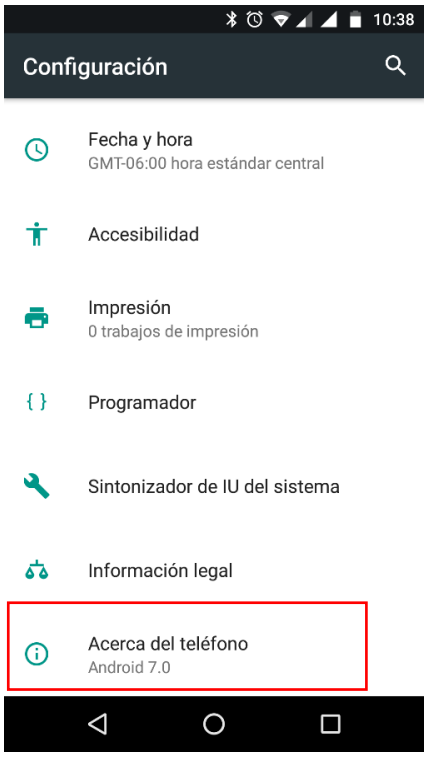
Windows®	Mac OS X®
Windows XP®(SP3 or greater) Windows Vista® Windows 7® Windows 8® Windows 10®	Mac OS® 10.5.8 Leopard Mac OS® 10.6 Snow Leopard Mac OS® 10.7 Lion Mac OS® 10.8 Mountain Lion Mac OS® 10.10 Yosemite
Works with devices running Android®,Motorola OS, or Windows Mobile® operating systems	Works with devices running Android®
Download for Windows	Download for Mac

IV. Una vez descargado, procederemos a instalar el controlador, damos Siguiente → Aceptar términos de licencia → Comienza a instalarse →Cerrar.

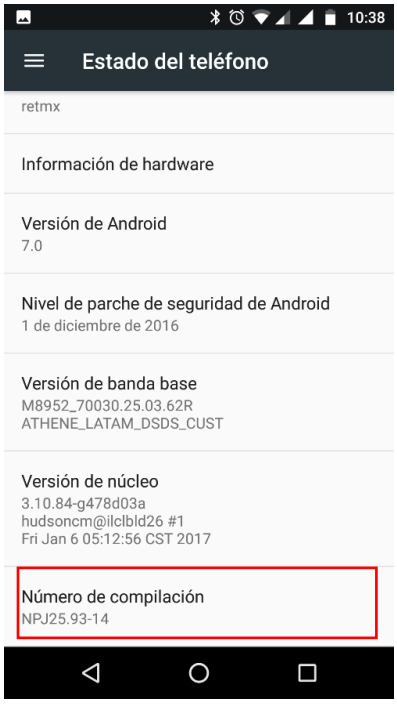

Motorola Device Manager - InstallShield Wizard
Contrato de licencia
Lea cuidadosamente el acuerdo de licencia siguiente.
TÉRMINOS Y CONDICIONES
Lea la siguiente política de privacidad y condiciones de uso relacionados con el uso de Motorola Device Manager. Debe aceptar estos términos y condiciones antes de utilizar Motorola Device Manager. Estos términos y condiciones constituyen todo el acuerdo legal entre usted y Motorola Mobility, Inc. ("Motorola") en relación con el uso de Motorola Device Manager
Acepto los términos del acuerdo de licencia
No acepto los términos del acuerdo de licencia
Imprimir
< Atrás
Instalar
Cancelar


Motorola Device Manager - InstallShield Wizard
Motorola Device Manager
Motorola Device Manager está configurando la instalación del software nuevo.
Paso 1 de 3: Extrayendo archivos...
< Atrás
Siguiente >
Cancelar

