EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Pabellón de Arteaga
Departamento de Ciencias Económico Administrativas

# REPORTE FINAL PARA ACREDITAR RESIDENCIA PROFESIONAL DE LA CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

## [Softv-App]


**Sistemas Administrativos De Tv Restringida**

ING. Miguel Vázquez Martín del Campo
M.A.T.I. Jorge Norberto Mondragón Reyes

Fecha (Diciembre, 2019)

***CAPÍTULO 1: PRELIMINARES***

***2. Agradecimientos.***

La trayectoria que se ha recorrido hasta este momento ha requerido de un gran esfuerzo y muchas horas de empeño. Después de varios años académicos es indudable la avanzada madurez que se ha desarrollado en las aulas de clases y en la vida personal. Primeramente es importante agradecer a todos aquellos maestros que formaron parte de mi formación académica dentro del instituto a todos sus consejos y enseñanzas que nos brindaron, su apoyo fue totalmente fundamental para la creación de este proyecto. A todos mis compañeros de clase que me compartieron de su conocimiento para poder cumplir cada una de mis metas, a mis compañeros de residencia que fueron fundamentales para poder cumplir el proyecto que la empresa solicitaba. A todas las personas que estuvieron tras este gran proyecto, sin dejar atrás al pilar que siempre me mantuvo de frente mi familia que gracias a sus consejos y su apoyo para continuar con mis estudios a nivel superior.

***3. Resumen.***

El presente documento contiene un proyecto importante para la empresa SOFTV la cual como principal objetivo es modificar la forma en que un instalador de telecomunicaciones hace su trabajo atreves de un sistema móvil nativo en Android, analizando cada una delas etapas que estos manejan en el sistema ERP SOFTV-WEB y así poder moldear la aplicación para que sea parte del sistema que actualmente se desarrolla.

***4. Índice.***

# Índice

***Lista de Figuras***



***Lista de tablas***

## *CAPÍTULO 2: GENERALIDADES DEL PROYECTO*

### ***5.- Introducción***

La tecnología móvil ha evolucionado en los últimos años, lo que ha llevado a un crecimiento en el mercado de dispositivos móviles personales a un costo menor del que tenían hace algún tiempo, permitiendo que llegue a más personas en donde los usos que les dan van desde el recreativo, medio de comunicación o cómputo empresarial. Por otro lado, las necesidades y los retos en la empresa “SOFTV” actualmente generan la necesidad de diseñar más y mejores estrategias, utilizando todos los recursos disponibles, ya sean humanos o tecnológicos. De esa manera, muchas estrategias han adoptado modelos que hacen uso de las tecnologías móviles para reforzar el proceso de generación de mejor rendimiento en sus sistemas. El uso de aplicaciones móviles en éste ámbito, les mostraremos desde un punto de vista general, las oportunidades propias del entorno. El desarrollo de una aplicación móvil para técnicos de empresas de telecomunicaciones o tv restringida, es una de los objetivos que SOFTV desea emplear a sus sistemas ERP, para hacer uno de los sistemas más completos e eficientes del país y latino américa. Debido a esta situación se dio a la tarea de crear una aplicación funcional, amigable e intuitiva para el usuario, se hace imprescindible aprender y adaptar los métodos de su sistema WEB a la APP MOVIL. En este reporte explicaremos las situaciones, los métodos y las estrategias oportunas para implementar las soluciones móviles, así como para conseguir el mejor rendimiento de las capacidades de los técnicos y el aprovechamiento de esta nueva tecnología.

***6. Descripción de la empresa u organización y del puesto o área del trabajo del residente***

SOFTV es una empresa que como principales actividades son la creación de sistemas Web y de escritorio ERP para empresas de tele-cable, en otras palabras el sistema que actualmente desarrollan son sistemas que se hacen cargo de distintas operaciones internas de una empresa, desde producción a distribución e incluso recursos humanos.

***Área de trabajo:***

El área que actualmente se desempeña en la empresa es el desarrollo de aplicaciones móviles

***Misión***

Maximizar el valor del portafolio de aplicaciones de nuestros clientes.

***Visión***

Trascender como proveedor global líder en Soluciones de TI y Procesos de Negocio, generando relaciones mutuamente benéficas, de largo plazo y cimentadas en una base de confianza ganada. Construiremos nuestro futuro siendo una empresa sólida y socialmente responsable, con un historial rentable. Proporcionamos servicios innovadores y de alta calidad, impulsados por la pasión de nuestra cultura centrada en el elemento humano.

***7. Problemas a resolver, priorizándolos.***

- Como primer problema fue que la empresa nunca había desarrollado ningún tipo de sistema móvil Android, ni el manejo de la plataforma Android Studio
- Otro fuel que no se tenía ninguna interfaz o sistema para el técnico por el cual se llevó la tarea de desarrollar una.
- El sistema debía ser un sistema limpio y fácil de usar para el usuario, por el cual el sistema sufrió grandes modificaciones para llegar a este objetivo.
- No se savia que métodos o que cosas usar para el consumo de datos mediante Request y formato JSON.
- SOFTV no tenía conocimientos en Android ni responsabilidad de pantallas en Android.

## *8. Justificación*

Actualmente todos los trabajos de instalación de telefonía, televisión e internet se realizan por un técnico seleccionado en el domicilio para instalar dichos servicios, estas órdenes se llevan a cabo por mesa de control haciendo uso del sistema SOFTV WEB.
Las ordenes agendadas al técnico son en base a una bitácora en papel y el técnico en base a sus trabajos realiza las actividades según la dirección de domicilio.
Con este proyecto se pretende crear una aplicación móvil en plataforma Android para que el técnico día a día y en tiempo real sepa que actividades a realizar y ejecutarlas desde la misma app sin necesidad de informar a mesa de control. La app registrara la hora de ejecución así como el lugar sin necesidad de llenar ningún documento, será intuitiva y fácil de manejar para tener un proceso exacto y sin manipulaciones por parte del técnico.

La aplicación móvil le dará a conocer las órdenes y reportes de diversos clientes de telefonía, internet o televisión que tenga contratada la empresa.
Dentro de la mima app se podrán ejecutar varios trabajaos como una instalación de aparato que esta se realiza después de haber contratado un servicio de internet, telefonía o televisión según sea el caso y es la principal prioridad de la aplicación, se podrán hacer retiros de aparato para cuando el cliente cancele el servicio que tiene contratado, cambios de aparato por si tiene algún aparato dañado así como cambios de domicilio para cuando el cliente decida cambiar su estancia a otra parte.

