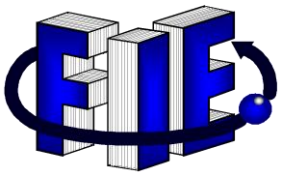

Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

Portal de Alumnos SIIA Móvil para dispositivos Android

TESIS

Que para obtener el título profesional de

INGENIERO EN COMPUTACIÓN

Presenta

Fermín Ruiz Ruiz

Asesor de tesis

Ing. Bertha Georgina Flores Díaz

Morelia, Michoacán., Febrero del 2020

Dedicatoria

Este trabajo se lo dedico con todo mi amor y cariño a mi familia en especial a mis padres, que fueron siempre el impulso que necesite durante el proceso de mis estudios y que hoy han visto como todas las apuestas que hicieron por mí, rindieron cuentas. Les dedico mi trabajo a ellos por su enorme apoyo y su confianza, por creer en que yo podría lograr una meta de esta magnitud, a mis hermanas les dedico este logro por ser ese apoyo femenino inigualable que siempre necesité, a mis hermanos y en especial a mi hermano menor, para que tenga un referencia de la cual apoyarse y al igual que yo les regale a mis padres un logro mejor que el mío. Siempre he presumido de mi familia, la nobleza y la humildad, el trabajo y la honestidad, el valor y la perseverancia, los amo.

Agradecimientos

Le agradezco a la Universidad Michoacana de San Nicolás de Hidalgo por abrirme las puertas de sus aulas y permitir que sus profesores me enseñaran y ayudaran en mi formación como profesionista.

Gracias a todas las y los profesores que fueron parte de mi desarrollo, les agradezco el haberme enseñado todo lo que fue necesario para terminar mi formación y poder culminar este logro.

Gracias a mi asesora por enseñarme y ayudarme en mi último paso para que yo lograra concluir mi meta.

Gracias a mi familia por todo su apoyo durante el tiempo que estuve fuera de casa, que siempre fui bien recibido cuando volvía a ella y que siempre me demostraban de manera cariñosa que se me echaba de menos. También, gracias por no negarse a seguir apoyándome a pesar de los obstáculos que algunas veces esto representaba, por hacer que no me rindiera y seguir animándome.

Gracias a mis amigos que siempre estuvieron a mi lado dándome sus ánimos, a los que conocí durante mi trayecto como estudiante, por apoyarme, trabajar y hacerme sentir que no estaba solo, gracias por aun permanecer conmigo.

Índice

Índice de Ilustraciones

Índice de códigos

Resumen

La presente tesis describe el proceso de desarrollo de la aplicación móvil “Portal de Alumnos SIIA” para dispositivos Android. La aplicación tiene el propósito de cubrir parte de las necesidades que presenta el Sistema Integral de la Información Administrativa (SIIA) de la Universidad Michoacana de San Nicolás de Hidalgo. El proyecto aborda la parte del portal de alumnos, donde se puede consultar las calificaciones, el kardex, las deudas, etc.

La principal razón del desarrollo de este proyecto es que los alumnos puedan contar con una herramienta alternativa para el proceso de consultas, que sea más sencilla, rápida, intuitiva y amigable que la actual aplicación web del SIIA.

El proceso de desarrollo fue llevado a cabo con herramientas que ofrece Google para los desarrolladores de aplicaciones Android. El lenguaje de programación que se utilizó es Java y se hizo uso del soporte de librerías necesarias para el desarrollo de aplicaciones móviles Android. El conjunto de pruebas y resultados obtenidos, confirma la facilidad de uso, la rapidez de consulta y la satisfacción por parte de los alumnos al usar la aplicación.

Palabras clave: Sistema Operativo, Entorno de Desarrollo Integrado, Kit de Desarrollo de Software, Lenguaje de Marcas Extensible, Herramientas de Desarrollo Android.

Abstract

This thesis describes the process of developing the mobile application “Portal de Alumnos SIIA Móvil” for Android devices. The application is intended to cover part of the needs presented by the Sistema Integral de la Información Administrativa (SIIA) of the Universidad Michoacana de San Nicolás de Hidalgo. The project addresses the part of the student portal, where students can check grades, kardex, debts, etc.

The main reason for the development of this project is that students can have an alternative tool for consulting that is simpler, faster, more intuitive and friendlier than the current SIIA web application.

The application was developed with Google for Android application developers tools. The programming language used is Java and the necessary library support was used for the development of Android mobile applications. The set of tests and results obtained confirms how easy it is to use, how fast consulting is and the students’ satisfaction when using the application.

Capítulo 1 Introducción

En el presente trabajo se presenta la propuesta del desarrollo de una aplicación móvil para dispositivos Android del Sistema Integral de Información Administrativa (SIIA) de la Universidad Michoacana de San Nicolás de Hidalgo (UMSNH). Se presenta como una opción más para los estudiantes, en la que podrán realizar algunas de las tareas que ofrece el portal de alumnos del sitio web del SIIA. En el portal Web, los estudiantes pueden realizar consultas de calificaciones, deudas y cursos o bien darle seguimiento a algunos trámites como de servicio social y titulación, resolver encuestas, etc. La aplicación aborda algunas de las tareas más comunes y de mayor facilidad que el alumno realiza durante su estancia, como lo son las consultas de: calificaciones, deudas y cursos.

El desarrollo de este proyecto se realizó por interés de la UMNSH de ofrecer una herramienta estudiantil rápida, fácil y eficiente que no interrumpan el tiempo de un estudiante, pero que finalmente entregue los mismos resultados que se obtienen de las herramientas ya existentes por parte del SIIA. Otros de los inconvenientes que se tenían y se consideraron para llevar a cabo el desarrollo de la aplicación es que el portal de alumnos del SIIA tiene la desventaja de que el contenido del sitio web no se adapte a los diferentes tamaños de las pantallas de los teléfonos inteligentes y el usuario tenía que hacer zoom con sus dedos para poder verlo. Debido a este problema se pensó en el desarrollo de este proyecto para que el contenido solicitado por el usuario tenga mejor presentación y sea responsivo. Se espera que con el desarrollo de esta aplicación aparte de que sea cómoda para los estudiantes, se abran más opciones para mejorar su funcionamiento donde se podrían incluir más funcionalidades como: cancelar algunas deudas, responder la encuesta sobre los profesores, iniciar los trámites de servicio social y titulación, búsquedas de calificaciones, etc., por mencionar algunas.

La investigación de campo se realizó en el trabajo de tesis “Portal De Alumnos SIIA Móvil para dispositivos móviles con sistema operativo iOS” elaborado por el Ing. José Enrique Montañez Villanueva, donde se realizaron encuestas a los estudiantes sobre que opinaban y si era necesaria una aplicación que permitiera realizar las tareas ya

mencionadas. A lo largo del desarrollo del proyecto, se decidieron las herramientas de trabajo a usar, y se hizo uso de los conceptos sobre el lenguaje de programación Java que se aprendieron durante la estancia como estudiante.

1.1 Planteamiento del problema

El Sistema Integral de la Información Administrativa de la Universidad Michoacana de San Nicolás de Hidalgo fue creado con el fin de ofrecer un sistema integral, donde la comunidad universitaria pueda contar con un conjunto de servicios administrativos, personales, escolares y financieros. Los usuarios pueden acudir a este sistema para gestionar algún trámite o consulta. Los servicios se orientan con las necesidades de los alumnos, académicos, administrativos y/o manuales.

El presente trabajo se enfoca específicamente con las necesidades académicas y administrativas de los alumnos. Con el desarrollo de una aplicación móvil para el portal de alumnos el SIIA se pretende pasar de ser un proceso presencial y tedioso a un proceso rápido y ágil que se pueda llevar a cabo desde cualquier dispositivos móvil con conexión a internet.

El listado siguiente muestra algunos de los servicios que se le ofrecen al alumno:

- Trámites para el servicio social
- Pagos diversos
- Consulta de calificaciones
- Consulta de deudas
- Tramites de titulación, etc.

La gran mayoría de la población estudiantil de la Universidad Michoacana cuenta con un dispositivo móvil inteligente y conexión a internet, por lo que se decidió desarrollar la aplicación móvil para el portal de alumnos del SIIA donde se contemplan las tareas más

comúnmente realizadas por los alumnos, que son: consulta de deudas, consulta de materias y consulta de calificaciones.

1.2 Antecedentes

"Corría el año 1998 cuando la compañía sueca de telecomunicaciones, Nokia, decidía incorporar una pequeña aplicación a sus móviles para que sus usuarios pudieran matar el tiempo durante las cotidianas esperas en la cola del supermercado, la del autobús o en un trayecto en tren. Se trataba de la réplica de un antiguo videojuego que había causado furor en los años 70. Su nombre es 'La Serpiente'". [1]

Las aplicaciones web o como también son conocidas páginas o sitios web, son el panorama del internet, son lo que le dan vida a ese concepto, desde sus inicios los sitios web pueden ser visitados con un programa llamada navegador de internet como ejemplos se tiene a Google Chrome, Safari, Opera, Mozilla, Internet Explorer, etc.

El desarrollo de aplicaciones web era bastante "sencillo", se basaba en mostrar su contenido en algunas pocas imágenes y texto plano, sin algún tipo de formato que le diera un poco más de colorido. Pero en una era llena de avances tecnológicos el desarrollo de aplicaciones no podía quedarse atrás, y debido a la implementación de nuevas tecnologías para el desarrollo de estas, fue que las aplicaciones web fueron evolucionando, hasta tener lo que ahora se puede ver al abrir un navegador de internet, miles de páginas, coloridas, dinámicas, animadas e interactivas, ahora es un mar lleno de diseños, ideas y herramientas aplicadas.

Hoy en día las aplicaciones móviles están a la par con el desarrollo de teléfonos inteligentes (*Smartphones*) para trabajar en conjunto y ofrecer herramientas de trabajo eficientes, como en la implementación de aplicaciones móviles que podrían sustituir algunas aplicaciones web.

Con el desarrollo de los primeros teléfonos celulares se podían realizar tareas como realizar llamadas o enviar mensajes de texto, pero con el desarrollo de los teléfonos inteligentes, ahora se cuenta con dispositivos con mayor potencial para realizar tareas más complejas como administrar correos, organizar agendas, navegar por Internet, entre otras. Con el paso del tiempo los teléfonos inteligentes se han convertido en una de las principales herramientas de entretenimiento, comunicación y productividad para la población en general.

Los expertos y entusiastas de la tecnología al ver que estos dispositivos móviles tenían un gran potencial, quisieron aprovecharlo desarrollando herramientas que lo explotaran al máximo poniendo siempre de frente la movilidad, surgiendo así la era del desarrollo de las aplicaciones móviles o bien conocidas como Apps. [2]

Posteriormente, grandes empresas como Apple y Google se unieron a este nuevo campo con sus respectivas plataformas para dispositivos móviles: iOS y Android; las cuales además del desarrollo de aplicaciones daban la opción de publicar dichas Apps para que el usuario al final pudiera comprarlas y descargarlas directamente en su dispositivo. [2]

No solo se desarrollan aplicaciones sin conexión a internet como, las calculadoras o algunos juegos, sino que también se han desarrollado aplicaciones que hacen uso de este recurso como juegos online, Facebook móvil, Google Chrome, etc., gracias a que el internet permite un gran campo de desarrollo para las Apps, estas hacen uso de internet para poder dar comodidad a las personas que son usuarios de estas tecnologías.

La forma en que el SIIA ofrecía sus servicios para los alumnos, desde sus orígenes, era de forma presencial, es decir, el interesado en este caso el alumno, tenía que acudir a las oficinas del SIIA a realizar los trámites o consultas deseadas, haciendo de esto una demora a la hora de esperar a que el alumno fuera atendido, de la misma manera para cada alumno que necesitara realizar algún trámite o consulta, además de que esto implicaba mucho trabajo por parte de los empleados del SIIA.