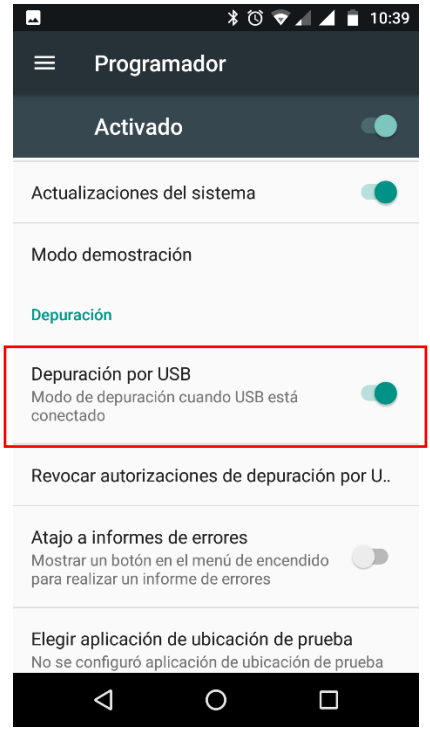
V. En nuestro dispositivo Móvil. En configuración, entramos en el apartado “Acerca del teléfono”.



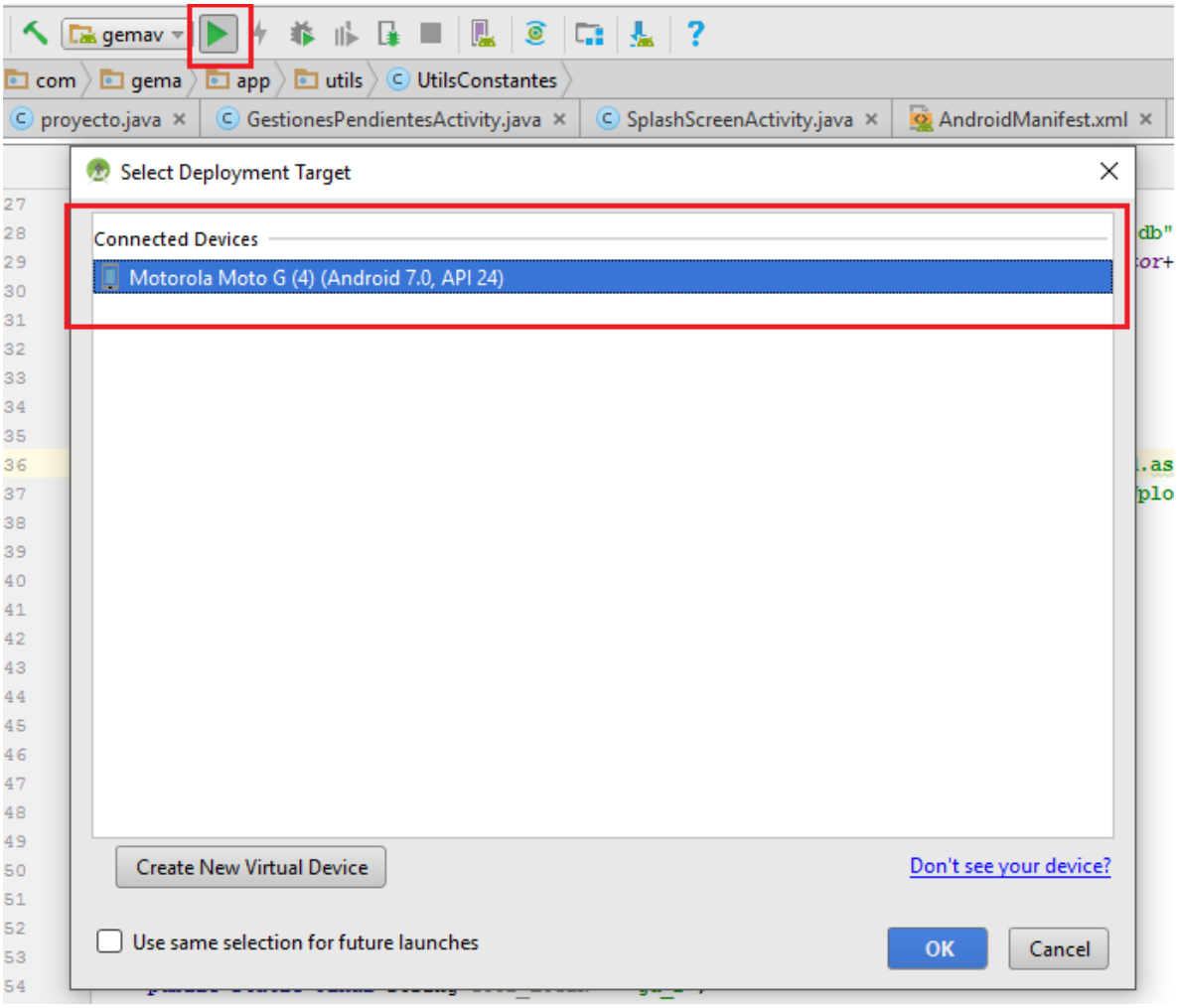
VI. Dentro buscamos la opción Número de Compilación y lo tocamos 7 veces, hasta que nos avise que está activado el modo programador.