***9. Objetivos (General y Específicos)***

***Objetivos generales:***

Como principal objetivo se desea cambiar la manera actual en que los técnicos de las empresas de telecomunicaciones realizan sus trabajos , ya que actualmente se le agenda el usuario al técnico para dar atención o mantenimiento, posteriormente crean órdenes y reportes de servicios que se deben de imprimir y entregar al técnico agendado las hojas con sus respectivos trabajos a realizar ya sean órdenes de instalación de aparatos o cambio de aparatos, cambio de domicilio, instalación de extensiones, retiro de aparatos.

Además de lograr tener un mejor registro de información para el sistema web que se maneja en las compañías ya que con la aplicación móvil para los técnicos se lograra tener un informe en tiempo real de los trabajos realizado, porque en el formato actual en que se manejan dichas empresas se tiene que recibir en mesa de control del sistema un compendio de hojas que llena el técnico al término de su labor (ya se ordenes o quejas) y toda esa información es vaciada en el sistema web, pero es imposible pasar todos los datos en un momento, lo que lleva a la persona encarga de ello tener que esperar horas o días para terminar con todas las hojas que llegan de parte de los técnicos, lo que provoca que el sistema esté registrado datos manipulados o en su defecto que un trabajo pueda estar en estatus de “Pendiente” pero en realidad ya esté en estatus de “Ejecutada” .

Tener informado al técnico sobre sus nuevas órdenes y reportes agendados al momento es una de las principales prioridades al momento de crear la aplicación móvil. En este momento es donde la tecnología móvil nos permite estar conectados todo el tiempo y teniendo en cuenta que el celular se ha vuelto indispensable para el ser humano lo ideal es explotar esta tecnología haciendo que el técnico pueda recibir en su celular notificaciones sobre nuevos trabajos que le hayan sido agendados, con datos sobre el trabajo a realizar, dirección y nombre del cliente y así poder optimizar el tiempo de ellos al saber que tiene una nueva orden que atender.

***Objetivos específicos:***

- Implementar una interfaz fácil de manipular y aceptación de usabilidad para el usuario haciendo uso de pantallas fragmentadas.
- Geo localizar al usuario para obtener la ubicación exacta de los clientes visitados.
- Crear pantalla principal con fácil usabilidad para el uso de Reportes y Órdenes
- Creación del menú reportes, así como la ejecución de este con formatos API RES y JSON para el mejor manejo de información

## *CAPÍTULO 3: MARCO TEÓRICO*

### *10. Marco Teórico (fundamentos teóricos).*

### ***Telecomunicaciones (definición)***

Las telecomunicaciones son ya una constante en la vida de las personas y hoy no es posible concebir el mundo sin ellas. Pero, ¿qué son las telecomunicaciones? Se trata de un conjunto de técnicas que permiten la comunicación a distancia, lo que puede referirse a la habitación de al lado o a una nave espacial situada a millones de kilómetros de distancia.

Las telecomunicaciones son una conexión entre usuarios a corta o larga distancia, de uso diario de intercambio de información de un sitio a otro en un tiempo realmente corto teniendo como medio de transmisión señales eléctricas que consisten en múltiples estaciones de receptores y transmisores integrados. Que intercambian información. Siendo la red más amplia y conocida el internet. *Historia de* las telecomunicaciones

Las telecomunicaciones, como lo conocemos hoy en día, tuvieron su primer punto de cambio en el año 1800 cuando Alessandro Volta inventa la celda eléctrica o pila eléctrica. El siguiente gran avance en esta materia fue el telégrafo electromagnético desarrollado por Samuel Morse en 1835 y enseguida la expansión del teléfono en 1876 cuando Alexander Graham Bell obtiene su patente en Estados Unidos. Desde ahí, el avance en las telecomunicaciones tuvo un crecimiento desenfrenado.

### ***Internet (definición)***

El Internet, es un sistema mundial de redes de computadoras, un conjunto integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede, en caso de contar con los permisos apropiados, acezar información de otra computadora y poder tener inclusive comunicación directa con otros usuarios en otras computadoras.

### ***Internet en México (historia)***

La historia de Internet en México propiamente da inicio en 1989, cuando el Instituto Tecnológico y de Estudios Superiores de Monterrey estableció el primer enlace dedicado a la red de la National Science Foundation (NSF), a través de la Escuela de Medicina de la Universidad de Texas, en la ciudad de San Antonio (UTSA), utilizando los protocolos de conexión propios de Internet. Entonces se conecta el primer equipo a Internet bajo el dominio .mx: dns.mty.itesm.mx con la dirección 131.178.1.1 El 28 febrero de 1989, la NSFnet reconoció oficialmente la conexión de México.

Poco después fue establecido un segundo nodo entre el Instituto de Astronomía de la Universidad Nacional Autónoma de México, y el Centro Nacional de Investigación Atmosférica (NCAR) de Boulder, Colorado, en Estados Unidos. Posteriormente el ITESM Campus Estado de México se conectó a la red de la NSF a través del NCAR, por medio de un enlace digital vía satélite.

De acuerdo con un estudio realizado por NIC México, en 1991 los servicios que con mayor frecuencia utilizaban los académicos e investigadores eran los siguientes:

- Acceso remoto (Telnet)
- Transferencia de Archivos (FTP)
- Correo Electrónico (E-mail)

***Demanda de internet en México***

En México hay 74.3 millones de usuarios de Internet de seis años o más, que representan el 65.8% de la población en ese rango de edad. El 51.5% de los internautas son mujeres y 48.5% son hombres. Se observa un crecimiento de 4.2 puntos porcentuales respecto a lo reportado en 2017, cuando se registraron 71.3 millones de usuarios. Del total de la población usuaria de internet de seis años o más, el grupo de entre 25 y 34 años es el que registra la mayor proporción de usuarios de internet, las mujeres en este rango de edad representan 10.4% y los hombres 9.8%. Por otro lado, la población de 55 años o más es la que menos usa internet, registrando cifras del 4.1% para las mujeres y 4.0% para los hombres. Las tres principales actividades de los usuarios de Internet en 2018 fueron: entretenimiento (90.5%), comunicación (90.3%) y obtención de información (86.9 por ciento).