Con el paso del tiempo se optó por desarrollar una aplicación web que ofreciera los mismos servicios que se ofrecían de forma presencial, es decir, que todo estuviera disponible en una página web, esto para evitar la problemática del traslado de los alumnos hasta las oficinas del SIIA. Cuando se realizó la implementación de la página Web del SIIA, parecía que todo estaba resuelto siempre y cuando el interesado contara con un ordenador y conexión a internet. Con el desarrollo y avance de nuevas tecnologías, como teléfonos inteligentes, la página Web se volvió obsoleta en el aspecto de la visualización del contenido. Lo anterior ocasiono que la página del SIIA pasara de ser una herramienta útil a una herramienta tediosa para ver su contenido mediante un dispositivo móvil. No se había considerado que se pudiera requerir una herramienta que adaptara el contenido a una pantalla mucho más chica como la de un celular. Por lo que al querer visualizar el contenido del sitio web se tenía que hacer zoom con los dedos a la pantalla para poder visualizar de manera más clara el contenido de la misma. Ahora existe una forma de visualizar mejor el contenido en todo tipo de dispositivos móviles, estas herramientas son las aplicaciones móviles responsivas.

Dadas las herramientas y recursos necesarios para el desarrollo de aplicaciones móviles responsivas, se ha logrado que hasta el momento se cuente con el desarrollo de la aplicación del portal de alumnos del SIIA pero solo para el sistema operativo iOS, por lo tanto es necesario que se lleve a cabo el desarrollo de la aplicación para dispositivos móviles con sistema operativo Android.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar una aplicación móvil para Android con la finalidad de ofrecer una alternativa al portal de alumnos del sitio web del SIIA de la UMSNH, donde los alumnos realicen consultas de calificaciones, créditos, deudas y carga de materias de una manera más rápida, sencilla y cómoda.

1.3.2 Objetivos específicos

Para poder desarrollar en su totalidad el contenido de esta tesis, han sido requisito realizar los siguientes objetivos específicos:

- Descargar e instalar el Entorno de Desarrollo Integrado (*IDE*) necesario para desarrollar el proyecto.
- Instalar y configurar los *Frameworks* necesarios para trabajar de manera eficiente dentro del IDE.
- Crear el proyecto con las plantillas y el lenguaje de programación necesarios dentro del IDE.
- Implementar el diseño de interface ya establecido para el desarrollo de la aplicación basado a sistemas operativos iOS.
- Poner a disposición de la comunidad estudiantil de la UMNSH la aplicación desarrollada.

1.4 Justificación

El desarrollo de la aplicación móvil para el portal de alumnos del SIIA que ofrece la Universidad Michoacana de San Nicolás de Hidalgo, es de gran ayuda para toda la comunidad estudiantil, ya que al contar con una interfaz sencilla de entender y fácil de manejar, resulta más práctica y rápida que visitar el sitio web.

Cabe aclarar que la aplicación móvil tiene el objetivo de ayudar a los estudiantes y no para afectar los ingresos de los lugares que ofrecen computadoras con conexión a internet, ya que al ser un aplicación portátil, los usuarios no tendrán la necesidad de buscar conexión a internet por medio de algún ordenador, pues el consumo mínimo de datos móviles también está considerado en el desarrollo de la aplicación.

Para el desarrollo de la aplicación se ha utilizado Android Studio que es el IDE oficial para el desarrollo de aplicaciones Android, anteriormente se hacía uso de Eclipse que es un entorno de desarrollo multiplataforma, la principal desventaja de Eclipse es que

para poder programar aplicaciones Android, tiene que hacer uso de un plugin llamado Herramientas de Desarrollo Android (*Android Developer Tools*, ADT por sus siglas en inglés). Eclipse tiene licencia de software libre al igual que Android Studio.

A continuación se mencionan algunas diferencias entre Android Studio y Eclipse:

- Android Studio tiene sistema de construcción Gradle a diferencia de Eclipse que trabaja con ANT. Gradle y ANT son herramientas de compilado y creación de programas Java. Gradle reúne en un solo sistema las mejores prestaciones de otros sistemas de compilación como ANT y Maven.
- Construye y gestiona los proyectos basados en Maven.
- Soporte para programar aplicaciones para Android Wear (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Cuenta con soporte para Google Cloud Platform (es una plataforma que permite crear, implementar y escalar aplicaciones, sitios web y servicios en la misma infraestructura que Google).
- Cuenta con vista en tiempo real de renderizado de layouts.
- Contiene nuevos módulos en el proyecto, se actualiza constantemente y tiene mayor rendimiento.
- Es más rápido que eclipse e intuitivo a la hora de usarse.
- Crea emuladores sin exigir demasiados recursos
- Utiliza *ProGuard* para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
- Crea licencias.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.

Estos puntos fueron considerados según Andrea Ardións y Academia Android [3] y [4].

1.5 Metodología

Este trabajo de tesis fue desarrollado en cuatro etapas:

a) Fundamentación teórica.
b) Selección de herramientas.
c) Desarrollo del proyecto.
d) Pruebas de la ejecución de la aplicación.

Estas etapas se describen a continuación.

1.5.1 Fundamentación teórica

Para llevarse a cabo el desarrollo de este trabajo de tesis ha sido necesario, recabar información acerca de las aplicaciones móviles Android y como desarrollarlas. Dentro del tema de las aplicaciones móviles se investigaron los elementos teóricos como:

- Funcionamiento de las aplicaciones móviles.
- Software necesario para el desarrollo.
- Evolución de las herramientas para desarrollo de aplicaciones móviles.
- Distintos lenguajes de programación para el desarrollo de aplicaciones móviles.

1.5.2 Selección de herramientas

En esta etapa y basándose con los fundamentos teóricos sobre el desarrollo de aplicaciones móviles, se hizo la selección de la herramienta más apropiada para el desarrollo de estas. La herramienta seleccionada fue Android Studio por ser el Entorno de Desarrollo Integrado (*IDE*) oficial de Google, ya que además de ser el IDE oficial ofrece una gran cantidad de herramientas, librerías y componentes. Android Studio cuenta con un área de trabajo amplia donde todo está organizado para que el usuario este cómodo a la hora del desarrollo de las aplicaciones.

1.5.3 Desarrollo del proyecto

En esta etapa se desarrollo el proyecto en su totalidad cumpliendo con los requerimientos generales propuestos. Se desarrollo la aplicación móvil con una interfaz fácil de usar e intuitiva y que cuenta con el conjunto de tareas a realizar las cuales son: consulta de calificaciones, consulta de carga de materias y consulta de deudas. Se hizo uso de los conceptos teóricos previamente estudiados y también se aplicaron los conocimientos adquiridos durante el proceso como estudiante.

El encargado de desarrollo del SIIA definió el formato de las vistas, ventanas, botones, acciones de los botones, imágenes, colores y leyendas utilizadas en el diseño de la aplicación, por lo que no fue necesario realizar un análisis de requerimientos estéticos, definir perfiles de usuarios, ni tampoco se participó en el diseño del sistema.

1.5.4 Pruebas de la ejecución de la aplicación

Durante el proceso de las pruebas del proyecto concluido, se analizo que la aplicación móvil corriera correctamente y ejecutara las tareas de manera rápida y eficaz, también se analizo que el contenido y la interfaz fuera responsivos para los distintos tamaños de pantallas en los dispositivos donde se realizaron las pruebas. Se pusieron a prueba todos los elementos y funciones implementados como lo son, las peticiones con o sin conexión a internet, prueba de las alertas de errores de conexión, de campos incompletos y de información inexistente, además también se puso a prueba que la aplicación funcionara en dispositivos con versiones de Android mayores o iguales a la versión especificada. Finalmente con las pruebas terminadas y aprobadas en su totalidad se dio por concluido el proyecto.

Cabe mencionar que el trabajo de campo ya no fue necesario realizarlo, debido a un estudio previo realizado en el trabajo de tesis “Portal De Alumnos SIIA Móvil” para dispositivos móviles con sistema operativo iOS por el Ing. José Enrique Montañez Villanueva, donde se realizó la investigación de campo para determinar la necesidad del

desarrollo de una aplicación móvil para el portal de alumnos del SIIA de la Universidad Michoacana de San Nicolás de Hidalgo, indistintamente del sistema operativo.

1.6 Descripción de los capítulos

En el capítulo 2 se tiene el marco teórico donde se describen todas las herramientas utilizadas para el desarrollo de la aplicación, desde el entorno de trabajo hasta las herramientas de programación y diseño utilizadas, como lo son las librerías, los lenguajes de programación, etc. Se hace una mención breve de los sistemas externos a la aplicación pero que tienen un papel importante, como lo son las bases de datos y los Web Service, como funcionan y cuál es la relación entre todos los elementos que componen este sistema de envió y recepción de datos. Se hace una breve descripción de cómo es el funcionamiento y el enlace entre la parte programable y la parte de diseño de la aplicación. También se realizan algunas comparaciones para justificar las librerías y entorno de trabajo utilizadas.

En el capítulo 3 se describe el desarrollo del proyecto, se habla de los segmentos de código más relevantes para el desarrollo de la aplicación y cómo funcionan. Además se describe la funcionalidad de las diferentes vistas o ventanas de la aplicación. Se describen también las consultas que se realizan al servidor para poder obtener los datos y trabajar con esa información para usarla y/o presentarla en la interfaz.

En el capítulo 4 se presenta un conjunto de pruebas realizadas en distintos dispositivos con diferente tamaño de pantalla para poder verificar que sea responsivo y que además el funcionamiento de la aplicación sea como se describe en el capítulo 3.

En el capítulo 5 se tienen las conclusiones finales y se describen las mejoras que se esperan a futuro para hacer de la aplicación un sistema más robusto y más completo que ofrezca mas funciones para los estudiantes.

Capítulo 2 Marco teórico

Existen distintos entornos para desarrollar aplicaciones Android, sin embargo, tienen como lenguaje de programación nativo a Java, también es utilizado XML como lenguaje de diseño. El Kit de Desarrollo de Software (*Software Development Kit*, SDK por sus siglas en inglés) necesario para el desarrollo de estas aplicaciones viene en conjunto con el Entorno de Desarrollo Integrado (*Integrated Development Environment,* IDE por sus siglas en inglés) oficial para el desarrollo de aplicaciones Android, este IDE es Android Studio.

En la *Ilustración 1* se muestra un esquema de todo el sistema que forma parte del funcionamiento de la aplicación.

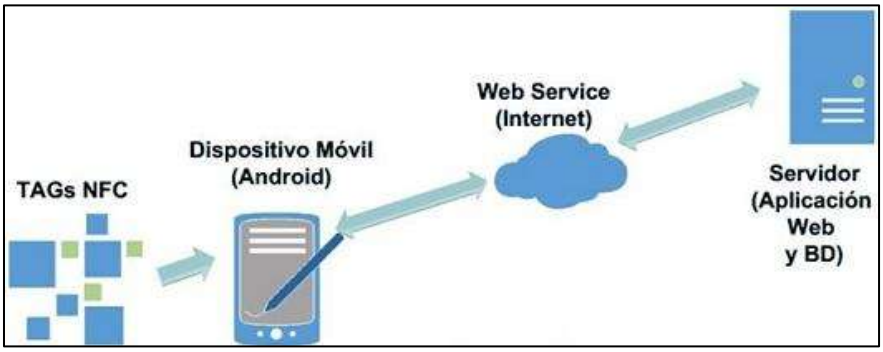

Ilustración 1Interaccion de la aplicación móvil con el servidor.
Fuente: [5]

2.1 Aplicación Móvil

La definición de Aplicación Móvil parte del concepto "*aplicación*". Una aplicación es un programa de computadora que ofrece interacción con el usuario, para poder realizar un conjunto de tareas o actividades. También son conocidas como software de aplicación. Existen diversos conceptos para agrupar lo tipos de aplicaciones, como las aplicaciones de escritorio, las aplicaciones web y las aplicaciones móviles. Para entender un poco en que se diferencia cada tipo de aplicación a continuación se describen brevemente.