VII. Ahora podemos entrar al apartado programador, una vez dentro habilitamos la opción “Depuración por USB".



VIII. Finalmente, comprobamos que todo funcione correctamente, dentro de nuestro IDE Android Studio, presionamos el botón RUN, y nos aparecerá una pantalla en la cual podemos seleccionar nuestro dispositivo conectado.



Apartado 2.

Glosario de términos.

A 2.1. Glosario.

Activity: En Android, es una ventana en la cual se puede colocar la interfaz de usuario, pueden ser de pantalla completa, incrustadas dentro de otra pantalla o flotantes.

ANT: Herramienta de construcción de proyectos, esta nativamente integrada en el IDE Eclipse. Es un software para procesos de automatización de compilación, desarrollado en lenguaje Java.

ArrayList: Clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño.

Asp.net: Modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#.

Backend: Parte que procesa la entrada de datos que se efectuó desde el front-end es decir, son los procesos que utiliza el sistema para resolver las peticiones de los usuarios. El front-end y el back-end interactúan en un sistema web o software para resolver las necesidades de los usuarios.

CRUD: Acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete).

C#: Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

DataSet: Colección de datos generalmente tabulados, se utiliza como un tipo de dato objeto en lenguaje C#, en general y en su versión más simple, un conjunto de datos corresponde a los contenidos de una única tabla de base de datos o una única matriz de datos estadística, donde cada

columna de la tabla representa una variable en particular, y cada fila representa a un miembro determinado del conjunto de datos en cuestión.

Debug: Aplicación o herramienta que permite la ejecución controlada de un programa o un código, para seguir cada instrucción ejecutada y localizar así errores (proceso de depuración), códigos de protección, etc.

Dev Express: Es una de las más completas suites de componentes de interfaz de usuario para el desarrollo en todas las plataformas de .NET como Windows Forms, ASP.NET, MVC, Silverlight y Windows 8 XAML, etc.

Eclipse: Entorno de Desarrollo Integrado perteneciente a Eclipse Foundation, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Eclipse

Frontend: Todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que se ejecutan del lado del navegador web, generalizándose más que nada en tres lenguajes, HTML , CSS Y JavaScript

Gradle: Herramienta de automatización de la construcción de nuestro código. Dispone de una gran flexibilidad que permite trabajar con ella utilizando otros lenguajes y no solo Java a diferencia de ANT. Dispone por otro lado de un sistema de gestión de dependencias sólido.

Grid: Del inglés Cuadricula, es un objeto utilizado para dibujar tablas en sistemas web.

Handler: Es un controlador que permite enviar y procesar mensajes y objetos ejecutables de un subproceso. Cada instancia de Handler se asocia con un solo subproceso y la cola de mensajes del subproceso. A partir de ese momento, entregará mensajes y ejecutables a esa cola de mensajes y los ejecutará a medida que salgan de él.

Handsontable: Es un componente de hoja de cálculo compuesto para aplicaciones y sitios web escritos en JavaScript y HTML.

Imei: Del inglés (International Mobile Station Equipment Identity, identidad internacional de equipo móvil) es un código USSD pregrabado en los teléfonos móviles GSM. Este código identifica al aparato de forma exclusiva a nivel mundial, y es transmitido por el aparato a la red al conectarse a esta.