***Uso de internet en México.***

En México hay 74.3 millones de usuarios de Internet de seis años o más, que representan el 65.8% de la población en ese rango de edad. El 51.5% de los internautas son mujeres y 48.5% son hombres. Se observa un crecimiento de 4.2 puntos porcentuales respecto a lo reportado en 2017, cuando se registraron 71.3 millones de usuarios. Del total de la población usuaria de internet de seis años o más, el grupo de entre 25 y 34 años es el que registra la mayor proporción de usuarios de internet, las mujeres en este rango de edad representan 10.4% y los hombres 9.8%. Por otro lado, la población de 55 años o más es la que menos usa internet, registrando cifras del 4.1% para las mujeres y 4.0% para los hombres. Las tres principales actividades de los usuarios de Internet en 2018 fueron: entretenimiento (90.5%), comunicación (90.3%) y obtención de información (86.9 por ciento).

***Telefonía***

Durante el año pasado, 73.5% de la población de seis años o más utilizó el teléfono celular. De éstos, ocho de cada diez usuarios, contaban con un celular inteligente (Smartphone), que les permitía conectarse a Internet. El número total de usuarios que disponen de celular inteligente (Smartphone) creció de 64.7 millones de personas en 2017 a 69.6 millones en 2018. Además, en 2018 hay un aumento de los usuarios que se conectan a internet desde un celular inteligente (Smartphone), pasando del 92.0% en 2017 a 93.4% en 2018; con una diferencia de 5.5 millones de personas. La conexión móvil a internet (conexión de datos) es la más utilizada por el 89.0% de los usuarios, mientras que el restante 11.0% se conecta a internet desde un celular inteligente (Smartphone) mediante WiFi. De los usuarios de celular inteligente (Smartphone), 45.5 millones instalaron aplicaciones en sus teléfonos: 89.5% de mensajería instantánea, 81.2% herramientas para acceso a redes sociales, 71.9% aplicaciones de contenidos de audio y video, y 18.1% alguna aplicación para acceder a banca móvil.

***Computadora***

Los usuarios de computadora de seis años o más alcanzaron en 2018 los 50.8 millones, equivalentes al 45.0% del total de la población en este rango de edad. Esta estimación es menor en 0.3 puntos porcentuales respecto del registrado en 2017, cuando fue de 45.3 por ciento. La proporción estimada de hogares que disponen de

una computadora registró un descenso de 0.5 puntos porcentuales, al pasar de 45.4% en 2017 a 44.9% en 2018.

### ***Aplicaciones móviles***

Una aplicación móvil la podemos definir como cualquier programa informático que ejecuta tu teléfono móvil para realizar una tarea, mostrar medios de información, facilitar la comunicación, entretener o brindar un servicio.

### ***Historia***

A inicios de los noventas, cuando el uso de teléfonos móviles se disparó, pudimos encontrar en ellos funciones como calculadoras, juegos arcade, SMS o ringtones, esas sencillas características que el día de hoy pasan desapercibidas en nuestros móviles fueron el cimento para el complejo mundo de aplicaciones del que hoy un 61.75% de la población mundial es usuario.

Muchas personas tienden a pensar que las apps no tienen más de una década de existencia, esto es porque su boom (tal como las conocemos) se dio en 2008 cuando de la mano del primer iPhone emergió un mercado que marcaría la historia y el uso de los móviles. Casi de inmediato conocimos un nuevo sistema operativo que competiría directamente con Apple: Android. Estos dos gigantes cuentan con más de 3M de distintas APLICACIONES MÓVILES EN SUS MERCADOS.

### ***Evolución***

La evolución que está sufriendo el mercado de las apps no sigue una línea continua, ya que el consumidor ha evolucionado de la misma manera que el mercado. Durante los años 2012 y 2013 el número de descargas de aplicaciones era desmedido. El consumidor quería conocer nuevas aplicaciones y usarlas.

En esta línea parece que tendría más sentido diseñar aplicaciones móviles enfocadas de un modo todo lo particular que se pueda e intentando no moverse en la generalidad, ya que es más difícil mantener la fidelidad con el público y su uso de las apps. Todavía existen diferentes ámbitos de mercado por explorar en donde veremos cómo se desarrollan las futuras aplicaciones.

Existe un gran mercado en desarrollo de las apps, casi se podría decir que hay un app para cada ámbito de nuestras vidas. Muchas marcas están aprovechando este nicho de mercado para crecer. Sin embargo, esta estrategia debe estar bien planteada, ya que el usuario de telefonía móvil es más consciente del poder que tiene en sus manos.

En este sentido, un buen servicio hacia el usuario final es una atención fundamental, ya que de esa manera se va construyendo una nueva relación que permitirá a la marca y al cliente establecer un canal de comunicación en el que puedan expresar sus necesidades. Ese servicio que se busca dar es determinante a la hora de que un usuario continúe usando o no la aplicación.

***JSON.***

***JSON (JavaScript Object Notation - Notación de Objetos de JavaScript)*** es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:
Una colección de pares de nombre/valor. En varios lenguajes esto es conocidos como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.
Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:
Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con llave de apertura y termine con llave de cierre. Cada nombre es seguido por: dos puntos y los pares nombre/valor están separados por, coma.

***Fragments.***

Un Fragment representa un comportamiento o una parte de la interfaz de usuario en una FragmentActivity. Puedes combinar varios fragmentos en una sola actividad para crear una IU multi-panel y volver a usar un fragmento en diferentes actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene un ciclo de vida propio, que recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una "sub actividad" que puedes volver a usar en diferentes actividades).

Android introduce los fragmentos en Android 3.0 (nivel de API 11), principalmente para admitir diseños de IU más dinámicos y flexibles en pantallas grandes, como las de las tablets. Como la pantalla de una tablet es mucho más grande que la de un teléfono, hay más espacio para combinar e intercambiar componentes de la IU. Los fragmentos admiten esos diseños sin la necesidad de que administres cambios complejos en la jerarquía de vistas. Al dividir el diseño de una actividad en fragmentos, puedes modificar el aspecto de la actividad durante el tiempo de ejecución y conservar esos cambios en una pila de actividades administrada por la actividad. Ahora están disponibles mediante la biblioteca de compatibilidad de fragmentos.

Para crear un fragmento, debes crear una subclase Fragment (o una subclase existente de ella). La clase Fragment tiene un código que se asemeja bastante a una Activity. Contiene métodos de devolución de llamada similares a los de una actividad, como onCreate(), onStart(), onPause() y onStop(). De hecho, si estás convirtiendo una aplicación de Android existente para utilizar fragmentos, deberías simplemente trasladar código de los métodos de devolución de llamada de tu actividad a los métodos respectivos de tu fragmento.