Aplicación de escritorio: son las desarrolladas con la finalidad de usar los recursos de computadoras, ya sean de escritorio o portátiles, este grupo de aplicaciones por lo

general son robustas y ofrecen muchas herramientas de trabajo. Por ejemplo Word de la empresa Microsoft.

Aplicación Web: son las aplicaciones que ofrecen contenido o actividades desde un explorador de internet, este tipo de aplicaciones están montadas en servidores remotos. Para poder hacer uso de estas es necesario tener servicio de internet. Por ejemplo se tiene Netflix, que es una plataforma de video en *streaming* de mucha importancia en todo el mundo.

Aplicación Móvil: este conjunto de aplicaciones son desarrolladas con la finalidad de que trabajen en los teléfonos inteligentes (*Smartphones*) haciendo uso de los componentes físicos y de las capacidades o recursos del dispositivo móvil. Por ejemplo la aplicación Facebook que sus inicios solo ofrecía sus servicios mediante una aplicación web y con el paso de los años se desarrolló la versión móvil que es más utilizada.

Entonces se puede determinar que una Aplicación Móvil es un programa de computadora desarrollado para que funcione en Teléfonos Inteligentes. Las aplicaciones Móviles parecen ser el siguiente paso de las Aplicaciones Web, ya que tienen mayor facilidad de uso.

2.2 Android Studio

Android Studio es el IDE oficial para el desarrollo de aplicaciones Android creado por Google. Android Studio ofrece la comodidad de contar con el conjunto de herramientas necesarias para el desarrollo de aplicaciones, evitando empezar desde cero cualquier proyecto. Ofrece elegir la versión en la que se va a desarrollar la aplicación, también permite elegir diseños ya predeterminados para el aspecto de la interfaz de la aplicación en desarrollo. También cuenta con un área de trabajo bien organizada que permite desplazarse entre los archivos de la aplicación, el código, y los diseños de interfaz. Cuenta con dos opciones de lenguajes para el desarrollo de Apps aparte de Java, los cuales son C++ y Kotlin, que son usadas como soporte para darle más variedad de herramientas a Android Studio.

2.2.1 Área de trabajo de Android Studio

La *Ilustración 2* muestra de manera organizada las partes de la ventana de trabajo de Android Studio, donde se realiza el desarrollo de la aplicación.

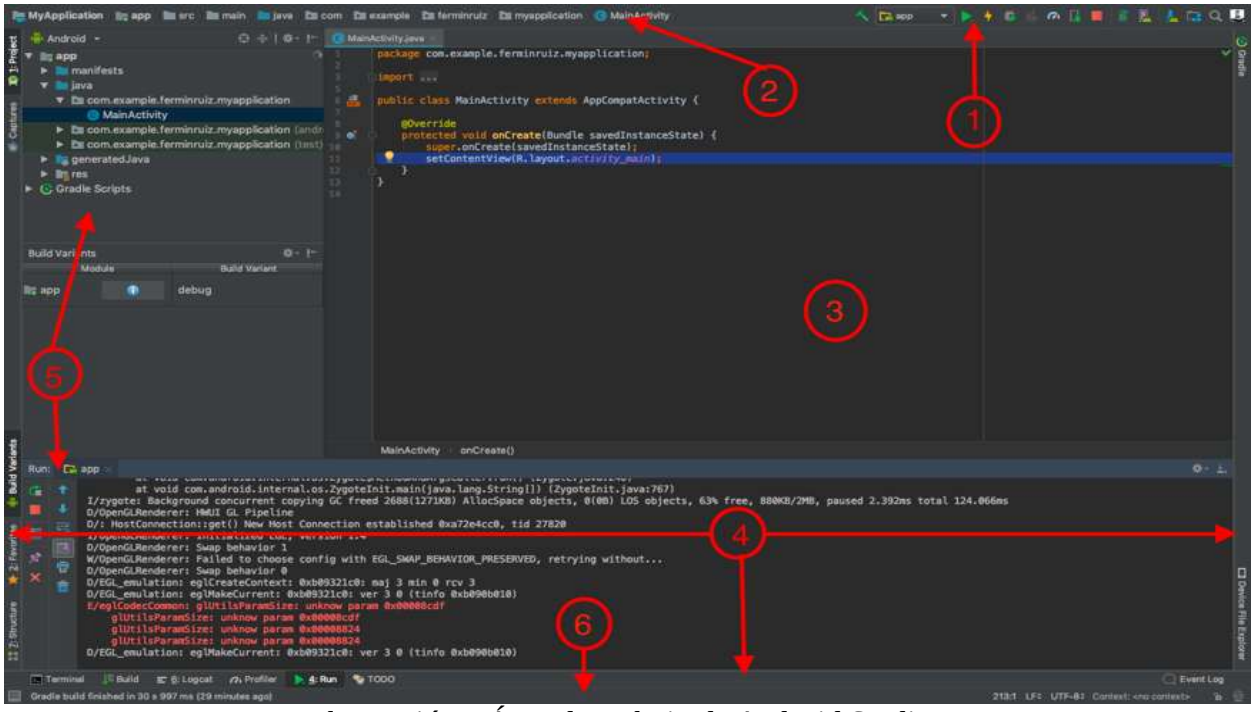

Ilustración 2 Área de trabajo de Android Studio.

Cada sección se describe a continuación:

1. **Barra de herramientas:** permite realizar una gran variedad de acciones, como la ejecución de la app, el debugeo de la misma, y el inicio de herramientas de Android.
2. **Barra de navegación:** ayuda a explorar el proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana proyecto.
3. **Ventana del editor:** es el área donde se puede crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño .xml, el editor muestra el editor de diseño.
4. **Barra de la ventana de herramientas:** se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que permiten expandir o contraer ventanas de herramientas individuales.

5. **Ventanas de herramientas:** permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Se pueden expandir y contraer.
6. **Barra de estado:** se muestra el estado del proyecto y del IDE en sí, como también cualquier advertencia o mensaje. [6]

2.2.2 Actividad (Activity)

Una Activity o Actividad es cada una de las pantallas que forman parte de la aplicación. Son los diseños de interface que muestran el contenido de la aplicación al usuario y es donde el usuario puede interactuar con la App. Las actividades tienen formatos ya predefinidos dentro de las herramientas de Android Studio, como por ejemplo: actividad vacía (*Empty Activity*), actividad para inicio de sesión (*Login Activity*), actividad con barra de desplazamiento (*Scrolling Activity*), etc. Por lo general las actividades vacías son las usadas para desarrollar la interface deseada ya que al no contener un formato, se les puede diseñar uno.

Las actividades al ser creadas generan dos archivos de distintas extensiones, uno para la parte de programación lógica que son los archivos Java, C++ o Kotlin, (en este caso se ha tomado Java) y los archivos XML que son los archivos de diseño de interfaz.

2.2.2.1 Archivos Java

Los archivos con extensión .java son los documentos que contienen el código encargado de realizar la parte lógica de la aplicación. Se encargan de realizar las tareas como procesar datos, devolver datos, enviar información, recibir información, etc. En el ámbito de desarrollo de aplicaciones son la parte importante, ya que si la parte lógica programada en estos archivos no funciona correctamente podría generar errores en la aplicación, haciendo que la aplicación no entregue los resultados deseados o que simplemente esta no se pueda ejecutar.

2.2.2.2 Archivos XML

La característica de los archivos .xml es que trabajan la parte grafica o de diseño de la aplicación, es decir, son los archivos donde se especifican las elementos que se utilizan en la interfaz de la aplicación, como los botones, las etiquetas, los textos, etc.

El formato de los archivos .xml son grupos de texto entre campos y estos se limitan entre pares de etiquetas, algo parecido a HTML. En la *Ilustración 3* se muestra un ejemplo de un etiquetado XML en base al tema de aplicaciones móviles Android.

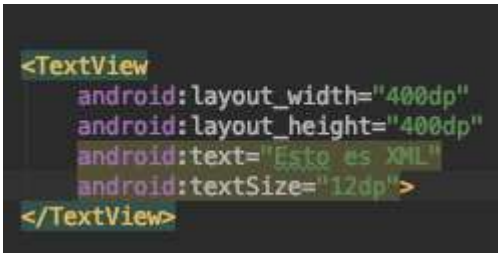

Ilustración 3 Ejemplo de formato XML.