JavsScript: Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

JQgrid: Herramienta de JQuery utilizado para la creación de Grids, utilizado para sitios web en JavaScript.

JQuery: Biblioteca multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

JSON: Es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript, se considera un formato de lenguaje independiente.

Linq: Conjunto herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: objetos, xmls, bases de datos, etc. Para ello, usa un tipo de funciones propias, que unifica las operaciones más comunes en todos los entornos, con esto, se consigue un mismo lenguaje para todo tipo de tareas con datos.

Listview: Es un grupo de vistas que muestra una lista de elementos desplazables. Los elementos de la lista se insertan automáticamente en la lista con un Adapter que toma contenido de una fuente, como una matriz o consulta de base de datos, y convierte cada resultado en una vista que se dispone en la lista.

Request: En un sistemas web, un request es la petición que se le hace al servidor, esta puede contener en su descripción parámetros y una descripción del tipo de dato que se esta enviando.

Response: En un sistemas web, un response es el resultado de un request, este va en dirección contraria, es decir, del servidor al cliente y contiene lo necesario para satisfacer la petición.

SQL: Es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.

SQL SERVER: Sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.

Stored Procedure o SP: Procedimiento almacenado físicamente en una base de datos, el cual puede ser ejecutado tantas veces como se requiera y está elaborado para realizar una tarea específica.

WebMethod: Técnica que nos permite llamar desde javascript a un método de servidor implementado en c#, vb o cualquier otro lenguaje .NET.

WebService: Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

REFERENCIAS

Barry & Associates, Inc. (2015). *http://www.service-architecture.com/*. Obtenido de http://www.service-architecture.com/articles/web-services/web_services_explained.html.

bluemessaging. (1 de 03 de 2015). *https://bluemessaging.com/*. Obtenido de https://bluemessaging.com/empresa.php.

campusMVP. (25 de 08 de 2015). *https://www.campusmvp.es/*. Obtenido de https://www.campusmvp.es/recursos/post/Desarrollador-web-Front-end-back-end-y-full-stack-Quien-es-quien.aspx.

CPC. (15 de 03 de 2014). *www.cpc.com.mx*. Obtenido de www.cpc.com.mx/sistemas/manual/manual-sab.

CPC. (12 de 06 de 2015). *www.cpc.com.mx*. Obtenido de www.cpc.com.mx/sistemas/manual/manual-siac.

CPC. (12 de 05 de 2016). *www.cpc.com.mx*. Obtenido de www.cpc.com.mx/sistemas/manual/manual-gema.

Eclipse. (10 de 10 de 2010). *https://eclipse.org*. Obtenido de https://help.eclipse.org/neon/index.jsp.

Google. (12 de 02 de 2010). *Android Developers*. Obtenido de https://developer.android.com/reference/android/app/Activity.html

Handsoncode. (18 de 07 de 2013). *https://handsontable.com/*. Obtenido de https://handsontable.com/examples.html?headers.

Invarato, R. (29 de 02 de 2015). *https://jarroba.com*. Obtenido de https://jarroba.com/cliente-servidor-peticion-del-cliente/.

Jquery. (6 de 03 de 2012). *https://jquery.com/*. Obtenido de http://api.jquery.com/jquery.getjson/.

Microsoft. (1 de 11 de 2007). *https://msdn.microsoft.com/*. Obtenido de https://msdn.microsoft.com/es-mx/library/byxd99hx(v=vs.90).aspx.

Oracle. (1 de 01 de 1993-2016). *docs.oracle.com*. Obtenido de https://docs.oracle.com/javase/8/docs/api/java/util/Base64.Encoder.html.

SQLite. (09 de 05 de 2000). *https://www.sqlite.org/*. Obtenido de https://www.sqlite.org/about.html.

w3schools. (13 de 05 de 2012). *https://www.w3schools.com/*. Obtenido de https://www.w3schools.com/js/js_json_intro.asp.