***MAIN_ACTIVITY.***

Las apps proporcionan varios puntos de entrada.
Las apps para Android se compilan como una combinación de componentes que pueden invocarse de manera individual. Por ejemplo, una actividad es un tipo de componente de app que proporciona una interfaz de usuario (IU).

La actividad "principal" comienza cuando el usuario presiona el ícono de tu app. También puedes dirigir al usuario a una actividad de otro lugar, como una notificación o incluso una app diferente.
Otros componentes, como los receptores de emisión y los servicios, permiten que tu app realice tareas en segundo plano sin una IU.

***RETROFIT.***

Retrofit es un cliente REST para Android y Java, desarrollada por Square, muy simple y fácil de aprender. Permite hacer peticiones GET, POST, PUT, PATCH, DELETE y HEAD, gestionar diferentes tipos de parámetros y parsear automáticamente la respuesta a un POJO.

Supongamos que yo tengo una app Android que se va a conectar a un servidor, lo que normalmente se hace es hacer una petición GET a una API la cual es una URL, lo que hará será hacer una petición al servidor que le devolverá un JSON, entonces esto es muy fácil, lo único que tienes que hacer es importar la librería de Retrofit a tu app de Android, también se usa cualquier biblioteca HTTP para comunicarse con la solicitud y la respuesta el servidor y luego usar varios análisis sintácticos disponibles para analizar su respuesta de nuevo por ejemplo GSON que se encarga de analizar la respuesta del JSON, entonces veamos cómo funciona.

***Geolocalización Android.***

Los dispositivos móviles Android (Smartphone, tablets,…) de determinar su ubicación física en coordenadas GPS, lo que denominamos Geolocalización. Debemos de agregar a nuestra aplicación permisos para usar el GPS los cuales son:
-INTERNET, -ACCES_FINE_LOCATION.

***POJO***

POJO son las iniciales de «Plain Old Java Object», que puede interpretarse como «Un objeto Java Plano Antiguo». Un POJO es una instancia de una clase que no extiende ni implementa nada en especial. Para los programadores Java sirve para enfatizar el uso de clases simples y que no dependen de un framework en especial. Este concepto surge en oposición al modelo planteado por los estándares EJB anteriores al 3.0, en los que los Enterprise JavaBeans (EJB) debían implementar interfaces especiales.

Por ejemplo, un Controller de Spring tiene que extender de SimpleFormController, e implementar los métodos abstractos de ese tipo: por eso, no es un POJO. Un Servlet, tiene que extender de HttpServlet por lo que tampoco es un POJO. En cambio, si defines una clase Cliente con atributos y unas cuantas operaciones, tienes un simple y modesto POJO.

***MVC***

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

## *CAPÍTULO 4: DESARROLLO*

### *11. Procedimiento y descripción de las actividades realizadas.*

***Diseño de bar_activity y fragments***

Para que el usuario pueda usar de forma fácil e intuitiva la aplicación móvil se hizo un contenedor de pantallas llamado MainActivity en el cual se reúnen todas las pantallas fragmentadas o fragments de Android Studio.

En el cual se le da uso a los implementos de Android para obtener un efecto swipe o de desliz entre los diversos fragmentos el implemento es llamado como ViewPager.OnPageChangeListener (Ilustración 1).

```
public class MainActivity extends AppCompatActivity implements ActionBar.TabListener, ViewPager.OnPageChangeListener {
    private ViewPager mViewPager;
    ScrollView hzScrollView;
    Button info;
    int position;
    RelativeLayout layoutAnimado;
    public static TextView NombreTec, Contrato, Status, Nombre, Direccion, InfoServicios;

    Request request = new Request();

    @Override
    protected void onCreate(Bundle onSaveInstanceState) {
        super.onCreate(onSaveInstanceState);
        setRetainInstance(true);
        validar_Permiso();
        setContentView(R.layout.activity_swipe);
        info= findViewById(R.id.info);
        layoutAnimado= findViewById(R.id.animado);
        hzScrollView= findViewById(R.id.scv);
        NombreTec= findViewById(R.id.tecniconame);
        Contrato= findViewById(R.id.contrato);
        Status= findViewById(R.id.status);
        Nombre= findViewById(R.id.infonombre);
        Direccion= findViewById(R.id.infodireccion);
        InfoServicios= findViewById(R.id.infoservicios);
        setTitle(null);

       // NombreTec.setText(nombre_tecnico);
        Contrato.setText(request.contraroMA);
        Status.setText(request.statusMA);
        NombreTec.setText(Util.getNombreTecnicoPreference(Util.preferences));
        //* Boton de informacion
        info.setOnClickListener((v) -> {
                request.getInfoCliente(getApplicationContext());
                    request.getServicios(getApplicationContext());
                if(layoutAnimado.getVisibility()==View.GONE) {
                    layoutAnimado.setVisibility(View.VISIBLE);
                    hzScrollView.setVisibility(View.VISIBLE);

                    info.setText("Ocultar");
                }
                else{
                    layoutAnimado.setVisibility(View.GONE);
                    hzScrollView.setVisibility(View.GONE);
                    info.setText("Info");
                }
        });
```

*Ilustración 1 ViewPager.OnPageChengeListener*

También se usó un implemento más para tener acción y ordenamiento en la barra de actividades, donde se concentra el nombre y el fragmento asignado para cada tarea el cual es conocido como ActionBar.TabListener (Ilustración 2 e Ilustración 3)

```
    //* Swipe
    PagerAdapter adapter = new PagerAdapter(getSupportFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.pager);


    mViewPager.setAdapter(adapter);

    mViewPager.setOnPageChangeListener(this);


    ActionBar actionBar = getSupportActionBar();


    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);


    ActionBar.Tab tab = actionBar.newTab().setText("Trabajo").setTabListener(this);
    actionBar.addTab(tab);

    tab = actionBar.newTab().setText("Horas").setTabListener(this);
    actionBar.addTab(tab);

    tab = actionBar.newTab().setText("Material").setTabListener(this);
    actionBar.addTab(tab);

    tab = actionBar.newTab().setText("Finalizar").setTabListener(this);
    actionBar.addTab(tab);

}
```

*Ilustración 2 ActionBar.TabListener (controlador)*

Gracias a estos implementos y al contenedor tenemos un efecto swipe en la barra de tareas de la app. Es importante el ordenamiento de las activitys y sus respectivas respuestas de ordenamiento en el cual hacemos uso de un PageAdapter para la secuencia y ordenamiento de las pantallas, para que el usuario tenga identificado que es lo siguiente o que sección se debe ejecutar después