2.2.2.3 Interacción entre archivos Java y archivos XML

Las Actividades generadas contienen la parte lógica y la de diseño, porque es así como existe el intercambio de información entre dichos archivos. Los archivos .xml reciben eventos del usuario, datos, muestran información, etc. Los archivos Java hacen uso de una función en específico para hacer referencia a su parte de diseño. En el segmento de *Código 2.1* se da el ejemplo de la función encargada de realizar el enlacen entre dichos archivos.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_inicio);
}
```

Código 2.1 Ejemplo de uso de la función setContentView().

La parte lógica procesa la información o realiza las actividades necesarias para poder entregar de forma correcta los resultados a la parte de diseño.

2.2.3 Panel de diseño de Interfaz de Usuario (UI)

La paleta de diseños que ofrece Android Studio para el desarrollo de las interfaces de usuario es bastante amplia, en la *Ilustración 4* se muestra de forma general todos los tipos de herramientas de diseño que ya están predefinidas.

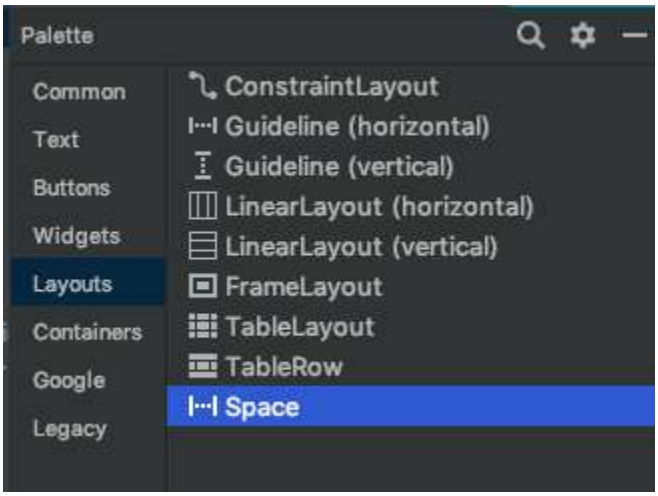

Ilustración 4 Paleta de herramientas de diseño de Android Studio.

2.2.3.1 Vistas (Views)

Las vistas son un objeto de la clase *Android.view.View*. Estos elementos son controles de entrada y/o salida de texto, de selección, distribución de elementos, interacción con elementos físicos del dispositivo, etc. Con la finalidad de que el usuario de Android Studio localice las herramientas necesarias para el desarrollo de la aplicación, estas Vistas están clasificadas en base a su funcionalidad. Los distintos grupos de vistas están separados en *Layouts*, *Buttons*, *Text*, *Legacy*, *Widgets* y *Containers*, tal como se muestra en la *Ilustración 4*. En la *Ilustración 5* se muestra un ejemplo de un elemento de tipo *Buttons*, en la parte superior el elemento gráfico y en la parte superior el código .xml.

Ilustración 5 Ejemplo de un Botón y su código XML.

En la *Ilustración 6* se muestra un ejemplo de un elemento de tipo *Text*, en la parte superior el elemento gráfico y en la parte superior el código .xml.

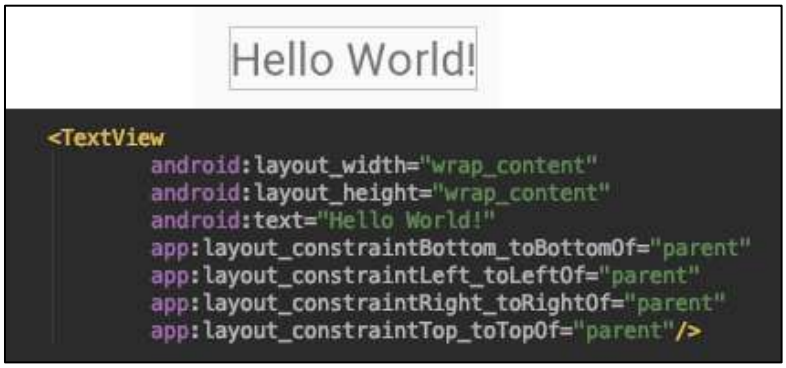

Ilustración 6 Ejemplo de Texto y su código XML.

2.2.3.2 Layouts

Los *Layouts* son los tipos de Vistas que le dan el formato a la interface de la aplicación, es decir, son las áreas que estructuran el contenido grafico que ofrece la aplicación. Son formas cuadradas o rectangulares que pueden contener adentro: botones, iconos, listas, texto, imágenes, etc. Además, pueden definirse más *Layouts* dentro de un *Layout* permitiendo así poder crear interfaces visualmente más complejas. Por defecto al crear una Actividad nueva, el archivo XML generado contiene un *Layout* llamado *ConstraintLayout* y eso permite tener el primer elemento para ir generando la visualización de la interface de usuario.

2.3 Base de datos

Las bases de datos (*Database*, DB por sus siglas en inglés) son estructuras programadas para poder almacenar grandes cantidades de información en distintas tablas de distinto tipo o estructura. Fueron creadas con la necesidad de poder almacenar información para posteriormente poder consultarla.

Las bases de datos pueden ser manejadas con un Sistema Gestor de Base de Datos (SGBD) o *DataBase Manager System*, (DBMS, por sus siglas en inglés), estos sistemas son un conjunto de programas que permiten crear, gestionar, administrar u obtener información de una base de datos, la ventaja de los SGBD es que permiten que varios usuarios pueden acceder a una base de datos al mismo tiempo.

Algunos ejemplos de SGBD son:

- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- MySQL

Para crear y estructurar una base de datos se utiliza el Lenguaje de Definición de Datos (*Data Definition Language*, DDL por sus siglas en inglés), que es un lenguaje que permite definir las estructuras que almacenarán los datos, este lenguaje es ofrecido por cada SGBD.

La manipulación o consulta de información de la base de datos se puede realizar con un lenguaje especial que ofrece cada SGBD, este lenguaje se denomina Lenguaje de Manipulación de Datos (*Data Manipulation Language*, DML por sus siglas en ingles).

2.4 Servicio Web

Un Servicio Web o Web Service es una aplicación Web que permite la interconexión entre los servidores, en este caso la base de datos y los dispositivos móviles, siempre y cuando estos dos equipos cuenten con servicio de internet.

El funcionamiento de un Web Service es sencillo de expresar, simplemente existe un cliente y un servidor, el cliente hace peticiones al Web Service y este se encarga de realizar la consulta al servidor, después el Web Service obtiene respuesta del servidor y por medio de protocolos de mensajería el Web Service entrega la respuesta al cliente mediante un formato de respuesta. Existen distintas maneras de hacer peticiones a un Web Service, de manera directa o mediante envió de parámetros, las peticiones pueden ser de tipo GET, POST, PUT y DELETE.

En el entorno del desarrollo de aplicaciones que necesitan realizar peticiones a bases de datos es necesario que exista un Web Service de por medio.

2.4.1 Tipos de Web Service

2.4.1.1 Servicios Web SOAP

Los servicios Web SOAP, o servicios Web *"big"*, utilizan mensajes XML para intercomunicarse que siguen el estándar SOAP (*Simple Object Access Protocol*), un lenguaje XML que define la arquitectura y formato de los mensajes. Dichos sistemas normalmente contienen una descripción legible por la máquina de la descripción de las operaciones ofrecidas por el servicio, escrita en WSDL (*Web Services Description Language*), que es un lenguaje basado en XML para definir las interfaces sintácticamente. [7]

2.4.1.2 Servicios Web RESTful

Los servicios Web RESTful (*Representational State Transfer Web Services*) son adecuados para escenarios de integración básicos *ad-hoc*. Dichos servicios Web se suelen integrar mejor con HTTP que los servicios basado en SOAP, ya que no requieren mensajes XML o definiciones del servicio en forma de fichero WSDL. [7]

2.5 Librería Volley

Volley es una librería desarrollada por Google para optimizar el envío de peticiones HTTP desde las aplicaciones Android hacia servidores externos. La clase *HttpUrlConnection* también es utilizada para el envió de peticiones a un Web Service. Este componente actúa como una interfaz de alto nivel, liberando al programador de la administración de hilos y procesos tediosos de parsing, para permitir publicar fácilmente resultados en el hilo principal. [8]

La librería *Volley* ha demostrado ser mejor que la clase *HttpUrlConnection* ya que es más fácil de implementar, además de que no se genera código repetitivo y las respuestas son más entendibles esto permiten que el manejo de la información de respuestas sea más rápido, siempre y cuando las peticiones no contengan descargas de archivos demasiado grandes.

Volley cuenta con distintos tipos de respuesta los cuales son: StringRequest, JsonObjectRequest, JsonArrayRequest e ImageRequest.

2.5.1 JSON

Cuando se trata de peticiones a servidores externos se espera recibir una respuesta. Para este proyecto las respuestas se reciben en un formato denominado JSON (*Java Script Object Notación*), es un formato que permite el intercambio de datos entre clientes y servidores basado en la sintaxis de *JavaScript* para representar estructuras en forma

organizada y alto nivel, con el fin de acercar a una definición mucho más amigable por los desarrolladores. JSON es un formato en texto plano independiente de todo lenguaje de programación, es más, soporta el intercambio de datos en gran variedad de lenguajes de programación como *PHP*, *Python*, *C++*, *C#*, *Java* y *Ruby*, por mencionar algunos [9].

En las aplicaciones móviles el uso del formato JSON es muy utilizado ya que es una de las mejores opciones para poder hacer el intercambio de información con los servidores.

En el siguiente ejemplo se muestra el formato que tiene una respuesta de tipo JSON.

```
{
  "IdProducto": 1856,
  "NombreProducto": "ProductoX",
  "Tipo": Tipo1,
  "Precio": 00.00
}
```

El formato está escrito entre pares de clave valor separados por coma (,) y encerrados entre llaves { }.

Capítulo 3 Desarrollo del proyecto

La aplicación ofrece inicialmente una ventana de inicio de sesión donde el usuario, en este caso el alumno, tendrá que ingresar sus datos: *matricula y contraseña*. El inicio de sesión correcto permitirá al alumno observar una segunda ventana como inicio, donde se muestran avisos. La aplicación también cuenta con la ventana de consultas que le permite al alumno poder realizar tres actividades de consulta: calificaciones, cursos y deudas. La ventana de ajustes ofrece al alumno mostrar los datos personales, así como información acerca de la aplicación y la opción de cierre de sesión. En la *Ilustración 7* se muestra el diagrama de bloques de la aplicación.

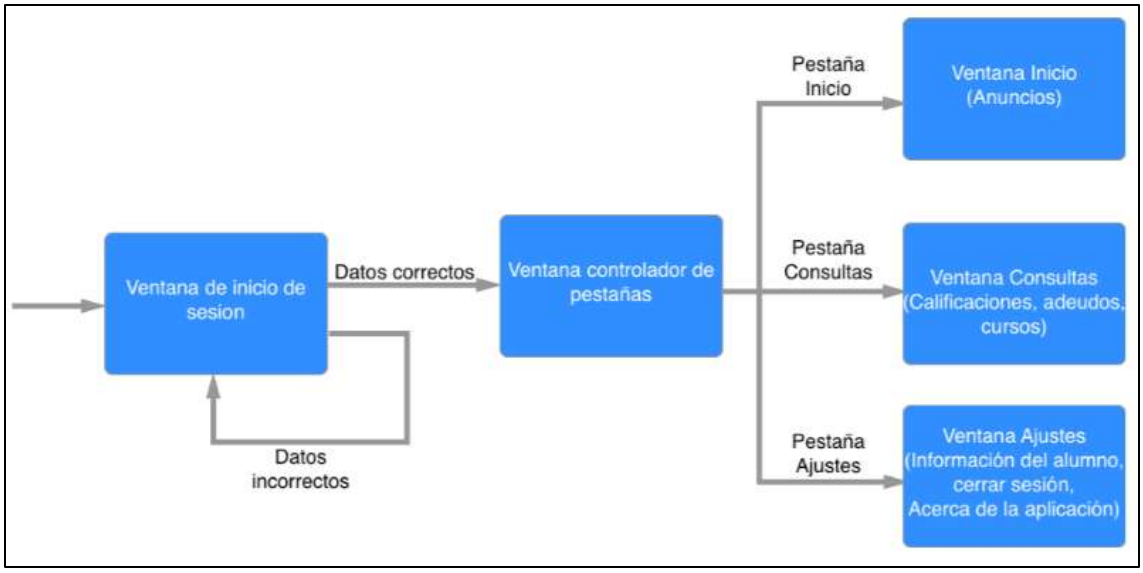

Ilustración 7 Diagrama de bloques de la aplicación.

Fuente: [10].

3.1 Barra de navegación

También se cuenta con una barra de navegación en la parte inferior de algunas ventanas de la aplicación. Este elemento ofrece las opciones para poder desplazarse en las tres ventanas principales de la aplicación, inicio, consultas y ajustes y solo estará presente

en las ventanas de programas, la ventana de calificaciones, ciclos y detalles de ciclos, en la ventana de deudas y detalles de las deudas. La *Ilustración 8* se muestra el diseño de la barra de navegación.

Ilustración 8 Diseño de la barra de navegación.

3.2 Ventana de carga

La *Ilustración 9* muestra la primera ventana en mostrarse al iniciar la aplicación. Cuando la aplicación arranca se muestra el escudo de la UMSNH centrado por algunos segundos.

Ilustración 9 Ventana de carga

3.2.1 Funcionamiento de la actividad Cargando.

El funcionamiento de la ventana está implementado de tal manera que al iniciarse la aplicación, la *actividad Cargando* se inicia de manera automática y a la par el método *postDelayed* que se muestra en el segmento de *Código 3.1*. El método *postDelayed* recibe dos parámetros los cuales son una interface *Runnable* y un número que indica el tiempo en milisegundos. El primer parámetro que es la interface *Runnable* simplemente se encarga de contener el método *run* y dentro de este se realizan tres tareas. La primera tarea crea un *Intent* para invocar la siguiente actividad que es la actividad *Login*, la siguiente tarea es iniciar el *Intent* con el método *startActivity* y finalmente el método *finish* para terminar la actividad actual. El número 2000 es el segundo parámetro del método *postDelayed* e indica el tiempo que se tiene que esperar para poder ejecutar el método *run*, en este caso el tiempo tiene un valor de 2000 milisegundos equivalentes a 2 segundos. Al terminar el tiempo especificado de segundos de espera y posteriormente ejecuta el método *run*, la actividad actual (*actividad Cargando*) se cierra y se muestra la ventana siguiente que es la ventana de *Inicio de sesión*.