*Ilustración 3 ActionBar.TabListener (Vista)*

```
    public PagerAdapter(FragmentManager fm) { super(fm); }

    public Fragment getItem(int arg0) {
        switch (arg0) {
            case 0:
                return new Trabajos();
            case 1:
                return new InstalacionFragment();
            case 2:
                return new Materiales();
            case 3:
                return new EjecutarFragment();
            default:
                return null;
        }
    }


        FragmentManager manager = getSupportFragmentManager();



    public int getCount() { return 4; }
}
@Override
public void onPageScrolled(int i, float v, int il) {

}

@Override
public void onPageSelected(int i) { getSupportActionBar().setSelectedNavigationItem(i); }

@Override
public void onPageScrollStateChanged(int i) {

}

@Override
public void onTabSelected(ActionBar.Tab tab, FragmentTransaction fragmentTransaction) {
   // mViewPager.setCurrentItem(tab.getPosition());
    position=tab.getPosition();
    mViewPager.setCurrentItem(position);

}
```

*Ilustración 4 PageAdapter  (Manejador de objetos fragment)*

## ***GEOLOCALIZACION***

Para obtener las coordenadas del usuario en tiempo real se hizo uso de los permisos de localización e internet llamados, ACCES_FINE_LOCATION, INTERNET, ACCES_COARSE_LOCATION

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" tools:node="remove" />
```

*Ilustración 5 Permisos (Geolocalización)*

Al iniciar la aplicación comprobamos que los permisos de localización GPS estén activos des pues de activarlos comprobamos si el dispositivo móvil cuenta con internet, si el dispositivo cuenta con internet el GPS lanzara las coordenadas mediante el administrador de localización NETWORKPROVAIDER, si el dispositivo no cuenta con internet el administrador tomara el GPS interno del dispositivo para lanzar las coordenadas

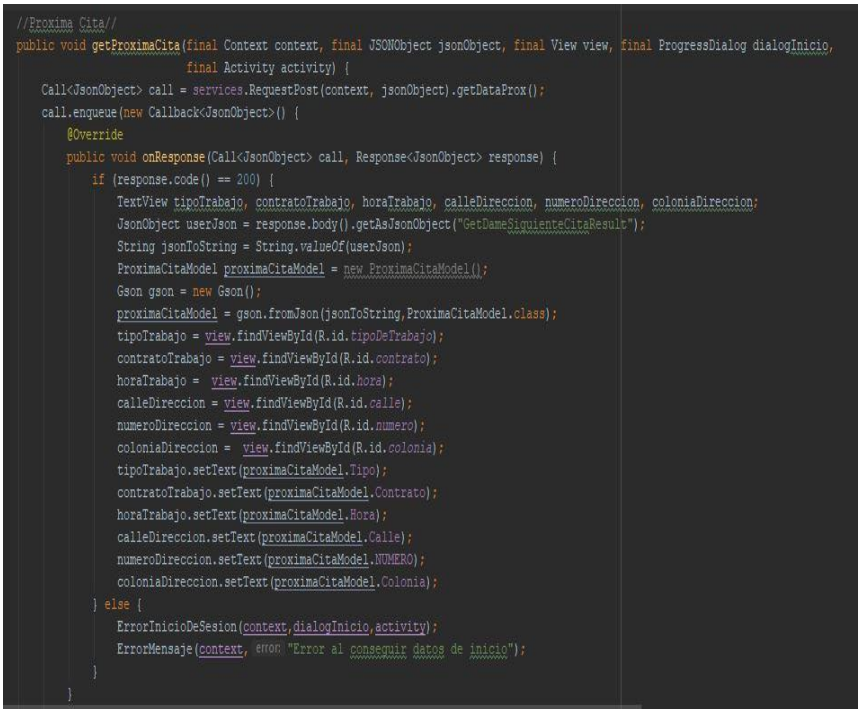
*Ilustración 6 Controlador GPS*


*Ilustración 7 Vista GPS*

***Ejecución y lanzamiento de datos para sección reportes***

Retrofit 2 fue la librería que más se adapta a las necesidades del proyecto debido a que además de tener una forma de realizar peticiones rápidas y seguras al momento de manejar JSONObject y JSONArray era mucho más sencillo que las demás librerías.
Ya con esta herramienta fue con la cual creamos el login y mantuvimos para todas las peticiones y envió de información.

Usamos modelos POJO para poder guardar la información que recibimos como respuesta del servidor en formato JSONArray y JSONObject, pero al momento de manejar JSONObject usamos Gson ya que usar listas sencillos la aplicación se cerraba con los elementos nulos lo que en JSONArrays no pasaba.



```
//Proxima Cita//
public void getProximaCita(final Context context, final JSONObject jsonObject, final View view, final ProgressDialog dialogInicio,
                           final Activity activity) {
    Call<JsonObject> call = services.RequestPost(context, jsonObject).getDataProx();
    call.enqueue(new Callback<JsonObject>() {
        @Override
        public void onResponse(Call<JsonObject> call, Response<JsonObject> response) {
            if (response.code() == 200) {
                TextView tipoTrabajo, contratoTrabajo, horaTrabajo, calleDireccion, numeroDireccion, coloniaDireccion;
                JsonObject userJson = response.body().getAsJsonObject("GetDameSiguienteCitaResult");
                String jsonToString = String.valueOf(userJson);
                ProximaCitaModel proximaCitaModel = new ProximaCitaModel();
                Gson gson = new Gson();
                proximaCitaModel = gson.fromJson(jsonToString,ProximaCitaModel.class);
                tipoTrabajo = view.findViewById(R.id.tipoDeTrabajo);
                contratoTrabajo = view.findViewById(R.id.contrato);
                horaTrabajo =  view.findViewById(R.id.hora);
                calleDireccion = view.findViewById(R.id.calle);
                numeroDireccion = view.findViewById(R.id.numero);
                coloniaDireccion =  view.findViewById(R.id.colonia);
                tipoTrabajo.setText(proximaCitaModel.Tipo);
                contratoTrabajo.setText(proximaCitaModel.Contrato);
                horaTrabajo.setText(proximaCitaModel.Hora);
                calleDireccion.setText(proximaCitaModel.Calle);
                numeroDireccion.setText(proximaCitaModel.NUMERO);
                coloniaDireccion.setText(proximaCitaModel.Colonia);
            } else {
                ErrorInicioDeSesion(context,dialogInicio,activity);
                ErrorMensaje(context, error: "Error al conseguir datos de inicio");
            }
        }
```

*Ilustración 8 Request (Ejemplo de request con Retrofit 2)*