```
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        Intent intent = new Intent(Cargando.this, Login.class);
        startActivity(intent);
        finish();
    }
},2000);
```

Código 3.1 Código encargado de contar los segundos de vida de la actividad Cargando.

3.3 Ventana de inicio de sesión

La actividad de Login es la encargada de mostrar al alumno la ventana de inicio de sesión de la aplicación. Esta ventana contiene una Vista de tipo ImageView y tres Vistas más, dos de tipo Text y una de tipo Button. Las Vistas de tipo Text son las encargadas de recibir la información necesaria introducida por parte del alumno. La primera Vista de tipo

Text tiene la leyenda “Matrícula” que es donde el alumno ingresara su matrícula. La segunda Vista también de tipo Text más específicamente un TextView para Password contiene la leyenda “Contraseña”, es donde el alumno introducirá su contraseña. La Vista de tipo Button es el botón encargado de dar inicio al proceso lógico interno de validación de usuario este botón tiene la leyenda “Iniciar sesión”. En la *Ilustración 10,* se puede observar el diseño de la interfaz para el inicio de sesión.

Ilustración 10 Ventana de inicio de sesión.

3.3.1 Funcionamiento de la actividad Login

El proceso lógico de validación de usuario está dividido en dos etapas: la etapa de validación de datos introducidos y la etapa de validación desde el Web Service.

La primera etapa, se encarga de hacer un chequeo de que los campos estén llenos o contenga información. En caso de que alguno de los campos este vacío o ambos estén vacios y el alumno presione el botón de Inicio de sesión, se lanzara una alerta como la que se muestra en la *Ilustración 11*. En caso contrario la validación es aceptada y se pasa a realizar la segunda etapa.

Ilustración 11 Alerta de mensaje "campos incompletos"

La segunda etapa, es la parte donde se realiza la petición al Web Service. La petición se realiza mediante el método POST y los parámetros que se envían son, la matricula y la contraseña del alumno. La respuesta del Web Service viene dada de un conjunto de datos con el formato de un objeto JSON. Dependiendo de la validación del inicio de sesión, puede haber una respuesta verdadera (true) o falsa (false). Cada caso se describe a continuación:

1. True: en caso de que el inicio de sesión sea válido, el Web Service entrega una respuesta donde se especifican datos personales del alumno, como: apellidos, nombre, email y el campo de mayor importancia "sesionValida" con el valor "true". Esto implica que se ha iniciado sesión correctamente.
2. False: Si el inicio de sesión no fue valido entonces la respuesta solo contendrá el campo de "sesionValida" como *"false"*, lo cual implica que no se pudo iniciar sesión. En este caso se mostrará una alerta de mensaje como la que se muestra en la *Ilustración 12*, donde se especifica que los datos (*matricula o contraseña*) son incorrectos.

Ilustración 12 Alerta de datos incorrectos.

3.4 Ventana de inicio

En la ventana de inicio el alumno podrá visualizar los avisos o notificaciones que pertenecen a su cuenta, en caso de que no exista contenido que mostrar, se le mostrara un aviso por defecto como el que se muestra en la *Ilustración 13*.

Ilustración 13 Ventana de inicio sin anuncios.

Si la cuenta del alumno tiene contenido que mostrar, este contenido se presentará en forma de páginas que podrán deslizarse hacia izquierda o hacia la derecha, donde un indicador de tipo *punto* ubicado en la parte inferior de la pantalla, mostrará cuántos avisos contiene y en cual está posicionado actualmente, tal como se muestra en la *Ilustración 14*. Los anuncios mostrados se moverán de forma animada, es decir, se estarán desplazando de manera automática tras cierto tiempo y estos podrán detenerse al mantener presionada el área del anuncio con el evento del touch.

Ilustración 14 Ventana de inicio con anuncios.

La ventana de inicio contiene una sección para presentar los avisos y está compuesta por tres elementos, dos de estos elementos son Vistas de tipo Text que contienen el título del aviso y el contenido del mismo respectivamente. El tercer elemento de esta sección es el indicador de páginas en forma de puntos ubicado en la parte inferior. Toda la sesión de avisos está delimitada entre el título de la ventana y la barra de navegación inferior.

3.4.1 Funcionamiento de la actividad Inicio

El funcionamiento lógico de la actividad Inicio depende de solo una sección de toda la ventana, esa sección contiene los campos de texto a mostrar, la detección de los eventos touch en la sección de avisos y los desplazamientos del indicador de avisos.

3.4.1.1 Anuncios dentro de una PageAdapter

Los avisos se obtienen como respuesta del Web Service, este recibe como parámetro la matricula. Un vez que se ha tenido respuesta del Web Service, se analiza si existe contenido o no. El contenido existente es mandado como parámetro a la clase *ViewPagerAdapter* que es la encarga de procesar los avisos, estos son ingresados a las Vistas tipo *Text*: título y cuerpo del aviso. En palabras menores la función de la clase es simular un arreglo de avisos y regresar una vista con todos los avisos para posteriormente ser mostrados en la ventana inicio. La actividad Inicio obtendrá la vista el arreglo de avisos y la función encarga de adaptar dicha vista a la sección de avisos de la ventana inicio es *setAdapter()*, tal como se muestra en el segmento de *código 3.2*.

```
1  private void setViewPagerItemsWithAdapter() {
2  viewpageradapter = new  ViewPagerAdapter(this,Titulos,Contenidos);
3  viewPager.setAdapter(viewpageradapter);
4  viewPager.setOnPageChangeListener(viewPagerPageChangeListener);   }
```

Código 3.2 Código para adaptar los anuncios en la sección de anuncios.

La línea número 2 es la parte encargada de asignar la vista con avisos a partir de la clase *ViewPagerAdapter* que recibe los parámetros: contexto, títulos (de los avisos) y contenidos (de los avisos).

La línea número 3 muestra cómo la vista que contiene los avisos es pasada como parámetro a la función *setAdapter()* para que esta muestre la vista de anuncios en la vista molde que se mostrarán en la ventana de inicio.

En la línea número 4 se agrega a la vista molde la función encargada de detectar los cambios de anuncios, en el segmento de *código 3.3*, se muestra la implementación de dicha funcion.

3.4.1.2 Detección de eventos touch en la sección de avisos

Se necesita detectar los toques del usuario en la sección de avisos dentro de la ventana, ya que permitirán realizar acciones de desplazamiento o detener de los avisos para que el usuario pueda leer el contenido de manera más detallada. Para detectar los desplazamientos hacia la izquierda o hacia la derecha, se hace uso de la propiedad *onPageChangeListener* de la clase *ViewPager* que se muestra en el segmento de *código 3.3*. Esta propiedad ofrece un método en específico para detectar la posición del aviso que se está mostrando en el momento. El *código 3.3* muestra el método *onPageSelected()* encargado de realizar dicha tarea. Cabe mencionar que este método solo captura la posición de la página o aviso, y esto ocurre cuando existe un desplazamiento hacia izquierda o derecha efectuado con el dedo del usuario sobre la pantalla, más específicamente en la sección de avisos.

La función *setCurrentItem()* se encarga de actualizar el nuevo aviso, en base a la posición capturada en el método *onPageSelected()* de la sección de *código 3.3*. Esta función recibe como parámetro el número de página que se mostrará y un valor booleano (true o false) para especificar si se desea una animación durante la transición del cambio de aviso.

```
ViewPager.OnPageChangeListener viewPagerPageChangeListener = new
ViewPager.OnPageChangeListener() {

    @Override
    public void onPageSelected(int position) {
        paginaActual = position;

        for (int i = 0; i < dotsCount; i++) {
            dots[i].setTextColor(getResources().getColor(android.R.color.darker_gray));
        }
        dots[position].setTextColor(getResources().getColor(android.R.color.background_dark));
    }

    @Override
    public void onPageScrolled(int arg0, float arg1, int arg2) {

    }

    @Override
    public void onPageScrollStateChanged(int arg0) {

    }
};
```

Código 3.3 Implementación de la propiedad OnPageChangeListener de la clase ViewPager.

La implementación de la función *SetCurrentItem* se muestra en el segmento de *código 3.4*.

```
final Handler handler = new Handler();

final Runnable Update = new Runnable() {
    public void run() {
        if (paginaActual == contadorPuntos) {
            paginaActual = 0;
        }
        viewPager.setCurrentItem(paginaActual++,true);
    }
```

Código 3.4 Implementación de la función setCurrentItem.

En el *código 3.4* dentro del método *run* se tiene una condición de tipo *if* la cual solo verifica si el aviso actual es el ultimo para reiniciar la variable *paginaActual* y crear el efecto de que los avisos vuelven a comenzar de nuevo.

Para detener con un touch un aviso y evitar que este continúe cambiando de manera automática se hace uso del método *setOnTouchListener()* de la clase *ViewPager*. Como se cuenta con una vista molde (sesión de avisos) de tipo *ViewPager* para mostrar los avisos, entonces se puede hacer uso del método *setOnTouchListener()* para detectar los eventos de touch en la sección de avisos. Este método se muestra en el segmento de *código 3.5*, el método a su vez contiene otro método llamado *onTouch* que detecta el evento de touch en la sección de avisos. Con la implementación de un *case* se determina si la pantalla es pulsada con la propiedad ACTION_DOWN o si ha dejado de ser presionada con la propiedad ACTION_UP.

La existencia de una variable booleana global permite detectar si la pantalla está en ACTION_DOWN o en ACTION_UP y con ello se puede detener el hilo encargado de ejecutar la función de actualización del aviso dándole al usuario la opción de detener el aviso para poder leerlo detenidamente.

```
viewPager.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                touch = false;
                break;
            case MotionEvent.ACTION_UP:
                touch = true;
                break;
            default:
                break;
        }
        return false;
    }
});
```

Código 3.5Funcion para detectar los eventos touch sobre la pantalla.

3.4.1.3 Desplazamiento del indicador de avisos

La actualización de los indicadores de página o aviso, se realiza en la sección de *código 3*.3 dentro del método *onPageSelected*, donde se captura la posición del aviso

actual. El funcionamiento consiste en dos partes, primero el ciclo *for* colorea todos los puntos del indicador de avisos de un solo color. Este ciclo realiza iteraciones dependiendo de la cantidad de avisos que existan. La línea de código después del *for* coloca un color distinto al anterior en el indicador de puntos dependiendo del aviso que se muestre en ese momento, por lo tanto el usuario podrá ver que el indicador de puntos cambia de color cuando se desplaza de un aviso a otro.

3.5 Ventana de consultas

Esta ventana ofrece al alumno poder visualizar tres opciones: Calificaciones, Carga de materias y Deudas. Estas opciones responden de igual manera que un elemento botón lo cual significa que llevaran a visualizar otra ventana al ser presionados. Los vistas implementadas para ofrecer los botones mencionados son de tipo *LinearLayout*, estos a su vez contiene dos elementos de tipo *Text* uno para la leyenda del botón y otro para el icono. Tal como se muestra en la *Ilustración 15*.

Ilustración 15 Ventana de consultas.

Para permitir que las Vistas *LinearLayout* respondan de igual manera que un botón al ser presionados, solo basta con agregar en la sección del *Layout* correspondiente en el archivo XML la propiedad *onClick*, seguido de un signo igual y entre comillas la actividad que se requiere abrir. Como se muestra en el ejemplo de *código 3.6*.

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:onClick="MyActivity">
</LinearLayout>
```

Código 3.6 Uso de la propiedad onClick en un LinearLayout.

3.5.1 Calificaciones

Esta sección le permitirá al alumno observar el listado de los programas que tiene para posteriormente ver la lista de calificaciones del programa seleccionado.

3.5.1.1 Ventana de programas para calificaciones

Esta ventana ofrece una lista de los programas que el alumno tiene en su cuenta. Cada programa es un *LinearLayout* diseñado de manera similar a los botones de la ventana de consultas descrita en la *sección 3.5*. En la *Ilustración 16* se muestra el diseño de la ventana de programas.

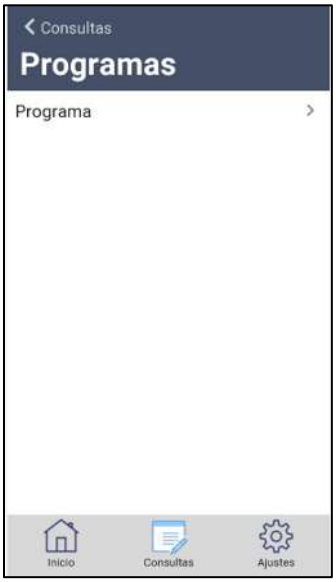

Ilustración 16 Diseño de la ventana de programas.