```java
public class ProximaCitaModel {
    @SerializedName("Calle")
    @Expose
    public  String Calle;
    @SerializedName("Clave")
    @Expose
    public int Clave;
    @SerializedName("Colonia")
    @Expose
    public String Colonia;
    @SerializedName("Contrato")
    @Expose
    public String Contrato;
    @SerializedName("Hora")
    @Expose
    public String Hora;
    @SerializedName("NUMERO")
    @Expose
    public  String NUMERO;
    @SerializedName("Tipo")
    @Expose
    public String Tipo;

    public String getCalle() { return Calle; }

    public String getColonia() { return Colonia; }
```

*Ilustración 9 Modelo POJO (Ejemplo)*

Para obtener la lista de quejas que el técnico tiene agendadas es necesario usar un Request haciendo petición de la información con método GET ya que en retrofit es el método utilizado.

```java
public void getQuejas(final Context context) {
    Service service = null;
    try {
        service = services.getOrdSerService(context);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    Call<Example> call = service.getDataOrdenes();

    call.enqueue(new Callback<Example>() {
        @Override
        public void onResponse(Call<Example> call, Response<Example> response) {
            if (response.code() == 200) {
                Example jsonResponse = response.body();
                array.dataque = new ArrayList<List<Queja>>(asList(jsonResponse.getDameOrdenesQuejasTotalesResult.getQueja()));
                Iterator<List<Queja>> itData = array.dataque.iterator();
                while (itData.hasNext()) {
                    List<Queja> dat = (List<Queja>) itData.next();
                    for (int i = 0; i < dat.size(); i++) {
                        if (dat.get(i).getStatus().equals("Ejecutada")) {
                            try {
                                Inicio.RE = dat.get(i).getTotal();

                            } catch (Exception e) {
                                Inicio.RE = 0;
                            }
                        }
                        if (dat.get(i).getStatus().equals("Pendiente")) {
                            try {
                                Inicio.RP = dat.get(i).getTotal();
                            } catch (Exception e) {
                                Inicio.RP = 0;
                            }
                        }
                        if (dat.get(i).getStatus().equals("Visita")) {
                            try {
                                Inicio.RV = dat.get(i).getTotal();
                            } catch (Exception e) {
                                Inicio.RV = 0;
                            }
                        }
```

*Ilustración 10 Ejemplo de Request con método GET)*

Después que tenemos y lanzamos el request esperamos que nos dé un response 200 como respuesta, si este no es así mandamos un toast diciendo error al conseguir la información y se vuelve a ejecutar hasta vaciar la información.

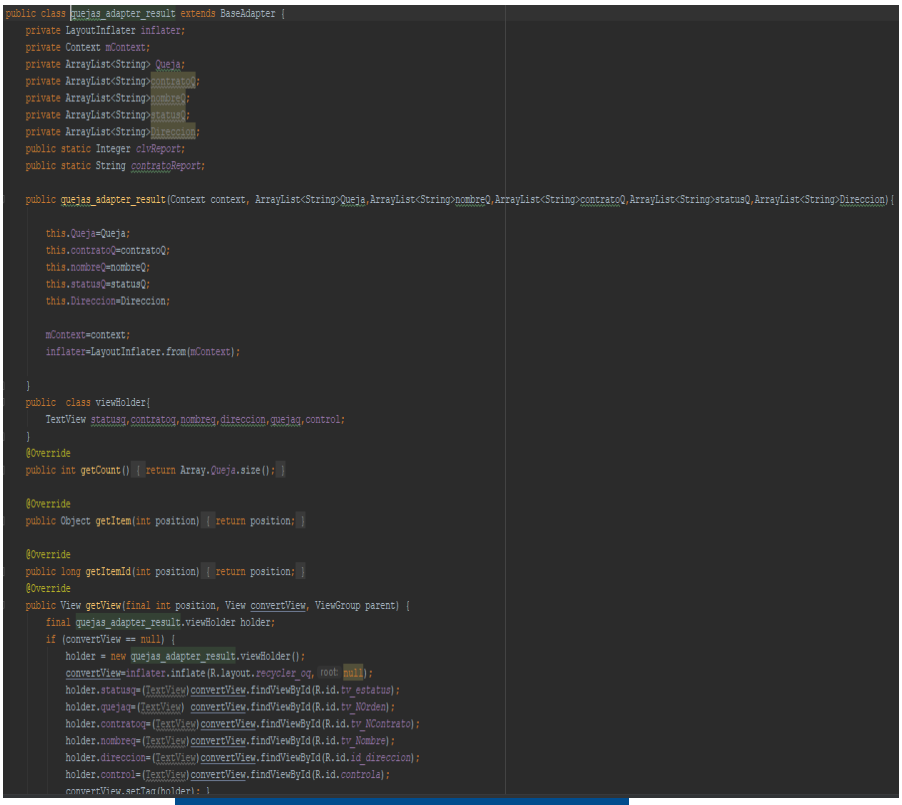
```
public void getListQuejas(final Context context) {
    Service service = null;
    try {
        service = services.getListQuejasService(context);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    Call<QuejasList> call = service.getQuejasAgendadas();
    call.enqueue(new Callback<QuejasList>() {
        @Override
        public void onResponse(Call<QuejasList> call, Response<QuejasList> response) {
            if (response.code() == 200) {
                QuejasList jsonResponse = response.body();
                array.dataquejas = new ArrayList<List<ListadoQuejasAgendadas>>(asList(jsonResponse.GetDameListadoQuejasAgendadasResult()));
                Iterator<List<ListadoQuejasAgendadas>> itData = array.dataquejas.iterator();
                while (itData.hasNext()) {
                    List<ListadoQuejasAgendadas> dat = (List<ListadoQuejasAgendadas>) itData.next();
                    Array.Queja.clear();
                    Array.nombreQ.clear();
                    Array.statusQ.clear();
                    Array.contratoQ.clear();
                    Array.Direccion.clear();
                    for (int i = 0; i < dat.size(); i++) {
                        Array.Queja.add(String.valueOf(dat.get(i).getClvQueja()));
                        Array.contratoQ.add(String.valueOf(dat.get(i).getContrato()));
                        Array.nombreQ.add(String.valueOf(dat.get(i).getNombre()));
                        Array.statusQ.add(String.valueOf(dat.get(i).getStatus()));
                        Array.Direccion.add(String.valueOf(dat.get(i).getCalle() + ", " + dat.get(i).getNUMERO() + ", " + dat.get(i).getColonia()));