El funcionamiento lógico de la actividad *ProgramasCalif* consiste en hacer una petición al Web Service enviándole como parámetro la matricula del alumno. El Web

Service responderá con el objeto JSON donde contiene toda la información de los programas del alumno. Se obtienen los nombres de los programas y un valor denominado *matri* que es el identificador de cada programa. Finalmente por cada uno de los programas se creará el *LinearLayout* que funcionará como botón, en el cual se introduce el nombre del programa y el icono. Esto le permitirá al alumno no solo ver el programa que cursa actualmente, si existen programas previos o distintos, el alumno podrá visualizarlos todos permitiéndole presionar en el que desea ver las calificaciones.

3.5.1.2 Ventana Lista de Calificaciones

La ventana de calificaciones se muestra cuando se presiona alguno de los programas y le mostrará al alumno el total de créditos o años, el nombre del programa y el listado de las calificaciones por semestre o año. Está ventana está compuesta por una distribución de distintos *LinearLayouts* con orientación vertical y horizontal que a su vez contienen campos de tipo *Text* para mostrar los datos de semestre, programa, materias, calificaciones, etc. Además también contiene una Vista tipo *ScrollView* para poder mostrar todo el contenido de las materias al alumno. En la *Ilustración 17* se muestra el diseño de la ventana de calificaciones.

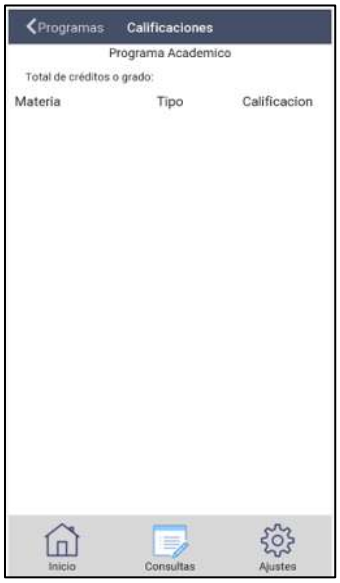

Ilustración 17 Diseño de la ventana de calificaciones.

El funcionamiento lógico de la Actividad ListaCalificaciones se basa en llenar los campos de texto con la información obtenida del Web Service. La petición al Web Service se realiza mandando como parámetro la *matri* que es el identificador de programa del que se requieren las calificaciones. La respuesta del Web Service contiene los semestres con las materias y sus respectivas calificaciones, esta información es procesada de tal manera que se obtiene cada dato para posteriormente llenar los campos de texto de la ventana de calificaciones. La ventana de calificaciones está dividida en dos secciones, una parte contiene campos de texto estáticos, donde se encuentran el nombre del programa académico y el total de créditos o grado, los cuales se llenan con la información que se recibe del Web Service. La otra sección es dinámica y se va creando dependiendo de la cantidad de semestres y de materias por semestre que tenga ese programa académico. En esta parte se muestra el ciclo escolar, fecha de inicio del ciclo escolar, el nombre de la materia, el tipo de calificación y la calificación numérica. Inicialmente se crean los campos de texto necesarios para el ciclo escolar y para la fecha de inicio. Después son introducidos dentro de un *LinearLayout* de tal manera que tome el formato correcto y finalmente este *LinearLayout* es introducido en otro más que funciona como molde y es el que se visualizará en la ventana. Para las calificaciones que se mostrarán por cada materia el proceso es similar, se crean los campos de texto para el nombre de la materia, el tipo de calificación y la calificación. Después son introducidos dentro de un *LinearLayout* de forma ordenada y finalmente este *LinearLayout* es introducido en el molde de la ventana. En pocas palabras se tiene un molde principal en el cual se agrega un ciclo escolar y su fecha de inicio, después sus respectivas materias con su tipo de calificación y la calificación numérica, todo esto de manera iterada hasta terminar con todos los ciclos escolares del programa educativo seleccionado. El *ScrollView* tiene su implementación únicamente en el área de los semestres y las calificaciones, es decir, la parte donde se muestra el programa académico y los créditos, estará fuera del *ScrollView* dando la apariencia de que estarán estáticos y solo se podrá desplazar verticalmente sobre las materias.

3.5.2 Cursos

En la sección de cursos se le ofrece al alumno visualizar la lista de ciclos escolares por programa para que posteriormente seleccione uno de los ciclos y este muestre una ventana con la información de las materias que pertenecen al ciclo.

3.5.2.1 Ventana de programas para cursos

Esta ventana tiene la misma funcionalidad que la venta de programas para calificaciones descrita en la *sección 3.5.1.1* y es activada cuando se presiona el botón cursos de la ventana consultas. El diseño de la ventana es el mismo que el diseño de la ventana de programas para calificaciones y se muestra en la *Ilustración 16*. Cabe mencionar que esta ventana de programas no es la misma que la de los programas de calificaciones, aunque tienen el mismo formato y muestran la misma información, al ser presionado alguno de estos programas se abre una ventana distinta.

3.5.2.2 Ventana de ciclos escolares

La ventana de los ciclos escolares se muestra después de presionar algún programa de la ventana de programas para cursos. Esta ventana muestra en una lista todos los ciclos escolares de la cuenta del alumno del programa. Los ciclos escolares son elementos de tipo *LinearLayout* y responden de igual manera que un botón. En la *Ilustración 18* se muestra el diseño de la ventana de ciclos escolares.

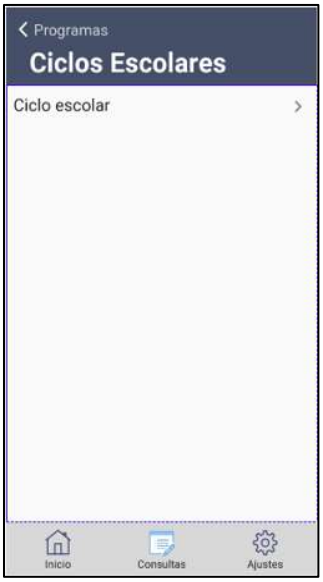

Ilustración 18 Diseño de la ventana de ciclos escolares.

El funcionamiento lógico de la actividad Ciclos se basa en tener una Vista principal (*contenedor de ciclos*) del tipo *LineaLayout* procedente del archivo XML de la actividad, la cual se usará para ingresar todos los ciclos de algún programa escolar.

Inicialmente se realiza la petición al Web Service mandándole como parámetro la *matri* (*identificador de programa escolar*) del programa seleccionado. En la respuesta del Web Service se reciben los ciclos escolares con sus respectivos cursos. Una vez que se tiene una respuesta correcta del Web Service, se pasa a crear un elemento *LinearLayout* que funcionará como *"botón"*, el cual contiene dos elementos de tipo *Text* donde se ingresa el nombre del programa y en otro el icono, tal como se muestra en la *Ilustración 18*.

El elemento *botón* se encuentra dentro del *LinearLayout* que funciona como *contenedor de ciclos* y finalmente el contenedor se encuentra dentro de un *ScrollView* para permitirle al usuario poder desplazarse en la lista de todos los ciclos escolares del programa. El nombre del ciclo escolar que se pondrá dentro del botón se obtiene del objeto JSON que devolverá el Web Service. El conjunto de cursos que están en el objeto JSON, son ingresados dentro de un *Array* de tipo *String*. Lo anterior con la finalidad de que cuando el alumno presione el botón de un ciclo escolar, se muestre una nueva actividad (*Ventana detalles de cursos*) donde se pasan como parámetros los cursos de dicho ciclo

para su uso dentro de la nueva actividad. Estos parámetros son enviados mediante el uso del método *putExtra()* en el momento que se activa la propiedad de captura de eventos llamado *OnClickListener*.

En el segmento de *código 3.7*, se muestra la implementación de la propiedad *OnClickListener* y dentro el método *onClick* donde se hace el paso de parámetros entre actividades.

```
View.OnClickListener CLICK = new View.OnClickListener() {
   @Override
   public void onClick(View v) {

                    Intent LC = new Intent(getApplicationContext(),DetalleCiclo.class);

      LC.putExtra("cursos",cursos[v.getId()]);
      LC.putExtra("matri",matri);
      LC.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
      startActivity(LC);
      finish();

   }
};
```

Código 3.7 Propiedad OnClickListener con paso de parámetros entre actividades.

3.5.2.3 Ventana detalles de cursos

En esta ventana el alumno puede visualizar los cursos en forma de lista que contiene el ciclo escolar, cada apartado de la lista muestra el nombre de la materia, clave, sección y grado.

En la *Ilustración 19* se muestra el diseño dentro de la venta de detalles de cursos.

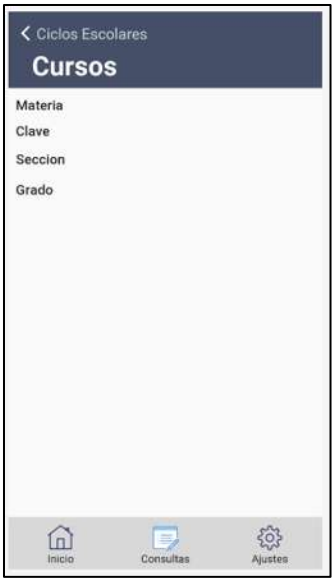

Ilustración 19 Diseño de la ventana de los detalles del ciclo escolar.

El funcionamiento lógico de esta actividad no realiza petición al Web Service, ya que los datos que se necesitan procesar para ser mostrados en la interface son recibidos mediante el paso de parámetros desde la actividad de los ciclos. Estos parámetros se reciben mediante el método *getIntent()* y el método *getStringExtra().* Tal como se muestra en el segmento de *código 3.8*.

```
String contenido = getIntent().getStringExtra(“cursos");
```

Código 3.8 Uso de los metodos getIntent() y getStringExtra().

El parámetro que recibe el método *getStringExtra(),* es el nombre del identificador de los datos enviados, este nombre tiene que coincidir con el nombre que se le da en la actividad desde donde se están enviando. Los datos recibidos son el conjunto de cursos de un ciclo escolar. El formato con el que se entregan los cursos es en un objeto JSON. Los cursos son separados en un arreglo de 4 posiciones y de tamaño igual a la cantidad de cursos, es decir por cada curso existen 4 datos. El arreglo de datos se pasa como parámetro a una clase. Esta clase se llama Adaptador_Cursos y hereda de la clase *BaseAdapter*. La clase se encarga de recibir el contexto de la aplicación y los datos de los cursos, para

posteriormente en su método *getView()*, procesar los datos y regresar una Vista con todos los cursos dentro del formato especificado en la *Ilustración 19*.

En la actividad *DetalleCiclo* se recibe dicha Vista y es adaptada a un *ListView* mediante el método *setAdapter()*, tal como se muestra en el segmento de *código 3.9*. Así el alumno podrá visualizar los cursos de cada ciclo en forma de una lista.

```
ListView lista_cursos.setAdapter(new Adaptador_Cursos(getApplicationContext(),infoCursos));
```

Código 3.9 Uso del metodo setAdapter().

3.5.3 Deudas

En la sección de deudas el alumno puede consultar si su cuenta tiene adeudos, de ser así el alumno puede seleccionar el adeudo que desea ver y en otra ventana se muestran los detalles de la deuda como son el título, el monto, fecha de generación, etc.

3.5.3.1 Ventana lista de deudas

En esta ventana el alumno puede visualizar el listado de las deudas que tiene su cuenta, si es que existen. Si en la cuenta del alumno no existen deudas, se muestra una alerta de mensaje, donde se notifica que la cuenta no presenta adeudos. Esta alerta se muestra en la *Ilustración 20*. El mensaje de alerta contiene un botón con la leyenda *entendido* que al ser presionado cerrará la ventana de deudas y abrirá la ventana de consultas.

Ilustración 20 Alerta de mensaje de no adeudos.

Las deudas son presentadas en Vistas de tipo *LinearLayout* de igual manera que los elementos mostrados en la ventana de consultas descrita en la *sección 3.5*. Estos elementos funcionan como botones y están integrados por dos elementos de tipo *Text*, el primer elemento contiene el título de la deuda mientras que el segundo elemento contiene el monto y un icono, tal como se muestra en la *Ilustración 21*.

En la parte lógica de la actividad Deudas se hace la petición al Web Service mandándole como parámetro la matricula. La repuesta del Web Service devolverá las deudas que presenta la cuenta del alumno. Si existen deudas en la cuenta, se procesará la información de la respuesta, para obtener la descripción de la deuda, el monto y el número de deuda. Durante el procesamiento de la información se crea un *LinearLayout* que funcionara como botón. También se crearan dos elementos de tipo *Text* los cuales tendrán la descripción de la deuda, el monto y el icono e irán dentro del botón, tal como se muestra en la *Ilustración 21*.

El botón se agrega dentro de un *LinearLayout* que funciona como contendor de botones (*botones de deudas*). El cual ya está definido dentro del archivo XML de la misma actividad. El evento del click en cualquiera de los botones de alguna deuda muestra la actividad de los detalles de la deuda, donde se envía como parámetro el número de deuda con el método *putExtra().*

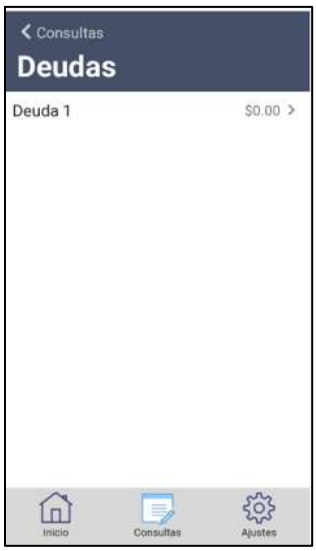

Ilustración 21 Diseño de la ventana de deudas.

3.5.3.2 Detalles de deuda

En esta ventana el alumno puede ver todos los detalles de la deuda, se muestra el número de deuda, título de la actividad, la dependencia, el programa educativo, la descripción de la deuda, ciclo escolar, fecha de generación, fecha límite y el monto. En la *Ilustración 22* se muestra el diseño de la ventana de los detalles de la deuda.

Ilustración 22 Diseño de la ventana detalles de la deuda.

El funcionamiento lógico de esta actividad consiste en capturar el número de deuda que recibió esta actividad desde la actividad de deudas, esto con el método *getIntent()* y el método *getStringExtra()*. Nuevamente se realiza la petición al Web Service el cual responde con las deudas y los detalles. Se procesa la información recibida en la respuesta y mediante condiciones lógicas se localiza la deuda con el número de deuda que se capturó anteriormente, después se obtienen los datos necesarios de la deuda para mostrar en la ventana de detalles. Para mostrar los detalles de la deuda se realiza la referencia a los elementos de tipo *Text* que se encuentran implementados en el diseño de la ventana en el archivo XML. Posteriormente se asigna el valor a cada elemento correspondiente con el método *setText()*.

3.6 Ventana de ajustes

Esta ventana le ofrece al alumno poder visualizar su información personal como su nombre y su matrícula, además un botón con el cual puede ver información acerca de la aplicación en otra ventana y un botón para cerrar la sesión. Los dos botones mencionados son elementos de tipo *LinearLayout* que están formados por otros elementos como *TextView* e *ImageView*. El botón con la leyenda *acerca de* solo abrirá otra ventana al ser presionado. El botón *cerrar sesión*, le permitirá al alumno hacer el cierre de sección tanto

en la aplicación como también en el Web Service. Al ser presionado mostrará la ventana de *Login* (*Inicio de sesión*). El diseño de la ventana de ajustes se muestra en la *Ilustración 23*.

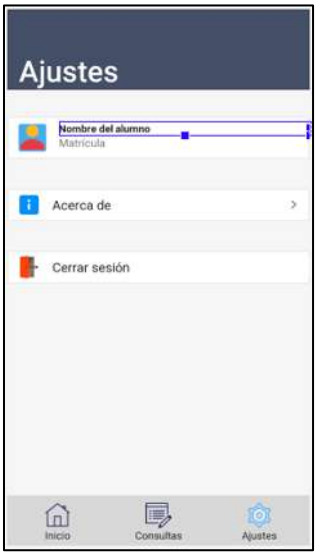

Ilustración 23 Diseño de la ventana de ajustes.

El funcionamiento del botón de cierre de sesión, inicia con una petición al Web Service sin envió de parámetros y simplemente se espera cerrar la sesión en el Web Service y salir a la ventana de inicio de sesión de la aplicación.

3.6.1 Ventana Acerca de

Esta ventana muestra el icono de la UMSNH, una leyenda con el título de *licencias* y una leyenda con los derechos de la aplicación. La leyenda *licencias* será responsiva al touch del alumno, es decir, al ser presionada abrirá una ventana. Este elemento es de tipo *TextView* pero con la propiedad *onClick* para que pueda responder de igual manera que un botón y así poder redirigir al alumno a una nueva ventana. El diseño de la ventana se muestra en la *Ilustración 24*.

Ilustración 24 Diseño de la ventana Acerca de.

3.6.1.1 Ventana licencias

Esta ventana le ofrecerá al alumno, poder visualizar las licencias de la aplicación y de las librerías implementadas en el desarrollo de tal, así como cualquier sitio web de donde se hayan obtenido herramientas como, imágenes, audios estilos, etc. En la *Ilustración 25* se muestra el diseño de esta ventana.

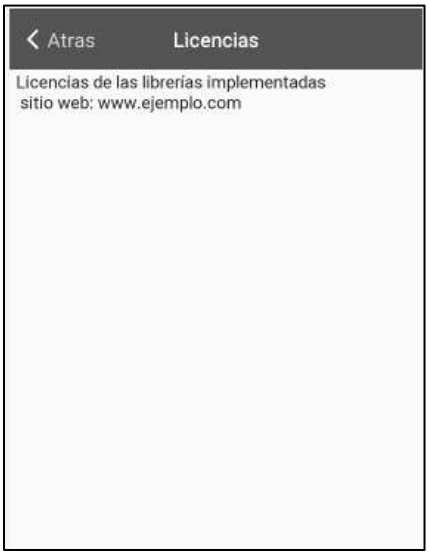

Ilustración 25 Diseño de la ventana de licencias.

Capítulo 4 Pruebas y resultados

Android Studio cuenta con distintos tipos de dispositivos virtuales para poder realizar las pruebas necesarias, para calificar el funcionamiento de la aplicación, estos dispositivos virtuales cuentan con distintas versiones de APIs, versiones del sistema operativo y distintos tamaños de pantallas. Por seguridad las pruebas finales de la ejecución de la aplicación fueron realizadas en distintos dispositivos reales. Ya que la aplicación está desarrollada con herramientas a partir de la API 23 : Android 6.0 (*Marshmallow*) hasta la API 28 : Android 9.0 (Pie) las pruebas fueron realizadas en un dispositivo Samsung Galaxy Grand Prime+ con S.O. de Android 6.0.1 con pantalla de 5'' y un dispositivo Samsung Galaxy J4 con S.O de Android 9 y una pantalla de 5.5''.

4.1 Resultados de la ejecución de la aplicación

Las pruebas realizadas se muestran con ilustraciones de ambos dispositivos y una breve descripción, las ilustraciones del dispositivo con S.O. Android 6.0.1 se muestran del lado izquierdo y el dispositivo con S.O. Android 9, se muestra del lado derecho.

4.1.1 Ventana de inicio de sesión

En la *Ilustración 26* se muestra la pantalla de inicio de sesión. Cuando se ingresen los datos (*matricula y contraseña*) del alumno de manera correcta y se presione el botón de color azul con la leyenda *Iniciar sesión* se avanzará a la siguiente pantalla.

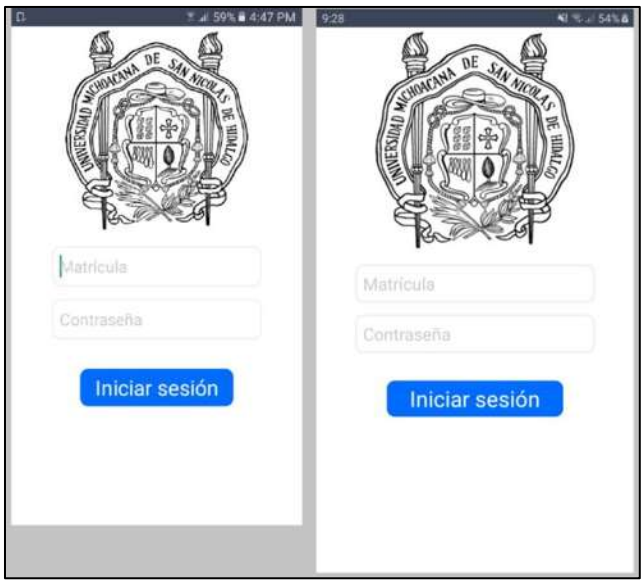

Ilustración 26 Pantalla de inicio de sesión.

4.1.2 Ventana de inicio

Después del inicio de sesión correcto por parte del alumno se muestra la ventana de anuncios. En la *Ilustración 27* se muestran los anuncios de la cuenta del alumno. Se puede notar que existen distintos avisos, y es por eso que en cada pantalla se presenta un aviso con titulo distinto.

Ilustración 27 Pantalla de inicio.

4.1.3 Ventana de consultas

La ventana de consultas se muestra en la *Ilustración 28* muestra al alumno las opciones actuales de consulta de calificaciones, cursos y deudas.

Ilustración 28 Pantalla de consultas.

4.1.4 Consulta de calificaciones

El proceso para consultar las calificaciones parte de la ventana de consultas. Enseguida el alumno presiona la opción de *Calificaciones* y se muestra la siguiente ventana que es la ventana de programas. En la *Ilustración 29* se muestran los programas en los que el alumno ha estado o está inscrito.

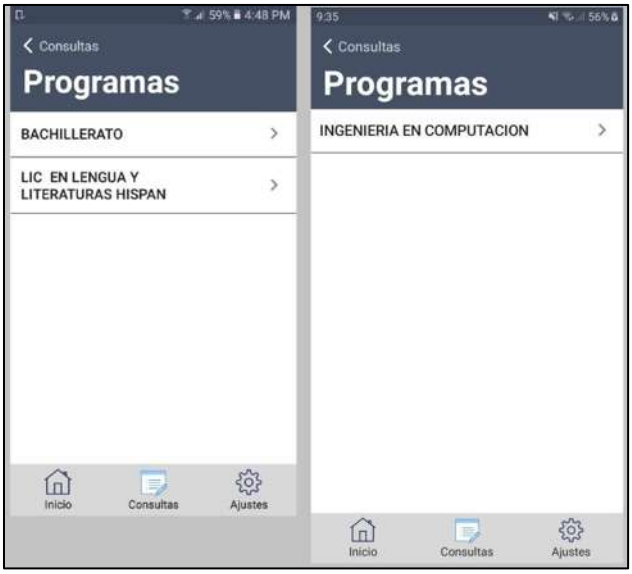

Ilustración 29 Pantalla de programas.

El alumno podrá presionar alguno de los programas mostrados y se generará una nueva ventana donde se mostrará toda la información referente a las calificaciones de ese programa. En la *Ilustración 30* se muestra un rango de calificaciones del programa seleccionado.

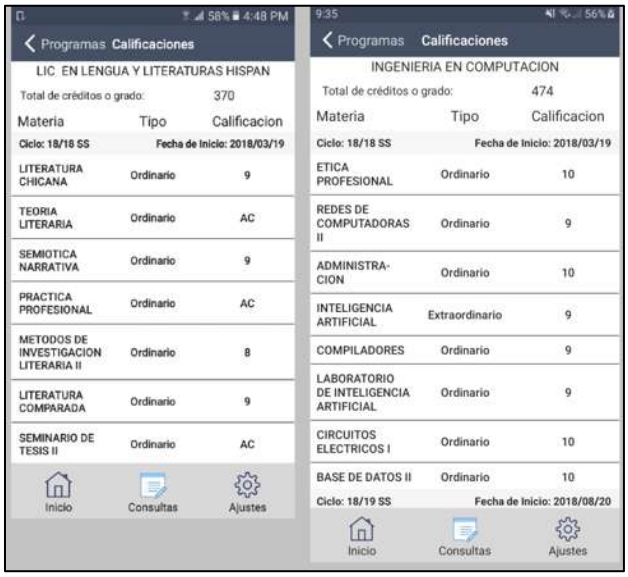

Ilustración 30 Pantalla de calificaciones.

4.1.5 Consulta de Kardex

El proceso para la consulta del Kardex consiste en que el alumno inicia de la ventana de consultas y selecciona la opción de *Cursos* y esto genera una nueva ventana con el mismo formato que la *Ilustración 29.*

Cuando el alumno presiona alguno de los programas se genera una nueva ventana, la cual muestra los ciclos escolares del programa presionado. En la *Ilustración 31* se muestran los ciclos escolares del programa seleccionado.

Ilustración 31 Pantalla de ciclos escolares.

Cuando el alumno presione alguno de los ciclos escolares nuevamente se abre una ventana en la cual se muestran los cursos que se llevaron en dicho ciclo escolar. En la *Ilustración 32* se muestra un segmento de algunos curos del ciclo seleccionado.

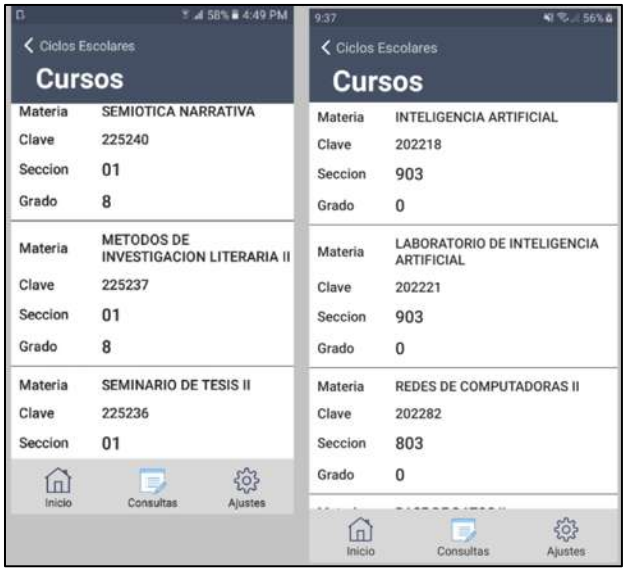

Ilustración 32 Pantalla de cursos.

4.1.6 Consulta de deudas

El proceso para la consulta de deudas comienza en la ventana de consultas, donde el alumno presiona la opción *Deudas* y esto genera una nueva ventana donde se muestran las deudas del alumno. En la *Ilustración 33* se muestran dos distintas ilustraciones de lado izquierdo una cuenta sin deudas y de lado derecho una cuenta con deudas.

Ilustración 33 Pantalla de deudas.

Cuando el alumno presiona alguna de las deudas mostradas, se genera una nueva ventana. En la *Ilustración 34* se muestran los detalles de la deuda seleccionada.

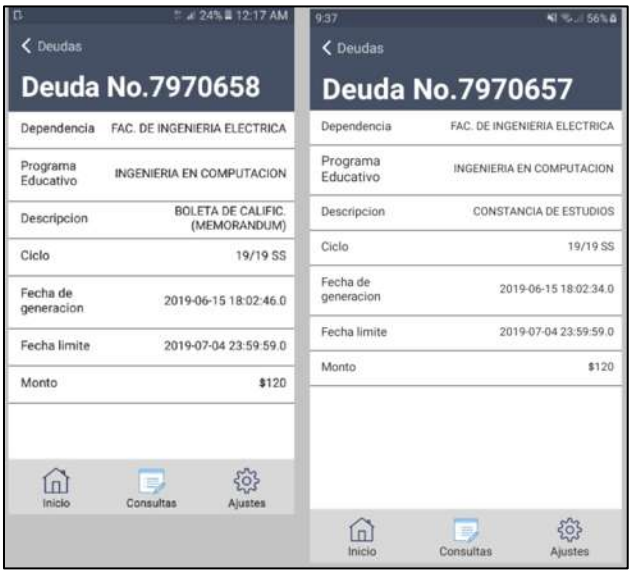

Ilustración 34 Pantalla de los detalles de la deuda.

4.1.7 Ventana de ajustes

La ventana de ajustes es la tercer pestaña principal de la aplicación, aquí el alumno ve sus datos personales como nombre y matricula, además un botón con la leyenda *Acerca de* para visualizar los datos de la aplicación y el botón para el cierre de sesión. En la *Ilustración 35* se muestra la ventana de ajustes.

Ilustración 35 Pantalla de ajustes

4.1.8 Ventana de licencias

El proceso para ver las licencias de la aplicación parte de la ventana de ajustes, donde el alumno cuenta con el botón *Acerca de*l cual al ser presionado abre una nueva ventana con el logotipo de la UMSNH y una leyenda con la palabra *Licencias*, tal como se muestra en la *Ilustración 36*.

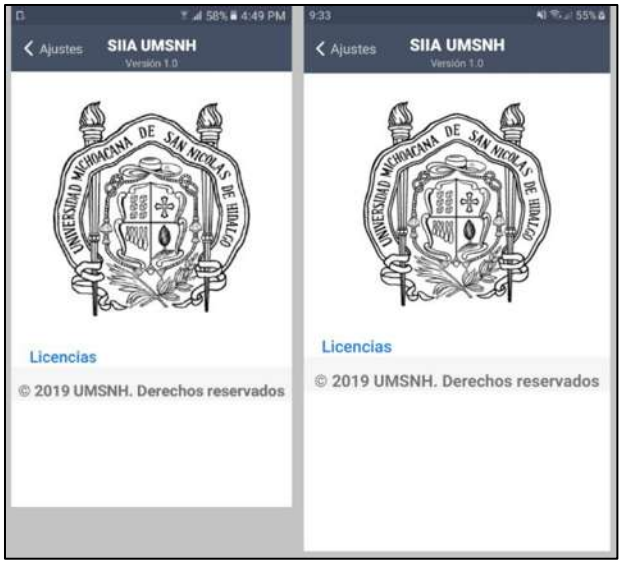

Ilustración 36 Pantalla acerca de la aplicación

El alumno podrá presionar la leyenda *Licencias* y esto genera una nueva ventana donde se muestran las licencias de las librerías externas al soporte de Android o java que se usaron en el desarrollo de la aplicación. En la *Ilustración 37* se muestra dicha ventana.

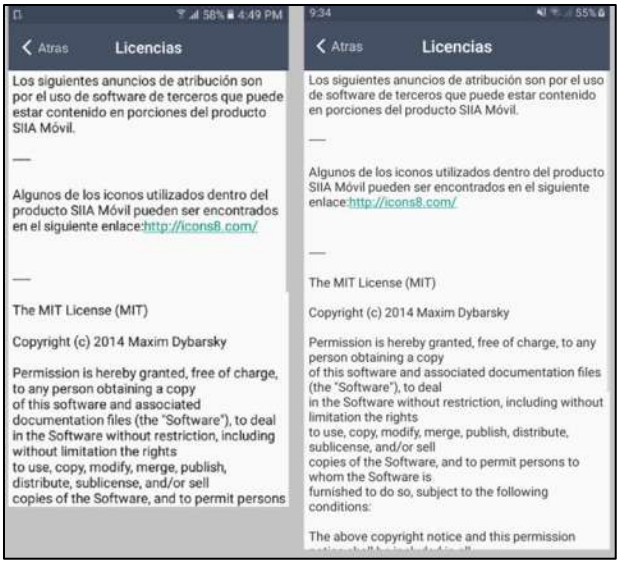

Ilustración 37 Pantalla de licencias

Capítulo 5 Conclusiones y trabajos futuros

5.1 Conclusiones

La aplicación móvil se desarrolló hasta la fase de pruebas y en base a los resultados y las satisfacciones del alumno, se puede determinar que ha cumplido con los objetivos especificados. Siendo así la herramienta necesaria que cumple con la comodidad que se esperaba y la sencillez en el manejo de esta, al momento de que los alumnos realizan sus consultas.

También se ha logrado que los alumnos prefieran tener la aplicación móvil, ya que el proceso que se realizaba mediante la aplicación web del SIIA era tardado en comparación al proceso mediante la App Móvil y esto implica un avance muy favorable tanto para los alumnos como para el SIIA pasando de un proceso tardado a un proceso más rápido.

5.2 Trabajos futuros

Como parte de los trabajos futuros que se podrían realizar para hacer esta una App más robusta y con más funcionalidades, se tienen contemplados los mismos puntos que se han considerado para el proyecto de desarrollo de la aplicación móvil para dispositivos iOS, que son los siguientes:

1. Desarrollo de un modulo para el llenado de los reportes del servicio social dentro de la aplicación y poderlos cargar al sistema.

2. Desarrollar un modulo para tener la opción de pagar los diversos conceptos con los que cuenta la universidad como son: credencial, memorándum de calificaciones, constancia, inscripciones, etc.

3. Desarrollar la funcionalidad para que los alumnos puedan llenar la encuesta de profesores dentro de la aplicación cuando esta esté disponible al término de cada semestre. Y que dentro de la encuesta se encuentre una opción para llenar todos los puntos a un nivel, cuando el profesor se esté desempeñando en ese rango. Ejemplo: que el profesor haya sido muy bueno y se quiera calificar todo con nivel satisfactorio.

4. Desarrollar un modulo para iniciar con los trámites de la titulación desde la aplicación.

5. Desarrollar la funcionalidad de búsqueda, para que los alumnos puedan encontrar una calificación de alguna materia de forma aun más rápida.

6. Publicar la aplicación en la tienda oficial Play Store para que todos los alumnos tengan acceso a ella de forma gratuita.

Bibliografía

[1] D. Escribano, «skyscanner,» 27 11 2018. [En línea]. Available: https://www.skyscanner.es/noticias/esta-es-la-historia-de-las-aplicaciones-moviles. [Último acceso: 20 03 2019].

[2] U. Veracruzana, «www.uv.mx,» [En línea]. Available: https://www.uv.mx/cdam/acercade/antecedentes/. [Último acceso: 25 03 2019].

[3] A. Ardións, «Android Studio Faqs,» 1 02 2016. [En línea]. Available: https://androidstudiofaqs.com/conceptos/android-studio-vs-eclipse. [Último acceso: 28 03 2019].

[4] A. Android, «www.academiaandroid.com,» 11 12 2014. [En línea]. Available: https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/. [Último acceso: 28 03 2019].

[5] A. Acosta Lopez, D. J. Manzano González, C. A. Martínez Morales Gonzalez y P. F. G. I. J., «Universidad Distrital Francisco José de Caldas,» 24 06 2014. [En línea]. Available: https://revistas.udistrital.edu.co/index.php/REDES/article/view/7117/8765.

[6] Developers, «developer.android.com,» [En línea]. Available: https://developer.android.com/studio/intro?hl=es-419. [Último acceso: 11 4 2019].

[7] U. d. Alicante, «http://www.jtech.ua.es,» 26 06 2014. [En línea]. Available: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html. [Último acceso: 12 05 2019].

[8] J. Revelo, «www.hermosaprogramacion.com,» 22 02 2015. [En línea]. Available: http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/. [Último acceso: 12 05 2019].

[9] J. Revelo, «www.hermosaprogramacion.com,» 24 01 2015. [En línea]. Available: http://www.hermosaprogramacion.com/2015/01/android-json-parsing/. [Último acceso: 12 05 2019].

[10] I. J. E. M. Villanueva, «Diagrama de bloques de la aplicacion.,» de *Portal de Alumnos SIIA Movil*, Morelia, Mich., 2019.