                    }
                }
                Intent intent1 = new Intent(context, Reportes.class);
                context.startActivity(intent1);
            } else {
                Toast.makeText(context, text: "Error al conseguir lista quejas", Toast.LENGTH_LONG).show();
            }
        }
```

*Ilustración 11 Obtención de datos (sección quejas)*

Luego que tenemos los datos, vaciamos la información en un list-adapter o array adapter para moldear y dar salida a una vista para e que el técnico manipule la información.

Para implementar array adapter necesitamos un template que sera utilizado como molde para pintar cada una de las filas de nuestra lista en base a la cantidad de elementos que estemos agregando a nuestro adapter.



```
public class quejas_adapter_result extends BaseAdapter {
    private LayoutInflater inflater;
    private Context mContext;
    private ArrayList<String> Queja;
    private ArrayList<String>contratoQ;
    private ArrayList<String>nombreQ;
    private ArrayList<String>statusQ;
    private ArrayList<String>Direccion;
    public static Integer clvReport;
    public static String contratoReport;

    public quejas_adapter_result(Context context, ArrayList<String>Queja,ArrayList<String>nombreQ,ArrayList<String>contratoQ,ArrayList<String>statusQ,ArrayList<String>Direccion){

        this.Queja=Queja;
        this.contratoQ=contratoQ;
        this.nombreQ=nombreQ;
        this.statusQ=statusQ;
        this.Direccion=Direccion;

        mContext=context;
        inflater=LayoutInflater.from(mContext);

    }
    public  class viewHolder{
        TextView statusq,contratoq,nombreq,direccion,quejaq,control;
    }
    @Override
    public int getCount() { return Array.Queja.size(); }

    @Override
    public Object getItem(int position) { return position; }

    @Override
    public long getItemId(int position) { return position; }
    @Override
    public View getView(final int position, View convertView, ViewGroup parent) {
        final quejas_adapter_result.viewHolder holder;
        if (convertView == null) {
            holder = new quejas_adapter_result.viewHolder();
            convertView=inflater.inflate(R.layout.recycler_oq, root: null);
            holder.statusq=(TextView)convertView.findViewById(R.id.tv_estatus);
            holder.quejaq=(TextView) convertView.findViewById(R.id.tv_NOrden);
            holder.contratoq=(TextView)convertView.findViewById(R.id.tv_NContrato);
            holder.nombreq=(TextView)convertView.findViewById(R.id.tv_Nombre);
            holder.direccion=(TextView)convertView.findViewById(R.id.id_direccion);
            holder.control=(TextView)convertView.findViewById(R.id.controla);
            convertView.setTag(holder); }
```

*Ilustración 12 Ejemplo de ArrayAdapter (Controlador)*


*Ilustración 13 Ejemplo Array adapter Vista (Tipos de servicio sección quejas)*

Para el diseño de pantallas en reportes y geolocalización tomamos de referencia el diseño de la página web ya que es la referencia que tenemos al momento.

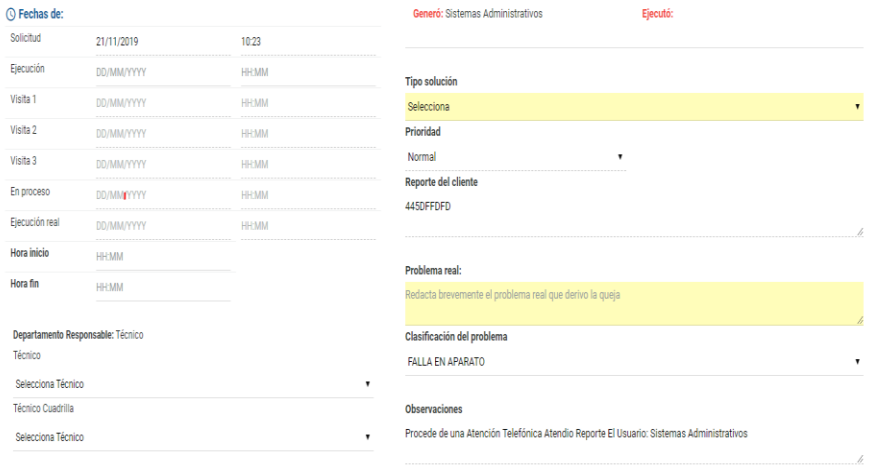



*Ilustración 14 Diseño del sistema Web (Ejecución Reportes)*

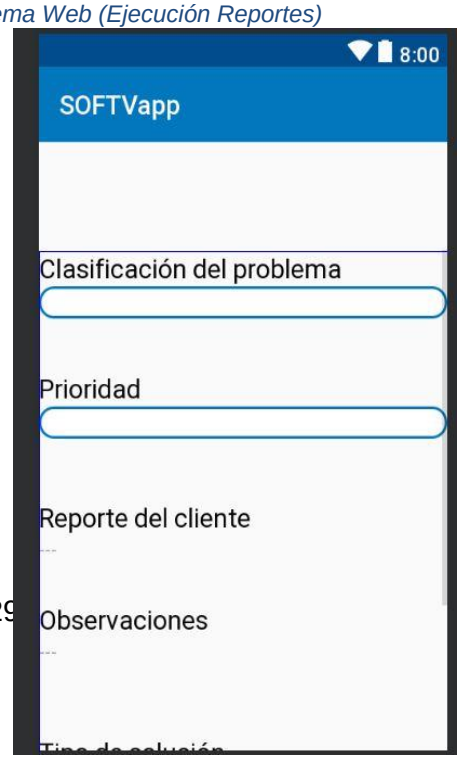



*Ilustración 15 Diseño del sistema Móvil* *Ilustración 16 Diseño sistema móvil*

Como el técnico nunca tuvo su propio módulo en la aplicación web se tuvo que crear tablas, procedimientos y servicios propios del mismo en las cuales venía, trabajos del día, datos del trabajo siguiente, lista de órdenes y quejas.

Teniendo la principal base de la aplicación ya solo faltaba crear los diseños de los diversos trabajos que debe realizar el técnico junto a sus respectivos request y MVC de cada trabajo, Instalaciones (6tipos), cambio de domicilio, cambio de aparato.

**Cronograma de actividades**

| **Actividades por Quincena** | **Ago -1a** | **Ago- 2a** | **Sept – 1a** | **Sept – 2a** | **Oct – 1a** | **Oct- 2a** | **Nov – 1a** | **Nov. – 2a** | **Dic- 1a** |
|---|---|---|---|---|---|---|---|---|---|
| Creación de menú principal y diseño de pantallas principales | | | | | | | | | |
| Obtención de datos de la base de datos | | | | | | | | | |
| Creación de tablas propias de la aplicación, procedimientos y WFC Services | | | | | | | | | |
| Creación de módulos de los diversos reportes que realiza el técnico | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Ejecución de reportes | | | | | | | | | |
| Correcciones de errores de código y diseño | | | | | | | | | |

*Tabla 1 cronograma de actividades*

## *CAPÍTULO 5: RESULTADOS*

### *12. Resultados*

| ***Objetivo Especifico*** | ***Resultado*** |
|---|---|
| Implementar una interfaz fácil de manipular y aceptación de usabilidad para el usuario haciendo uso de pantallas fragmentadas. | Se logró la creación de las pantallas y clases fragmentadas para el mejor uso y manipulación del sistema móvil, haciéndolo eficiente y rápido para el procesamiento del sistema en configuración de procesadores de baja gama. Obtuvimos la mejor aceptación en sistemas Android con APIS 21 en adelante, mostrando una buena fluidez en el sistema. |
| -Geo localizar al usuario para obtener la ubicación exacta de los clientes visitados. | Se logró el objetivo de implementar el sistema GPS en la aplicación móvil obteniendo una de las mejores y más exactas coordenadas en el los sistemas Android haciendo uso de las librerías establecidas. Obteniendo como resultado la ubicación exacta del usuario y del cliente visitado, quedando registrado el base de datos del sistema softv web |
| -Crear pantalla principal con fácil usabilidad para el uso de Reportes y Órdenes | Obtuvimos una gran aceptación por el coach del proyecto y del usuario final haciéndolo amigable y fácil de comprender por el usuario. |

*Tabla 2 Resultados*

## *CAPÍTULO 6: CONCLUSIONES*

### *13. Conclusiones del Proyecto*

Gracias a esta aplicación la empresa dio un avance muy grande haciendo de su sistema una de los mejores  y más completos sistemas del país y latino américa, ya que es n sistema totalmente completo desde recursos humanos hasta la parte del capital humano.

Aún sigue la aplicación en constante mantenimiento según los requerimientos de cada empresa y se está optando por usarla en cada trabajo que realice un técnico en cada domicilio.

En cuanto el aprendizaje se obtuvieron nuevo conocimientos en base de datos en cuanto a manejo de tablas, creación de nuevos procedimientos, control de usuarios, así como creación y modificación de servicios web api rest para consumo de información y mejoramiento de la lógica al momento de programar y de esta manera realizar los cambios de una manera más eficiente y en menor tiempo.

## *CAPÍTULO 7: COMPETENCIAS DESARROLLADAS*

### *14. Competencias desarrolladas y/o aplicadas.*

1. Aplicar habilidades de programación orientada a objetos que fueron necesarias para la reutilización de códigos que eran necesarios, también tener un código más estructurado con clases y herencias.
2. Implementación de Procedimientos almacenados que involucran inner join para tener consultas cruzadas con las tablas catálogo y las tablas maestras.
3. Creación de WCFServices en C# con varias capaz que nos permiten conectarnos a la base de datos, esto hace que los servicios regresen datos de los procedimientos almacenados en formato JSON.
4. Implementación de MVC en android para el manejo óptimo de vistas, modelos y controladores
5. Aplica el uso de modelos tipo POJO que ayudan al mejor manejo de información recibida por los WCFServices.
6. Creación, modificación y envío de JSONObjet y JSONArrays.
7. Aplicación de librerías Retrofit para conectar la aplicación con los WCFServices.
8. Implementa diversos diseños para pantallas con diferentes densidades de pantalla.
9. Implementa diseños con ConstrainLayout que permiten la adaptación de pantallas.
10. Implementa una splash activty que permite tener una pantalla de inicio más amigable.
11. Creación de un sharedpreferences para guardar datos de sesión aunque la aplicación se haya cerrado.
12. Aplique adapters para la creación de listas con clicklistener.
13. Utiliza las nuevas tecnologías de información y comunicación en la organización, para optimizar los procesos y la eficaz toma de decisiones.
14. Creación de fragment para el uso de swipe con lo cual podemos movernos por diversas pantallas.
15. Implementa módulo para poder tomar fotos de una instalación y guardarlas en el servidor.

16. Creación de sitios públicos dentro de servidores para poder acceder a los WCFServers.
17. Firmar aplicación móvil para tener un control de seguridad dentro de la aplicación al publicarlas en la Playstore.
18. Creación de flavers con las cuales podemos tener un control de versiones dentro de la misma aplicación.
19. Creación de archivos .abb para publicar la aplicación dentro de la Playstore y poder disminuir el tamaño de la aplicación un 30%.
20. Publicar aplicación en la Playstore

## Referencias

Android Developers. (01 de 12 de 2019). *Android Developers*. Obtenido de Android Developers: https://developer.android.com/guide/components/fragments?hl=es-419

ECMA-404 The JSON Data Interchange Standard. (21 de Febrero de 2017). *Introducción a JSON.* Obtenido de https://www.json.org/json-es.html#:~:targetText=JSON%20(JavaScript%20Object%20Notation%20%2D%20Notaci%C3%B3n,es%20simple%20interpretarlo%20y%20generarlo.&targetText=Estas%20propiedades%20hacen%20que%20JSON,para%20el%20intercambio%20de%20datos.

INEGI. (2 de Abril de 2019). *COMUNICADO DE PRENSA INEGI NÚM. 179/19.* Obtenido de https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2019/OtrTemEcon/ENDUTIH_2018.pdf

La evolución de Internet en México y su impacto en el ámbito educativo (De 1986 a 2006). (1 de Enero de 2006). *Studying flows to predict shapes.* Obtenido de https://www.fergut.com/la-evolucion-de-internet-en-mexico-y-su-impacto-en-el-ambito-educativo-de-1986-a-2006/

La Universidad de Alicante. (2 de Diciembre de 2019). *Modelo vista controlador (MVC).* Obtenido de https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html#:~:targetText=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos.