UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Aplicación móvil para la plataforma de recursos de la Editorial Universitaria Félix Varela

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Diosbel Enrique Lima Sánchez

Tutores:

Ing. Yenisel Díaz Llanes

Ing. Cesar Andrés González Rodríguez

Ing. Orlando Gabriel Toledano López

La Habana, julio de 2017

‘‘Año 59 de la Revolución’’

Declaración de Autoría

Declaración de Autoría

Declaro ser el autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

Diosbel E. Lima Sánchez

_____ _____

Firma del Tutor Firma del Tutor

Ing. Yenisel Díaz Llanes Ing. Cesar Andrés González Rodríguez

_____

Firma del Tutor

Ing. Orlando Grabiel Toledano López

Dedicatoria

El presente trabajo de diploma está dedicado a:

A mis padres: A mi madre especialmente quien siempre estuvo en los tiempos difíciles, y con quien tengo una deuda de por vida, por todo lo que ha hecho por mí. A mi padre, creador de los cimientos para forjar y concretar mi carrera universitaria.

A familiares: Para mis hermosas sobrinas, mi hermana y hermano. Mis primos, tíos y tías. A mis abuelas, especialmente para quien falleció hace un año, espero que dios la tenga en la gloria.

A cada uno de ellos les dedico este trabajo

Agradecimientos

Agradezco a todas aquellas personas que me apoyaron e hicieron posible la realización del presente trabajo de diploma.

Un agradecimiento especial a mis padres, mi abuela que en paz descanse y a toda mi familia en general que confiaron en mí y me guiaron por el camino correcto.

A los tutores: Yenisel, Cesar y Orlando que siempre estuvieron presente y atentos a cualquier duda que me surgiera en la confección de la tesis.

A todos mis compañeros del aula y de apartamento, especiales agradecimientos para ustedes colegas.

"Se cometen mucho menos errores usando datos incorrectos, que no empleando dato alguno"

Charles Babbage

Resumen

El creciente auge de la telefonía móvil en Cuba y el avance logrado en la informatización del país, ha hecho crecer el sector de desarrollo de *software* para teléfonos móviles, tanto es así que aplicaciones basadas en la web ahora necesitan tener soporte en estos dispositivos. En la Universidad de las Ciencias Informáticas, se está llevando a cabo el desarrollo de la plataforma de recursos para la Editorial Universitaria Félix Varela, la cual manejará todos los procesos que realiza dicha editorial. La plataforma permitirá el registro de los libros, así como sus secciones favoritas y recomendaciones a los clientes, además, se pretende que el usuario que interactúa con el sistema conozca la amplia gama de libros, y pueda, si el acceso lo permite descargar los de interés. La presente investigación tiene como objetivo desarrollar una aplicación móvil que permita a los usuarios de la editorial en desarrollo, conocer y solicitar los libros de su preferencia, además de guardar todos sus datos personales en el dispositivo, sin conexión de red. Para ello, se transitó siguiendo la metodología de desarrollo Mobile-D, usando el lenguaje de programación Java, sobre el entorno de desarrollo integrado Android Studio. Se realizaron los flujos de modelado, implementación y prueba indicados por la metodología. Obteniéndose finalmente una aplicación móvil para el sistema operativo Android, que permitió el uso de la plataforma de recursos en entornos de baja conectividad o de acceso poco frecuente a la red.

Palabras clave:

Aplicación móvil, Editorial Universitaria Félix Varela, teléfono móvil

Índice

CONTENIDO

Índice

Índice

Índice

Índice

Índice de ilustraciones

Ilustraciones

Índice de Tablas

Tablas

INTRODUCCIÓN

En la actualidad, las Tecnologías de la Información y las Comunicaciones (TIC), definen la utilización de múltiples medios tecnológicos o informáticos. Estas son útiles para almacenar, procesar y difundir todo tipo de información, visual, digital y con diferentes finalidades, como forma de gestionar, y organizar, ya sea en el mundo laboral o en el plano educativo (Soler Pérez, 2008).

Con el fin de incorporar las TIC en las distintas esferas de la sociedad, se han creado disímiles plataformas entre las que se encuentran las editoriales digitales. Estos son sistemas informáticos, que permiten publicar, promocionar y descargar contenidos del ámbito bibliográfico, tales como libros, revistas, documentos y noticias, mediante el uso de red de datos.

Una de las empresas que ha apostado por vincular su sistema a las editoriales digitales, es la Editorial Universitaria Félix Varela (EUFV), la cual es la encargada de la realización del proceso de edición de libros académicos que se corresponden con los programas de estudio de las carreras del Ministerio de Educación Superior (MES). Entre los servicios brindados por la editorial se encuentra la publicación y promoción de literatura académica y científica de una amplia variedad temática. Dichas publicaciones son de referencia para estudiantes y profesores cubanos e incluso de otras naciones, por lo que constituyen un vasto espacio de consulta y localización de libros y otros recursos educativos (Editorial Universitaria Félix Varela, 2013).

En el Centro de Tecnologías para la Formación (FORTES), de la Facultad 4, de la Universidad de las Ciencias Informáticas (UCI), es donde se encuentra en desarrollo la plataforma de recursos para la EUFV. La plataforma manejará los procesos de publicación de libros, autores, y de editoriales afiliadas a dicha institución.

Por otro lado, en Cuba han comenzado las etapas de prueba del nuevo sistema de comunicación para llevar internet a los hogares, así como el desarrollo de las zonas Wifi brindadas por la Empresa de Telecomunicaciones de Cuba (ETECSA).

Por tales motivos, el uso de dispositivos móviles en el país ha crecido considerablemente llegando a sobrepasar los 3 millones de usuarios (Juventud Rebelde, 2015). De acuerdo con (Cubadebate, 2016) *“En 2020 habrá 5 500 millones de usuarios de dispositivos móviles en el mundo, el 70% de la población mundial, con una media de 1,5 conexiones per cápita”.*

Introducción

La situación en América Latina no es muy diferente, el mercado de teléfonos móviles en la región es uno de los mayores del mundo y actualmente se estima que existen más de 200 millones de dispositivos inteligentes (Granados, 2015).

Esto ha generado el interés de la plataforma de recursos de la EUFV, decidiendo brindar su servicio para estos nuevos usuarios de la telefonía móvil. Para posibilitar esto, los clientes que acceden a la editorial necesitan tener acceso a la plataforma desde diferentes entornos, pero sucede que actualmente existen problemas de conectividad en el país, a pesar de que se está trabajando en aumentar los servicios de navegación, todavía no se ha podido propagar en todos los sectores de la isla.

Es por ende que la editorial necesita ofrecer su servicio con el menor uso de red posible, utilizando la conectividad solo para publicar o promover los libros que el usuario o cliente requiera. Además, debido a la portabilidad que ofrecen estos dispositivos inteligentes, se pretende que los usuarios tengan acceso privado a la plataforma, permitiendo guardar los cambios realizados en el sistema y acceder posteriormente sin conexión a la red.

A partir de la problemática anterior se define como problema a resolver: ¿Cómo ofrecer los servicios de la plataforma de recursos de la Editorial Universitaria Félix Varela a sus clientes en entornos de baja conectividad y de acceso poco frecuente a la red?

El **objeto de estudio** de la presente investigación es el desarrollo de aplicaciones para sistemas operativos móviles, enfocando el **campo de estudio** en aplicaciones móviles que utilizan servicios web para editoriales digitales en línea.

Teniendo en cuenta el problema a resolver se define como **objetivo general:** Desarrollar una aplicación móvil para la plataforma de recursos de la Editorial Universitaria Félix Varela, que permita buscar, guardar y visualizar libros en entornos de baja conectividad y de acceso poco frecuente a la red.

A partir del objetivo general se derivan los siguientes **objetivos específicos:**

- Definir el marco teórico de la investigación mediante el estudio y análisis de los principales estándares de desarrollo de aplicaciones para móviles.
- Definir la metodología, técnicas y herramientas para el desarrollo de la propuesta de solución.

- Desarrollar el análisis y diseño de la aplicación móvil para la plataforma de recursos de la Editorial Universitaria Félix Varela.
- Implementar las funcionalidades de la propuesta de solución.
- Realizar las pruebas de *software* al funcionamiento de la aplicación.

Se espera como **posibles resultados**:

1. Documento de tesis con todo lo referente al desarrollo de la aplicación móvil para la plataforma de recursos de la Editorial Universitaria Félix Varela.
2. Una aplicación móvil que permita la correcta interacción con la plataforma de recursos de la Editorial Universitaria Félix Varela en entornos de baja conectividad o acceso poco frecuente a la red.

Para dar cumplimiento a los objetivos propuestos, se definen las siguientes **Tareas**:

1. Analizar soluciones similares existentes.
2. Fundamentar en las tendencias actuales, tecnologías y conceptos relacionados al desarrollo de aplicaciones móviles.
3. Describir el uso de buenas prácticas, metodologías y estándares para el desarrollo de aplicaciones móviles vinculadas a plataformas, mediante servicios web.
4. Analizar las herramientas que se utilizan en el desarrollo de aplicaciones móviles y el uso de servicios web.
5. Analizar las tecnologías, herramientas y lenguajes de programación a utilizar en el desarrollo de las funcionalidades de la propuesta de solución.
6. Analizar el proceso de desarrollo de *software* a utilizar en la aplicación.
7. Definir la propuesta de solución teniendo en cuenta los elementos del diseño de una aplicación móvil.
8. Especificar los requisitos funcionales y no funcionales.
9. Implementar los requisitos descritos.
10. Ejecutar las pruebas funcionales del sistema.

Para el desarrollo de la investigación se emplearon los siguientes métodos de la investigación científica:

Teóricos:

- Histórico – Lógico: a través del cual se profundizó en los antecedentes y tendencias actuales de las editoriales digitales en aplicaciones móviles, así como los argumentos que antecedieron al problema.
- Analítico – Sintético: permitió procesar la información y llegar a conclusiones a partir de la revisión de conceptos, teorías, técnicas y herramientas.

Empíricos:

- Observación: se utilizó para obtener información mediante la percepción de objetos y recopilación de conocimientos para analizar la situación actual de sistemas similares a la solución propuesta. Se usa además para diagnosticar el problema e identificar posibles funcionalidades de la propuesta de solución.

El presente trabajo está conformado por tres capítulos, quedando de la siguiente manera:

Capítulo 1: Fundamentación Teórica.

Abarca los conceptos asociados al tema, necesarios para la comprensión de la solución del problema planteado. Se exponen los elementos teóricos utilizados en la investigación, se describen y seleccionan las tecnologías, metodología, herramientas y lenguajes de programación a utilizar en el desarrollo de la solución.

Capítulo 2: Propuesta de solución.

En este capítulo se exponen los elementos que permiten describir la propuesta de solución, tales como: modelo de dominio, requerimientos funcionales y no funcionales. Además, se hace una descripción textual de cada uno de ellos incluyendo los prototipos de interfaz de cada una de las historias de usuario para lograr un entendimiento claro del sistema a desarrollar.

Capítulo 3: Implementación y prueba.

Este capítulo está relacionado con la implementación, es decir se implementan todas las funcionalidades, y se obtiene un sistema con los requerimientos solicitados por el cliente. Una vez que concluye la implementación se describen las pruebas realizadas, para asegurar que cumple con las especificaciones requeridas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se abordan los conceptos relacionados con la investigación, que constituyen el conocimiento fundamentalmente teórico sobre los diferentes procesos y tecnologías asociadas al desarrollo de la solución requerida. Finalmente, se seleccionan y caracterizan las herramientas, metodología de desarrollo de *software* y los lenguajes de modelado y de programación a utilizar en el diseño e implementación de la propuesta de solución.

1.2. Conceptos asociados a la investigación

Resulta necesario conocer los conceptos relacionados a los dispositivos móviles, incluyendo los teléfonos inteligentes y los sistemas operativos que utilizan, así como de las editoriales digitales para facilitar la comprensión del objeto de estudio de la investigación.

1.2.1. Dispositivos móviles

Un dispositivo móvil se puede definir como un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras funciones más generales (Informajoven, 2006). Una gran cantidad de dispositivos electrónicos se clasifican actualmente como dispositivos móviles, desde teléfonos hasta *tablets*, pasando por dispositivos como lectores de RFID[1] (Pozo, 2015).

1.2.2. Teléfono inteligente

Un teléfono inteligente es capaz de realizar miles de tareas que puede hacer una computadora personal, y además brindar una mayor portabilidad. Estos dispositivos combinan lo que es un teléfono

[1] RFID (radio frequency identification) Identificación por radiofrecuencia.

celular estándar, con correo electrónico, navegación web, música, video, navegación GPS[2] y sensores que hacen de estos, dispositivos personales de gran utilidad (The Computer Language Company, 2015).

1.2.3. Sistema Operativo

Un Sistema Operativo (SO) es el programa que se ejecuta en un sistema de cómputo. Es el encargado de administrar el hardware de un sistema informático. Proporciona los mecanismos apropiados para asegurar el correcto funcionamiento del sistema informático e impedir que los programas de usuario interfieran con el apropiado funcionamiento del sistema. Cada computadora debe tener un SO capaz de ejecutar otros programas o aplicaciones (Vazquez, 2011).

1.2.4. Sistema Operativo móvil

Básicamente y al igual que en el concepto anterior es un programa que se encarga de manejar los procesos básicos de un dispositivo, pero en este caso solo para dispositivos móviles, permitiendo el uso de sus diferentes recursos tales como la cámara, el envío de mensajes y la recepción de llamadas. Es importante destacar que no todos los sistemas operativos son iguales, por lo que un programa que corre en un sistema operativo específico, probablemente no funcionará en otro (Alto Nivel, 2010).

1.2.5. Aplicación móvil

Una aplicación móvil es un *software* que se ejecuta en un sistema operativo móvil. Una aplicación de este tipo, permite realizar diversas acciones como enviar mensajes, realizar llamadas o chequear el correo electrónico. Las aplicaciones móviles pueden desarrollarse para cumplir un cierto objetivo y solo pueden ejecutarse en el sistema operativo para el que fueron programadas. Su instalación es básicamente sencilla, aunque pueden existir errores de compatibilidad con algunos dispositivos dependiendo del hardware (González, 2011).

[2] GPS (Global Positioning System) Sistema americano de navegación y localización mediante satélites.

1.2.6. Interfaz de programación de aplicaciones

Una Interfaz de programación de aplicaciones, comúnmente llamada API (siglas en inglés para *Aplication Programming Interface*) es un conjunto de especificaciones de comunicación entre componentes de *software*. Representa un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas, ya que no tiene que escribir códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa (ABC, 2015).

Desde hace algún tiempo es común encontrar APIs desarrolladas con el propósito específico de acceder a servicios web.

1.2.7. Editorial

Tradicionalmente, si se menciona editorial, se asocia esta expresión a libros, pero editorial, en estos tiempos modernos, va mucho más allá (Eguaras, 2014).

De acuerdo con el DRAE[3], es un artículo no firmado, que expresa la opinión de un medio de comunicación sobre un determinado asunto. Casas editoras. Perteneciente o relativo a editores o ediciones (DRAE, 2017). Esta definición, se relaciona con los contenidos, y los contenidos no se circunscriben exclusivamente a los libros y a lo impreso.

Un contenido, puede ser plasmado en distintos formatos y soportes, no solo en formato de libro y en soporte impreso (Eguaras, 2014).

1.2.8. Editorial digital

Como se mencionó anteriormente, el contenido de las editoriales puede ser plasmado en diferentes formatos, por lo que, con el avance de las TIC, y el ritmo acelerado de las novedades en el ámbito

[3] Diccionario de la Real Academia Española http://www.rae.es/

electrónico ha estimulado el crecimiento de editoriales digitales nativas, sellos pensados directamente desde la web, y que permiten a los usuarios descargar o acceder a recursos en forma de e-book[4] o pdf[5] (Kulesz, 2016).

1.3. Análisis de editoriales digitales con aplicaciones móviles

A nivel internacional, se pueden relacionar diversos sistemas de editoriales digitales con aplicaciones para dispositivos móviles, entre los que se destacan: la editorial de Amazon bajo el nombre de Kindle, Kobo, iBooks, y por último Play Books. A nivel nacional no fueron encontrados registros, de editoriales digitales para versiones móviles.

A continuación, se realiza un análisis de algunos de estos sistemas.

1.3.1. Sistemas similares a nivel internacional

***Amazon Kindle app*:** La aplicación de *Kindle* tiene una ventaja y es que puedes comenzar a leer en un dispositivo y continuar desde cualquier otro donde esté instalada esta app, con conexión a internet. Desde la biblioteca se puede ver la colección de libros en forma de íconos o de lista, los libros que se encuentran descargados en el dispositivo y los que se encuentran en la nube. Además, se pueden ordenar por “Reciente”, “Título” y “Autor”. La aplicación de Kindle está disponible para dispositivos con Android, iOS, *Windows*, Mac, Linux y prácticamente en cualquier plataforma con conexión (Vázquez, 2016).

Kobo: Es un lector de libros electrónicos producido en Toronto por *Kobo Inc*. Tiene la colección de *e-books* que compite directamente con Amazon, y es una de las mejores librerías actualmente. (Vázquez, 2016). Su aplicación para dispositivos móviles posee un sinfín de posibilidades, desde vincular los libros descargados desde la aplicación hasta sincronizar el punto de lectura, permite realizar marcadores,

[4] E-Book (Electronic Book) versión o evolución electrónica o digital de un libro.

[5] PDF (por sus siglas en inglés, Portable Document Format) es un formato de almacenamiento para documentos digitales independiente de plataformas de *software* o *hardware*.

notas y partes resaltadas. Además, se puede personalizar la forma de leer y hacer un seguimiento de cuánto se tarda en leer un libro (Marquina, 2015).

iBooks: Muy buena aplicación desarrollada por *Apple Inc.* En esta app los libros pueden integrar fotos, vídeos y animaciones interactivas. La novedad de este sistema de libros de texto electrónicos es la memorización a través de tarjetas, con términos por un lado y su definición por el otro. Las tarjetas cuentan con su propia interfaz permitiendo desplazarse por ellas, e incluso se pueden desordenar de forma aleatoria (Vázquez, 2016).

Google Play Books: Es la aplicación de Google para leer eBooks y aunque es de la más recientes, ha mejorado mucho desde su lanzamiento; ahora es mucho más visual y con funciones muy útiles. Se pueden leer ePubs descargados desde internet, pero se deben agregar desde la versión web, no desde la versión móvil (Vázquez, 2015).

El estudio de estos sistemas aportó conocimientos básicos en el desarrollo de aplicaciones móviles para editoriales digitales en línea. La investigación de cada una de las características que presentan dispuso utilizar algunas de las ventajas que poseen. De *Google Play Books* se toma en cuenta la interfaz, ya que presenta un diseño fluido e intuitivo, y de Amazon Kindle, iBooks y Kobo la organización y el manejo de los procesos de libros y autores.

1.4. Tipos de aplicaciones móviles

Para el desarrollo de la propuesta de solución se procede a realizar un estudio de las diferentes aplicaciones móviles con el objetivo de seleccionar una de ellas, dependiendo de las características y ventajas que aportan al sistema a desarrollar.

1.4.1. Aplicaciones nativas

Las aplicaciones nativas son las que se desarrollan específicamente para cada sistema operativo móvil, lo cual hace que su funcionamiento sea fluido y estable para el sistema operativo que fue creada (Pimienta, 2014).

Ventajas:

- Utilización de los recursos tantos del sistema como del hardware.
- Permite ser publicada en tiendas para su distribución.
- En su mayoría, no necesitan estar conectadas a Internet para su funcionamiento.

Desventajas:

- Solo pueden ser utilizadas por un dispositivo que cuente con el sistema para el cual fue desarrollada.
- Requiere de un costo para distribuirla en una tienda, y dependiendo el sistema, para el uso del entorno de desarrollo.
- Necesitan aprobación para ser publicadas en la plataforma.

1.4.2. Aplicaciones web

Son aquellas desarrolladas usando lenguajes para el desarrollo web como lo son html, css y *JavaScript* y un framework para el desarrollo de aplicaciones web, como por ejemplo *jquery mobile*, *Sencha*, y *Kendo UI*. Se podría decir que este tipo de aplicaciones es muy usada para brindar accesibilidad a la información desde cualquier dispositivo, ya que solo se necesita contar con un navegador para acceder a esta (Pimienta, 2014).

Ventajas

- Pueden ser utilizadas desde cualquier dispositivo sin importar el sistema operativo.
- Puede que requiera un coste para su desarrollo, pero este puede ser mínimo en comparación con las nativas.
- No requieren de ninguna aprobación para su publicación.

Desventajas:

- No pueden ser publicadas en plataformas para su distribución.
- No utilizan los recursos del sistema ni del dispositivo de manera óptima.

1.4.3. Aplicaciones híbridas

Por último, las aplicaciones híbridas, como su nombre lo indica tienen un poco de cada tipo, de las aplicaciones ya nombradas. Este tipo de aplicaciones se desarrolla utilizando lenguajes de desarrollo web y un *framework* dedicado para su creación, como por ejemplo *phonegap*, *titanium appacelerator*, y *Steroids*. La facilidad que brinda este tipo, es que no hay un entorno especifico el cual hay que utilizar para su desarrollo y la mayoría de las herramientas son de uso gratuito, también pudiendo integrarlo con las de aplicaciones nativas (Pimienta, 2014).

Ventajas:

- Uso de los recursos del dispositivo y del sistema operativo.
- El costo de desarrollo puede ser menor que el de una nativa.
- Son multiplataforma.
- Permite distribución a través de las tiendas de su respectiva plataforma.

Desventajas:

- La documentación puede ser un poco escasa y desordenada.

1.4.4. Selección del tipo de aplicación a utilizar

Se desarrollará una aplicación móvil nativa, debido a las ventajas que esta aporta. Están diseñadas específicamente para sacar el mayor provecho a los recursos del sistema, los tiempos de respuesta suelen ser menores. Además, al no tener que usar un navegador web, los datos del usuario pueden ser almacenados indefinidamente. En este caso es correcto usar este tipo de aplicación puesto que se ajusta mejor a las dimensiones y características del dispositivo, posibilitando una óptima interacción entre el usuario y la aplicación.

1.5. Sistemas operativos móviles

A continuación, se resume brevemente algunas de las características de los sistemas operativos móviles más usados en la actualidad. Siendo importante, conocer las características e información de estos, para permitir una correcta selección y evaluar si se ajusta a las necesidades de la solución.

1.5.1. Sistema Operativo móvil Android

Es un SO basado en Linux, liberado bajo la licencia de código abierto GNU/GPL[6], que provee una capa de librerías escritas en C y C++, y un marco de trabajo para el desarrollo de aplicaciones. Actualmente es uno de los más usados, incluyendo en el mercado internacional el rating de ventas en todos los dispositivos. Originalmente este marco de trabajo estaba solo basado en el lenguaje Java y posteriormente se liberó para aplicaciones nativas en C, aunque no es recomendable a menos que se necesite utilizar de manera excesiva al microprocesador. Incluye además una suite de aplicaciones iniciales, por ejemplo, la aplicación que manejan los contactos, posibilita también la instalación de otras aplicaciones que pueden ser descargadas desde internet. Está pensado para instalarse solamente en dispositivos móviles (TechTarget, 2015).

1.5.2. Sistema operativo móvil IOS

Sistema operativo desarrollado por la compañía *Apple Inc.*, exclusivamente para sus dispositivos. Originalmente fue creado para el teléfono *iPhone*, pero luego empleado en otros dispositivos como el iPod, iPad, iPad Mini y la segunda generación de *Apple TV*. iOS posee aplicaciones preinstaladas como *Mail*, Teléfono, Calendario, *Safari,* Música, Videos, Mensajes, Notas, Mapas, Calendario, *FaceTime*, y *Game Center*; aunque esto varía dependiendo del dispositivo. También posee una tienda exclusiva de aplicaciones llamada App Store. (Alegsa, 2016).

1.5.3. Sistema operativo móvil *Windows Phone*

Es el sistema operativo desarrollado por *Microsoft Inc.* posee una interfaz sencilla. Además, cuenta con una integración total con *Windows*, por lo que sincronizar el servicio de correo, contactos, *OneDrive*, y demás programas de Microsoft es casi ideal. La desventaja de este sistema es el número de aplicaciones con las que cuenta *Windows Phone*. No alcanza el número de aplicaciones que se pueden disfrutar en Android o iOS, aunque cada día se incorporan nuevas (Sánchez, 2015).

[6] GNU GPL (General Public License) licencia de derecho de autor más ampliamente usada en el mundo del *software* libre y código abierto.

1.5.4. Selección del sistema operativo móvil

Para la realización de la propuesta de solución, se ha decidido desarrollar la aplicación para el sistema operativo móvil Android, puesto que es uno de los más usados en la actualidad. Además, es un sistema robusto que cuenta con miles de desarrolladores y aplicaciones. Al estar desarrollado sobre el kernel de Linux, adopta la licencia GNU/GPL por lo que defiende la política enmarcada en Cuba, aprobando la utilización de *software* libre en todas las instituciones del país.

1.6. Análisis del sistema operativo Android

Después de haber seleccionado el sistema operativo Android como plataforma en la que se ejecutará la aplicación, es importante conocer los detalles referentes a este, incluyendo características y arquitectura, esta información sentará las bases el posterior desarrollo de la propuesta de solución.

1.6.1. Características y especificaciones

Las características principales del sistema operativo Android se describen con detalles en la siguiente tabla:

Tabla 1 Características y especificaciones de Android **(Molina Rivera, et al., 2012)**

Aplicación	**Características**
Diseño de dispositivo	Es adaptable a pantallas más grandes, *VGA*, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la *OpenGL* ES 2.0 y diseño de teléfonos tradicionales.
Almacenamiento	*SQLite*, como gestor de base de datos, base de datos liviana, usada para propósitos de almacenamiento de datos.
Conectividad	Las tecnologías de conectividad soportadas son: GSM/EDGE, IDEN, CDMA, EVDO, UMTS, Bluetooth, Wi-Fi, LTE y WiMAX.

Navegador web	El navegador web incluido está basado en el motor de renderizado de código abierto *WebKit*, emparejado con el motor *JavaScript* V8 de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test Acid3.
Soporte de Java	Las aplicaciones están escritas en Java, pero no se ejecutan desde esta máquina virtual, sino que primero se compila en un ejecutable *Dalvik* y se corre en la Máquina Virtual *Dalvik*. *Dalvik,* es una máquina virtual especializada, y diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP *Runner*.
Soporte multimedia	Se soportan los siguientes formatos multimedia: *WebM*, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HEAAC (en contenedores MP4 o 3GP), MP3, MIDI, *OggVorbis*, WAV, JPEG, PNG, GIF y BMP.
Soporte para *streaming*	*Streaming* RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video>). *Adobe Flash Streaming* (RTMP) es soportado mediante el Adobe Flash Player. *Adobe Flash* HTTP *DynamicStreaming* estará disponible mediante una actualización de Adobe Flash Player.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del *software*.
Multi-táctil	Soporte nativo para pantallas multi-táctiles.
Multitarea	Multitarea real de aplicaciones disponibles, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada.

Tethering	Por último, soporta *tethering*, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico (todos los teléfonos desde la versión 2.2, no oficial en teléfonos con versión 1.6 o superiores mediante aplicaciones disponibles en el *Android Market*, por ejemplo, *PdaNet*). Para permitir a un PC usar la conexión 3G del móvil se podría requerir la instalación de *software* adicional.

1.6.2. Arquitectura del SO Android

A continuación, se detallan los principales componentes de la arquitectura Android (Android Developers, 2016):

- **Núcleo de Linux:** el sistema depende de un núcleo de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de *drivers*. Esta capa del modelo actúa como capa de abstracción entre el *hardware* y el resto de la pila.
- **Librerías nativas:** incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto.
- ***Runtime* de Android:** posee un núcleo de librerías para la Máquina Virtual *Dalvik* (*VDK* por sus siglas en inglés) que se encarga de ejecutar las aplicaciones. *Dalvik* es optimizado para dispositivos móviles, por lo que consume menos memoria y provee rapidez de ejecución. A partir de Android 5.0 se reemplaza *Dalvik* por ART (*Android Runtime*). Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.
- **Marco de trabajo o entorno de aplicación:** los desarrolladores tienen acceso completo a las APIs del marco de trabajo usado por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes.
- **Aplicaciones:** Haciendo uso del marco de trabajo se encuentran las aplicaciones. Todas las aplicaciones, como la de contactos, los juegos o el correo usan el marco de trabajo de Android que a su vez usa el *runtime* y las librerías.

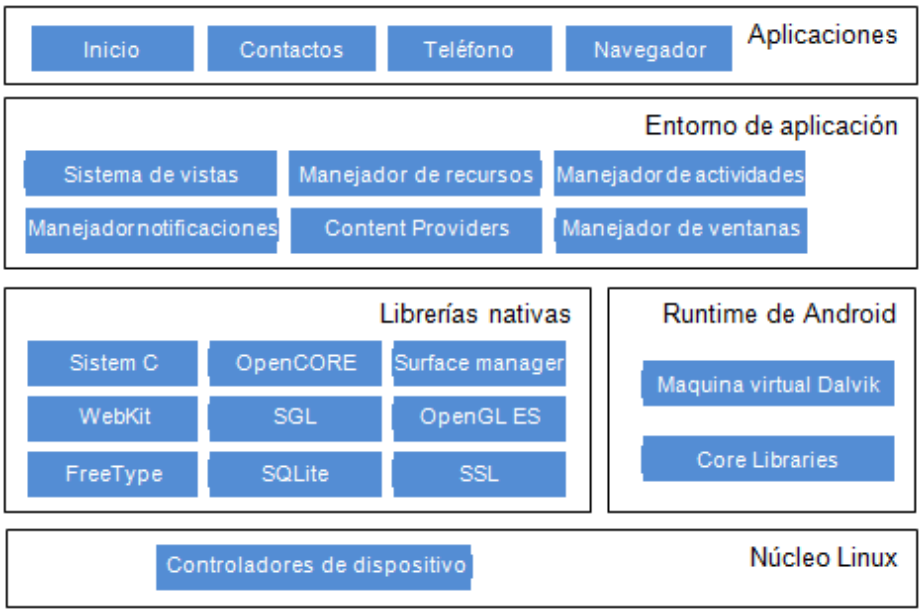


Ilustración 1 Arquitectura del sistema operativo Android **(Android Developers, 2016)**

1.6.3. Versiones del sistema operativo

El siguiente estudio permitió seleccionar las versiones del sistema operativo, en las que la propuesta de solución a desarrollar, debe tener soporte para ser ejecutada.

Tabla 2 Versiones de Android[7] **(Android Developers, 2016)**

Versión	Nombre	API	Distribución
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.2%

[7] Datos recopilados durante un período de 7 días hasta el 1 de agosto de 2016. No se muestran versiones con una distribución inferior al 0,1%.

4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Beam	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%

Se decidió dar soporte a las versiones a partir de 4.0 en adelante, tomando como mínimo de porciento de distribución global (X >= 1.5%).

1.6.4. Paquete de desarrollo

El Paquete de Desarrollo de *Software* de Android (SDK, por sus siglas en inglés *Software Development Kit*), proporciona las bibliotecas de API y las herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones para Android (Android Developers, 2016).

Cada vez que Google libera una nueva versión de Android también es liberada la SDK correspondiente. Para permitir el desarrollo de aplicaciones con las últimas características, los desarrolladores deben obtener e instalar cada versión de la SDK para cada dispositivo en particular.

La plataforma de desarrollo se encuentra para los sistemas operativos *Windows* (a partir de XP), Linux y Mac OS X (a partir de la versión 10.4.9). Cada componente puede descargarse de forma individual y añadir librerías para facilitar el desarrollo (Android Developers, 2016).

1.7. Análisis de los servicios web

La W3C[8] define "Servicio web" como un sistema de *software* diseñado para permitir interoperabilidad[9] máquina a máquina, y el cual puede ser ofrecido a través de la web. En general, los servicios web son APIs Web que pueden ser accedidas en una red, como internet, y ejecutadas en un sistema de servidor remoto (W3C, 2000).

Se puede definir de manera sencilla como un conjunto de tecnologías de estándares de *software* para el intercambio de datos entre aplicaciones. Estos pueden ser desarrollados en una gran variedad de lenguajes para ser implementados sobre muchos tipos de redes de computadores.

El conjunto de servicios y protocolos de los servicios web es conocido comúnmente como "*Web Services Protocol Stack*" y básicamente son utilizados para definir, localizar, implementar y hacer que un servicio web interactúe con otro. Este conjunto está conformado esencialmente de cuatro subconjuntos (W3C, 2000):

- Servicio de transporte
- Mensajería XML
- Descripción del servicio
- Descubrimiento de Servicios

1.7.1. Arquitectura orientada a servicios

Las Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) ha emergido como un paradigma de desarrollo de *software*. OASIS, en su Modelo de Referencia para SOA, la define como un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios o entidades (personas u organizaciones) (OASIS, 2006).

[8] W3C son las siglas de World Wide Web Consortium, consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución.

[9] Habilidad de dos o más sistemas para intercambiar información y utilizar la información intercambiada.

El funcionamiento de SOA, consiste en un mensaje creado en XML que posee tres partes:

- Una etiqueta conocida como <*Envelope*> la cual define un *framework* para describir el contenido del mensaje y sus instrucciones de proceso, esto mediante los *header* que son los que contienen control de información como los atributos de calidad de servicio y el *body* que contiene la identificación del mensaje y sus parámetros.
- Un conjunto de reglas de codificación para expresar instancias de los tipos de datos definidos en la aplicación
- Una convención que sirve para representar los llamados y respuestas a procedimientos remotos.

Los mensajes SOA presentan la dificultad de ser fundamentalmente de una sola vía de transmisión entre el que envía y el que recibe (W3C, 2000)

1.7.2. REST[10]

Es un estilo de arquitectura basado en un conjunto de principios que describen cómo se definen y se abordan los recursos en una red (Barry, 2016). Esta arquitectura, se centra en la solicitud de recursos y utiliza los principios básicos de la aplicación web (IBM, 2016):

Transporte de los datos mediante HTTP por medio de las operaciones básicas siguientes:

- Petición *GET*, el recurso se solicita a través de la URL al servidor web.
- Petición *POST*, el recurso se solicita mediante un conjunto de datos.
- Petición *PUT*, envía el recurso identificado en la URL desde el cliente hacia el servidor.
- Petición *DELETE*, solicita al servidor que borre el recurso identificado con el URL.

Los diferentes servicios son invocados mediante URI[11].

[10] REST (Representational State Transfer) Tranferencia de estado representacional.

[11] (Unified Resource Identifier, por sus siglas en inglés), es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

La codificación de datos es identificada mediante tipos MIME[12] (*text/html*, *image/gif*).

REST es una arquitectura simple, que tiene buenos tiempos de respuesta entre el cliente y el servidor, presenta gran estabilidad frente a los cambios, además de sencillez en su desarrollo para clientes; pero su inconveniente es que no se mantiene el estado por lo tanto cuando el servidor trata una solicitud lo hace de forma independiente sin recordar solicitudes anteriores.

1.7.3. Selección de la arquitectura de servicios web

Conforme a los servicios web presentados anteriormente, se selecciona REST como arquitectura de servicios web para esta elección se tuvo en cuenta que. SOA ha sido bastante difundido, pero no es adecuada su utilización en el sistema operativo Android, puesto que su complejidad hace que tenga un rendimiento menor en comparación con REST. Además, el sistema operativo seleccionado no posee librerías nativas para trabajar con SOA, mientras que si las tiene para REST.

1.8. Metodologías, tecnologías y herramientas a utilizar

El desarrollo de un *software* es un proceso complejo, en el cual se tienen muchos elementos que permiten su construcción con calidad. Para la implementación de una aplicación, es necesario tener en cuenta diversos aspectos de gran importancia tales como: la metodología, el lenguaje de programación, las herramientas de desarrollo y modelado a emplear, pues de esto dependerá totalmente el éxito en el desarrollo de la misma.

1.8.1. Metodologías de desarrollo de *software*

Una metodología de desarrollo de *software* es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de *software* en sus esfuerzos por implementar nuevos sistemas de información. Está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas

[12] (Multipurpose Internet Mail Extensions), es un estándar de internet que permite el intercambio de diferentes tipos de archivos en la web.

en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Fitzgerald, 2006).

Entre las metodologías de desarrollo de software

Metodologías tradicionales

Las metodologías tradicionales o robustas, están centradas específicamente en el control del proceso del proyecto, haciendo mayor énfasis en la planificación y especificación precisa de requisitos y modelado (Roberth G. Figueroa, 2016). Estas buscan imponer disciplina, estructura, orden y consistencia en el proyecto de desarrollo de *software*. Se basan fundamentalmente en la planificación total, extensa documentación, y una definición rigurosa de actividades, artefactos y roles.

Características:

- Documentación exhaustiva de todo el proyecto.
- El plan de proyecto se define en la fase inicial del desarrollo.
- Los costos son altos al implementar un cambio.
- No es una buena solución para entornos volátiles.
- El equipo de desarrollo debe ser grande.

Metodologías ágiles

Esta metodología nace como respuesta a los problemas de las metodologías tradicionales, basándose en dos aspectos puntuales, retrasar las decisiones y permitir una planificación adaptativa; permitiendo potenciar el desarrollo de *software* (Roberth G. Figueroa, 2016).

Características:

- Buena solución para proyectos a corto plazo.
- Facilidad de respuesta a cambios repentinos en el desarrollo.
- Equipos de desarrollo pequeños.
- Ciclos de desarrollo cortos.

1.8.2. Características que debe cumplir una metodología para el desarrollo móvil

En esta sección se detallan algunas de las características esenciales que deben proporcionar los métodos de desarrollo a analizar, con el fin de emplearse eficientemente para el desarrollo de *software* móvil (Rahimian, y otros, 2008).

- **Agilidad:** flexibilidad en el desarrollo y la productividad, proporcionando métodos que se adapten a los cambios y que aprenden de la experiencia. Las características específicas de los entornos móviles deben ser: procesos iterativos e incrementales, desarrollo conducido por *tests*, procesos adaptativos, hablar con el cliente continuamente, desarrolladores altamente cualificados, asegurar la calidad, revisiones continuas del proceso y prioridad de los requerimientos (Rahimian, y otros, 2008).
- **Conciencia de mercado:** el mercado actual está orientado hacia los productos *software* por lo que un proceso de desarrollo móvil debería enfocarse al desarrollo del producto y no del proyecto. Antes los procesos estaban orientados a las actividades técnicas ahora tienen que hacer un balance entre las actividades orientadas a mercado y las técnicas (Rahimian, y otros, 2008).
- **Soporte a toda la línea de producción:** el ciclo de vida de un *software* móvil es bastante corto, por lo que los desarrolladores tienden a desarrollar una familia de productos de *software* móviles en un solo intento y así reducir los costes. Se debe tener en cuenta la importancia de la línea de producción para la mejora de la calidad del *software* (Rahimian, y otros, 2008).
- **Desarrollo basado en arquitectura:** la eficiencia de la línea de producción de *software* depende del desarrollo de una plataforma común, por lo que la necesidad de una arquitectura genérica para una clase de productos es esencial, pudiendo reconfigurarse de forma específica para cada componente o producto determinado (Rahimian, y otros, 2008).
- **Soporte para la reusabilidad:** el desarrollo basado en componentes y el basado en capas ahorra costes de desarrollo, agiliza la entrega del producto y hace el *software* menos propenso a errores ya que los componentes no deben ser hechos desde cero cada vez (Rahimian, y otros, 2008).
- **Revisión continúa y sesiones de aprendizaje:** la metodología debería incorporar sesiones de revisión en todo el proceso para asegurar el análisis del producto y sesiones de aprendizaje

después de la entrega, para que la experiencia sea analizada y registrada, y así la abstracción del conocimiento obtenido retroalimente a todo el equipo (Rahimian, y otros, 2008).

- **Especificación temprana de la arquitectura física:** la arquitectura física debe ser elaborada en las etapas tempranas del desarrollo *software* gracias a que un alto número de riesgos técnicos son inherentes a las limitaciones de los dispositivos móviles y las diferencias en la implementación pueden ser obtenidas de características básicas. El uso de un prototipo mitigaría dichos riesgos técnicos (Rahimian, y otros, 2008).

1.8.3. Metodologías ágiles en el desarrollo de aplicaciones móviles

Debido al cambio continuo y actualizaciones que reciben estos dispositivos, el desarrollo de aplicaciones móviles se torna dependiente a la adaptación, por lo que las demandas que requiere desarrollar para un terminal móvil, deben estar sujetas a:

- Rápida evolución y capacidades limitadas.
- Diferentes estándares, protocolos y tecnologías de red.
- Necesidad de operar en diferentes dispositivos y plataformas.
- Requisitos de entrega rápida.

En el sistema operativo Android, y en toda la gama de terminales que usan este sistema existen diferentes arquitecturas. Además, los fabricantes de muchos de estos dispositivos optan por realizar cambios en el sistema, llegando a ser totalmente diferente al nativo Android con el que trabajan los programadores. Es, por ende, que los desarrolladores deben adquirir una metodología que esté sujeta al cambio continúo y permita al desarrollo de proyecto adaptarse rápidamente a las tecnologías. Todo esto desde una metodología tradicional supondría un coste elevado, por lo que resulta idóneo el uso de las metodologías ágiles (Rahimian, y otros, 2008).

Metodologías ágiles usadas actualmente en el desarrollo de aplicaciones móviles

Definiendo las características anteriores, y cumpliendo con la mayoría de estas se encontraron las siguientes metodologías que de acuerdo con (Amaya Balaguera, 2013), están actualmente siendo usadas en el desarrollo móvil.

Mobile-D

Se desarrolló como parte de un proyecto finlandés, ICAROS, alrededor del año 2004. Teniendo como objetivo, alcanzar ciclos de desarrollo muy rápidos en equipos muy pequeños. Está basado en metodologías conocidas pero aplicadas de forma estricta como: *Extreme Programming*, *Crystal Methodologies* y *Rational Unified Process*. Consta de cinco fases: exploración, iniciación, producción, estabilización y prueba del sistema. Cada una de estas fases tiene un número de etapas, tareas y prácticas asociadas. De acuerdo con (Abrahamsson, y otros, 2004), Mobile-D, obtuvo una certificación CMMI de nivel 2 y está pensado para grupos de no más de 10 desarrolladores colaborando en un mismo espacio físico. Aplicándose el ciclo de desarrollo propuesto, los proyectos deberían finalizar con el lanzamiento de productos completamente funcionales en menos de diez semanas. (Gomez Medina, y otros, 2016).

Hybrid Methodology Design

Esta metodología utiliza el modelo iterativo incremental para el proceso de desarrollo y así lograr la rápida entrega de *software* y mejorar las capacidades de gestión de riesgos. Algunas de las características ágiles que se destacan y que también se alinean con las necesidades de desarrollo de aplicaciones móviles son (Amaya Balaguera, 2013):

- Desarrollo basado en pruebas.
- Participación continúa del cliente.
- Establecimiento de prioridades en los requisitos.
- Comunicación efectiva.
- Calidad garantizada.
- Desarrolladores expertos.
- Revisión de todo el proceso y sesiones de aprendizaje.
- Proceso de adaptación.

Mobile Development Process Spiral

Esta propuesta metodológica utiliza el modelo de desarrollo en espiral como base, e incorpora procesos de evaluación de la usabilidad, priorizando la participación del usuario en todos los procesos del ciclo de vida de diseño, con el fin de garantizar un diseño centrado en el usuario, aun cuando se trata de un modelo de proceso orientado a proyectos grandes y costosos, ya que está destinado a ser un modelo de reducción de riesgos (Amaya Balaguera, 2013).

1.8.4. Selección de la metodología a emplear

Para guiar el proceso de desarrollo de la propuesta de solución se empleará la metodología ágil Mobile-D, debido a que:

- Facilita una documentación menos rigurosa y propicia mayor dinamismo para el desarrollo.
- Es apropiada para proyectos pequeños y ciclos de desarrollo cortos.
- Establece una comunicación más fluida con el cliente y participa activamente en el proceso de desarrollo de *software*.
- Permite afrontar rápidamente cambios que pueden surgir en la tecnología, durante el desarrollo.
- Se basa en el desarrollo basado en pruebas que es una de las mejores formas de asegurar la calidad.

1.9. Lenguaje de Programación

Un lenguaje de programación, es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Esta herramienta permite crear programas y *software*. Se representan en forma simbólica y de texto pudiendo ser leídos por una persona, pero interpretados únicamente por un computador (CCM, 2017).

1.9.1. Java

La principal característica de Java es la de ser un lenguaje compilado e interpretado, a la vez de ser de código abierto. Es un lenguaje orientado a objetos de propósito general y sirve de base a aplicaciones

de red, además de ser el estándar global para desarrollar y distribuir aplicaciones móviles, juegos, contenido basado en web y *software* de empresa. (Java, 2017).

El uso de java y sus facilidades para la creación de aplicaciones Android, así como la compatibilidad, documentación y el soporte brindado por Google, convierten a este lenguaje en la solución adecuada a incorporar, como parte de la línea de investigación que se lleva a cabo.

1.10. Sistema gestor de base de datos

Se define sistema gestor de bases de datos o SGBD, también llamado DBMS (*Data Base Management System*) como una colección de datos relacionadas entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina base de datos (Castillo, 2008).

1.10.1. SQLite

Sistema completo de bases de datos que soporta múltiples usabilidades, con la característica de ser ágil pero robusto. Es un SGBD relacional compatible con Atomicidad, Consistencia, Aislamiento y Durabilidad ACID (por sus siglas en inglés *Atomicity, Consistency, Isolation and Durability*), comprendida en una biblioteca relativamente pequeña. Es una librería escrita en el lenguaje C que implementa un motor de bases de datos para SQL92. Soporta tablas, índices y vistas que no necesita de un servidor para su utilización. Es capaz de escribir y leer directamente sobre ficheros que se encuentran en el disco duro. Es multiplataforma e imparcialmente se puede utilizar archivos en sistemas de 32 y 64 bits.

La base de datos se almacena en un único fichero a diferencia de otros SGBD que hacen uso de varios archivos. SQLite emplea registros de tamaño variable de forma tal que se utiliza el espacio en disco que es realmente necesario en cada momento (SQLite, 2017).

1.10.2. Realm[13]

Realm es una base de datos de objetos de código abierto, que inicialmente se lanzó para dispositivos móviles (Android / iOS), y que ahora se encuentra disponible para otras plataformas. Si ha utilizado un almacén de datos como SQLite o *Core Data*, a primera vista, *Realm Mobile Database* puede parecer un mapeador relacional de objetos (ORM) respaldado con una base de datos relacional ligera. Pero en realidad, utiliza un modelo de contenedor de datos, permitiendo almacenarlos como objetos (Realm, 2017).

Estas son algunas ventajas:

- Almacena objetos nativos: los objetos que almacena, son con los que trabaja en el resto del código.
- Acceso directo a los objetos: los datos no se copian dentro y fuera de la base de datos a la que se accede; se trabaja con los objetos directamente.
- Implementa el patrón de objetos en vivo: si tiene una instancia de un objeto almacenado en un dominio y algo más en su aplicación actualiza ese objeto, su instancia reflejará esos cambios.
- Es multiplataforma: siempre y cuando no almacene objetos específicos de plataforma en un Realm, los datos se pueden sincronizar a través de diferentes sistemas operativos. Los archivos de datos de Realm se pueden copiar entre plataformas.
- Uso sin conexión: funciona si está o no sincronizado a través del servidor de objetos y si el dispositivo está o no en línea en un momento dado. Cuando la conectividad está disponible, los cambios se sincronizan automáticamente.
- Por último, es compatible con ACID[14].

[13] https://realm.io

[14] Acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability: Atomicidad, Consistencia, Aislamiento y Durabilidad;

1.10.3. Selección del SGBD

A pesar de Realm, ser una muy buena opción, lleva muy corto tiempo como gestor de base de datos, por lo que no existe una vasta documentación; todo lo contrario, con SQLite que lleva más tiempo en el mundo de los SGBDs, además es el gestor de base de datos de Android por defecto. Por lo que se selecciona como SGBD a SQLite para guardar los datos de la aplicación a desarrollar.

1.11. Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (PC.net, 2016).

1.11.1. Entorno de desarrollo Eclipse

Eclipse es una plataforma de desarrollo de código abierto basada en Java. Por si misma, es simplemente un marco de trabajo y un conjunto de servicios para la construcción del entorno de desarrollo de los componentes de entrada. Afortunadamente, Eclipse tiene un conjunto de complementos, incluidas las herramientas de desarrollo de Java (JDT, *Java Devolopment Tools*). También, incluye el entorno de desarrollo de complementos (PDE), de interés principalmente para los desarrolladores que quieren extender el IDE, dado que les permite construir herramientas que se integran sin dificultades con el entorno (Eclipse, 2017).

1.11.2. Android Studio

Android Studio está basado en IntelliJ IDEA, un IDE para Java de *Jetbrains*, desarrollado por la multi-transnacional Google con características específicas para desarrollar en Android. Esta herramienta,

ofrece la posibilidad de ver en directo los cambios al diseño de las aplicaciones en las diferentes resoluciones que soporta Android. La interfaz permite visualizar la aplicación en diferentes idiomas.

También brinda información útil en el margen de la pantalla de código, como iconos o colores según los mencionemos en el código. Además, Android Studio incluye integración con *Google Cloud Messaging*, para no tener que salir del IDE (Julian, 2013).

1.11.3. Selección del entorno de desarrollo

Eclipse en conjunto con el ADT (Android Development Tools), complemento adicional para el desarrollo de aplicaciones Android, fue uno de los primeros entornos que surgieron en la necesidad del desarrollo para este sistema. Pero, comparado con las versiones actuales y las opciones que brinda el Android Studio, este IDE, pasa a estar obsoleto. Por lo que, se selecciona el IDE Android Studio ya que resulta recomendable para trabajar con Android, debido a que es la plataforma oficial para el desarrollo de este tipo de aplicaciones. Además, es desarrollado por Google, quien brinda el soporte necesario para dicha herramienta.

1.12. Lenguaje de modelado

1.12.1. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. UML se puede usar para modelar distintos tipos de sistemas: sistemas de *software*, sistemas de hardware, y organizaciones del mundo real (Popkin Software and Systems, 2000).

UML ofrece nueve diagramas en los cuales modelar sistemas de estos solo se utilizarán:

- Diagramas de clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de distribución para representar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares (Popkin Software and Systems, 2000).

1.13. Herramientas para la modelación del sistema

La Ingeniería de *Software* Asistida por Computadoras (CASE, por sus siglas en inglés, *Computer Aided Software Engineering*) es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores durante todos los pasos del ciclo de vida de desarrollo. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de *software* (Alfonzo Rivas, 2011).

1.13.1. DIA

DIA es un programa para la creación de diagramas GTK+ para GNU/ Linux, Unix y *Windows* liberado bajo la licencia GPL. Está inspirado en el programa comercial *Windows Visio*, aunque está más orientado hacia bosquejos sencillos para el uso ocasional. Actualmente posee objetos especiales para poder realizar diagramas de entidad relación, UML, de red y organigramas. También, es posible añadir soporte para nuevas formas de escritura mediante archivos XML, usando un subconjunto de SVG[15] para dibujar las formas (Villavicencio, 2014).

1.13.2. Rational Rose

Es una solución de gestión del ciclo de vida del *software* que permite colaboración contextual en tiempo real para equipos distribuidos. Proporciona una configuración del proceso, un marco de guía y obligatoriedad que da soporte a todo el entorno de entrega de *software* (IBM, 2006).

[15] Gráficos Vectoriales Redimensionables (del inglés *Scalable Vector Graphics*), es un formato de gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML.

- Mejora la colaboración del equipo con características integradas, que incluye la gestión de la configuración de *software*, la creación y los elementos de trabajo.
- Proporciona una gran visibilidad de actividades de proyectos y progreso del equipo con paneles de instrumentos de varios niveles y características de creación de informes.
- Facilita la planificación y ejecución de proyectos ágiles o formales con plantillas y herramientas de planificación. Las existencias de procesos coherentes mejoran la calidad del *software*.
- Ayuda a mejorar la productividad con control de origen avanzado para equipos distribuidos geográficamente.
- Mejora la colaboración del equipo con características integradas
- Proporciona una gran visibilidad de actividades de proyectos y progreso del equipo
- Ayuda a mejorar la productividad con control de origen avanzado

1.13.3. Visual Paradigm

El *Visual Paradigm* es una herramienta UML de ingeniería de *software* que favorece el desarrollo de los programas informáticos desde su planificación hasta el análisis y diseño. Su propósito general es llevar el ciclo de vida completo del desarrollo de *software* ya sea: análisis y diseño, construcción, pruebas y despliegue a través de la forma de representación de diagramas (Pressman, 2003). Como herramienta CASE posee numerosas ventajas como (Visual Paradigm, 2017):

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en la generación de artefactos automáticos.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Visual Paradigm permite la generación de innumerable diagramas, así como la seguridad de que el modelo y el código permanezcan sincronizados en todo el ciclo de desarrollo (Visual Paradigm, 2017)

1.13.4. Selección de la herramienta CASE

Por sus características, utilidades, posibilidades, e incluyendo las ventajas antes expuestas, se elige *Visual Paradigm* como herramienta de modelado, ya que se caracteriza por la disponibilidad en múltiples plataformas (Windows, Linux), permite la transformación de diagramas de Entidad-Relación en tablas de BD y posee una buena compatibilidad entre ediciones. Además, brinda grandes facilidades para la creación de los diversos diagramas, y cuenta con la posibilidad de realizar un control de versiones durante el ciclo de trabajo.

1.14. Conclusiones parciales

Una vez culminado el capítulo correspondiente a la fundamentación teórica se concluye en que:

- El estudio de los principales conceptos asociados al objeto y campo de la investigación, permitió obtener información acerca de los elementos fundamentales para desarrollar la propuesta de solución,
- El análisis de sistemas homólogos resalto las características que tienen las aplicaciones similares a la propuesta de solución, aportando los conocimientos necesarios en el desarrollo de aplicaciones para editoriales digitales en línea.
- Las metodologías ágiles resultan idóneas para el desarrollo de aplicaciones móviles, por las características que presentan a los cambios rápidos de respuesta en proyectos de corto plazo. Dentro de ellas destaca Mobile-D, por las ventajas que otorga especialmente para la solución.
- Mediante el estudio de los lenguajes y herramientas se evidenció la preponderancia del uso de Java, como lenguaje eficiente para el desarrollo móvil y de Android Studio como IDE aplicable en la solución.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1. Introducción

Este capítulo tiene como objetivo describir las actividades realizadas durante el proceso de análisis y diseño de la solución propuesta, con el objetivo de proporcionar un mejor entendimiento del sistema. Se especifican los requisitos funcionales y no funcionales que se deben cumplir para lograr este objetivo. Asimismo, se generan los artefactos definidos por la metodología de desarrollo seleccionada.

2.2. Fase de exploración

La fase de exploración es la primera fase planteada por la metodología Mobile-D. El objetivo de esta fase es definir las bases del proyecto, esto se realiza en tres etapas (Gomez Medina, y otros, 2016):

1. Establecimiento de actores: en esta etapa se definen a los involucrados del proyecto, así como los tareas, roles y responsabilidades que adquieren durante el desarrollo de la aplicación.
2. Definición de alcance: el propósito de esta etapa, es definir los objetivos para el proyecto incipiente con respecto a los contenidos, como de la línea de tiempo del proyecto.
3. Establecimiento de proyecto: es donde se definen y asignan los recursos (tanto técnicos como humanos) que se necesita para que el proyecto de desarrollo de *software* inicie. Esta etapa asegura que el proyecto pueda iniciar sin demoras causadas por falta de herramientas o entrenamiento.

2.2.1. Establecimiento de actores

Cliente: actor que trabaja activamente con el equipo de desarrollo de la aplicación móvil, aportando opiniones y sugerencias nuevas. Equipo de desarrollo de la plataforma de la EUFV

Usuarios: actor que interactúa con las funciones y que tendrá acceso a la plataforma de la EUFV.

Propuesta de producto: aplicación para el sistema operativo móvil Android, bajo el nombre de *FV Mobile*.

2.2.2. Definición de alcance

La aplicación que se propone tiene como objetivo interactuar con la plataforma de la Editorial Universitaria Félix Varela, utilizando una capa de servicios web, que garantiza la comunicación entre ambos sistemas. La misma permitirá el uso de las funcionalidades de la editorial desde teléfonos inteligentes que utilizarán el sistema operativo Android a partir de la versión 4.0.3 en adelante y que cuenten con una conexión a internet mediante Wi-Fi.

FV Mobile, permitirá al usuario permanecer autenticado y recibiendo actualizaciones del servidor de libros, además se mostrarán los libros recomendados por la editorial. La herramienta debe ser capaz de guardar en el dispositivo los datos, creando el archivo "*datafv.db*" usado para su posterior uso sin conexión.

2.2.3. Requisitos del sistema

Los requisitos de un sistema, son el proceso de establecer los servicios que el cliente requiere y los limites bajo los cuales los desarrolladores operan y se desarrolla. Estos reflejan las necesidades de los clientes ayudando a resolver problemas. Las malas o ineficientes prácticas en la definición de los requisitos pueden conllevar al fracaso del desarrollo del software y pueden provocar costes elevados dependiendo de cuan tarde estos sean descubiertos (Sommerville, 2005).

Estos se clasifican en funcionales y no funcionales.

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Sommerville, 2005).

A continuación, se listan los requisitos funcionales que cumplirá la aplicación a desarrollar. A cada requisito se le asignó un nivel de importancia o prioridad entre 1 (No muy importante) y 5 (Muy importante).

Tabla 3 Requisitos funcionales de la aplicación (Elaboración Propia).

Requisitos	Prioridad

1	**Cargar y guardar los datos**	5
2	**Actualizar datos manualmente**	4
3	**Iniciar sesión**	4
4	**Cerrar sesión**	4
5	**Listar libros recomendados**	3
6	**Listar libros más vistos**	3
7	**Listar todos los libros**	4
8	**Listar todos los autores**	4
9	**Listar todas las editoriales**	4
10	**Listar los recursos**	4
11	**Mostrar detalles del libro**	4
12	**Mostrar detalles del autor**	4
13	**Mostrar detalles de la editorial**	4
14	**Mostrar detalles del recurso**	4
15	**Visualizar libro**	5
16	**Descargar Recurso**	5
17	**Mostrar perfil de usuario**	3

Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema (Sommerville, 2005). Estos incluyen requisitos de tiempo, restricciones en el proceso de desarrollo y estándares. Los requisitos no funcionales identificados son:

Usabilidad

- RnF1. La aplicación debe poseer un diseño intuitivo, usando iconos en lugar de enormes textos para lograr una fácil utilización en usuarios con conocimientos básicos de computación o en el manejo de dispositivos móviles.
- RnF2. La aplicación será distribuida en el idioma español.
- RnF3. El sistema podrá ser utilizado por cualquier usuario sin autenticación, pero a la hora de ejercer cualquier función de visualización deberá autenticarse.

Hardware

- RnF4. Se debe disponer de al menos una interfaz de red para la conexión.
- RnF5. El dispositivo móvil que se disponga debe tener como mínimo 1.37 megabytes de espacio disponible de almacenamiento para instalar la aplicación y 3 megabytes para el uso de la base de datos.

Software

- RnF6. El dispositivo móvil deberá tener instalado el sistema operativo Android en su versión 4.0.3 o superior.

2.2.4. Establecimiento de proyecto

En esta etapa se definió el entorno tanto técnico como físico del proyecto.

- Tecnología: Android.
- Lenguaje de programación: Java.
- Librería Java: jdk8.
- Paquete de desarrollo: Android SDK
- API de desarrollo: 21
- IDE: Android Studio 2.3.
- Sistema Operativo: Android versión 4.0.3 o superior.
- Equipo: Un dispositivo móvil con sistema operativo Android versión 4.0.3 o superior.
- Metodología de desarrollo: *Mobile-D.*

2.2.4.1. Modelo de Dominio

Un modelo de dominio o modelo conceptual es una representación de las clases conceptuales del mundo real. Utilizando la notación UML, se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Permite representar conceptos del dominio de un problema determinado (Larman, 2003)

Según lo planteado por (Jacobson et al. 2000), el modelo del dominio captura, comprende y describe los objetos más importantes dentro del contexto de un sistema. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de requisitos o mediante la entrevista con los expertos del tema. Las clases que lo componen suelen aparecer en tres formas típicas:

- Objetos del negocio que representan los elementos que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o que han ocurrido.

A continuación, se presenta el diagrama del modelo del dominio de la aplicación, el cual constituye una representación gráfica de los conceptos fundamentales a tener en cuenta para su desarrollo.

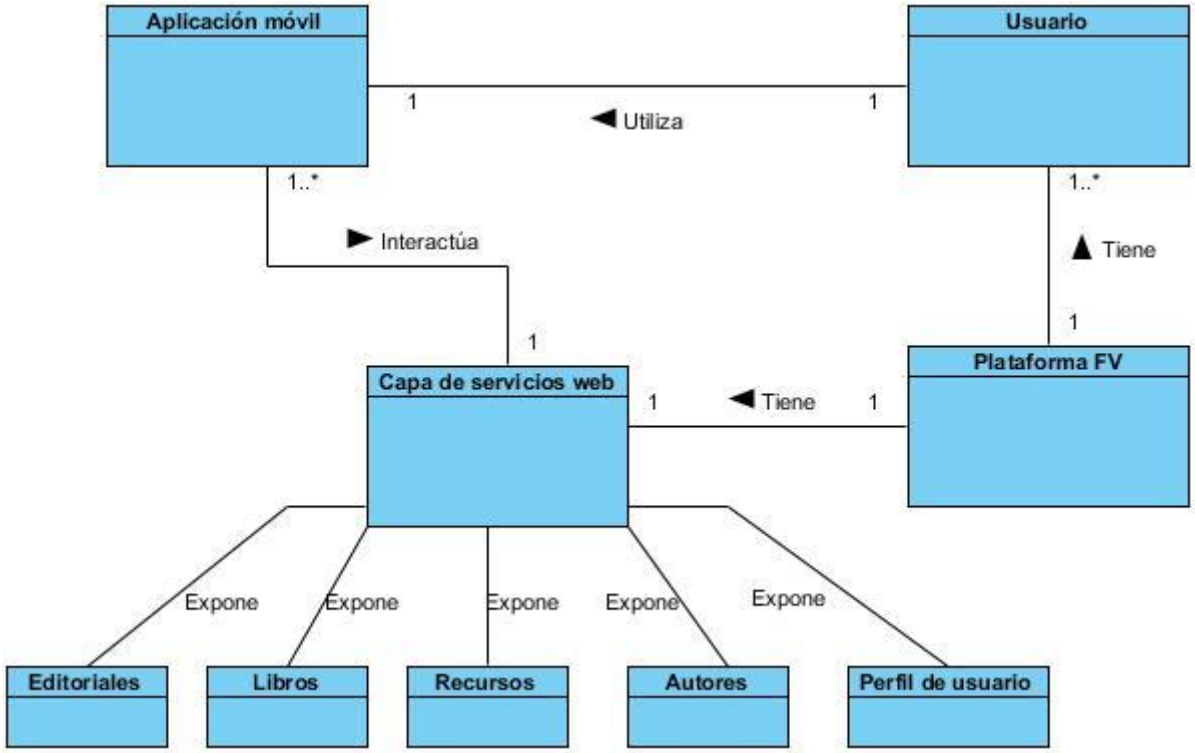


Ilustración 2 Diagrama de clases del modelo del dominio (Elaboración propia)

Conceptos asociados al dominio de la investigación

Aplicación móvil: identifica a la aplicación para el sistema operativo Android que se encarga de solicitar los datos del sistema mediante la capa de servicios web.

Usuario: todos los usuarios que tendrán acceso al sistema con el objetivo de interactuar mediante la aplicación móvil.

Plataforma FV: se refiera a la plataforma de la EUFV.

Capa de servicios web: es la herramienta que facilita la interoperabilidad entre el sistema y aplicaciones externas al mismo.

Editoriales, Libros, Recursos, Autores: cada uno de los elementos que expone la capa de servicios web.

2.2.4.2. Definición de arquitectura

La arquitectura de *software* de un programa o sistema de cómputo, es la estructura o estructuras del sistema, que comprende a los componentes del *software*, sus propiedades externas visibles y las relaciones entre ellos.

En la presente investigación se define que la arquitectura a emplear es Modelo Vista Controlador (MVC).

Patrón arquitectónico Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC), es un patrón de arquitectura de *software* encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, permitiendo no mezclar lenguajes de programación en el mismo código (Bahit, 2011).

Patrón arquitectónico MVC en Android

En el sistema operativo para dispositivos móviles Android los componentes quedan representados de la siguiente forma:

- **Modelo:** recoge la información (la lógica de la aplicación). Por ejemplo, la base de datos. Es la parte más reutilizable, se puede portar fácilmente todo el modelo de una app, a otra.
- **Vista:** se refiere a los *layouts*, a lo que el usuario ve por pantalla en cuánto ejecuta la aplicación. En Android sería todo lo que tiene lenguaje XML.
- **Controlador:** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

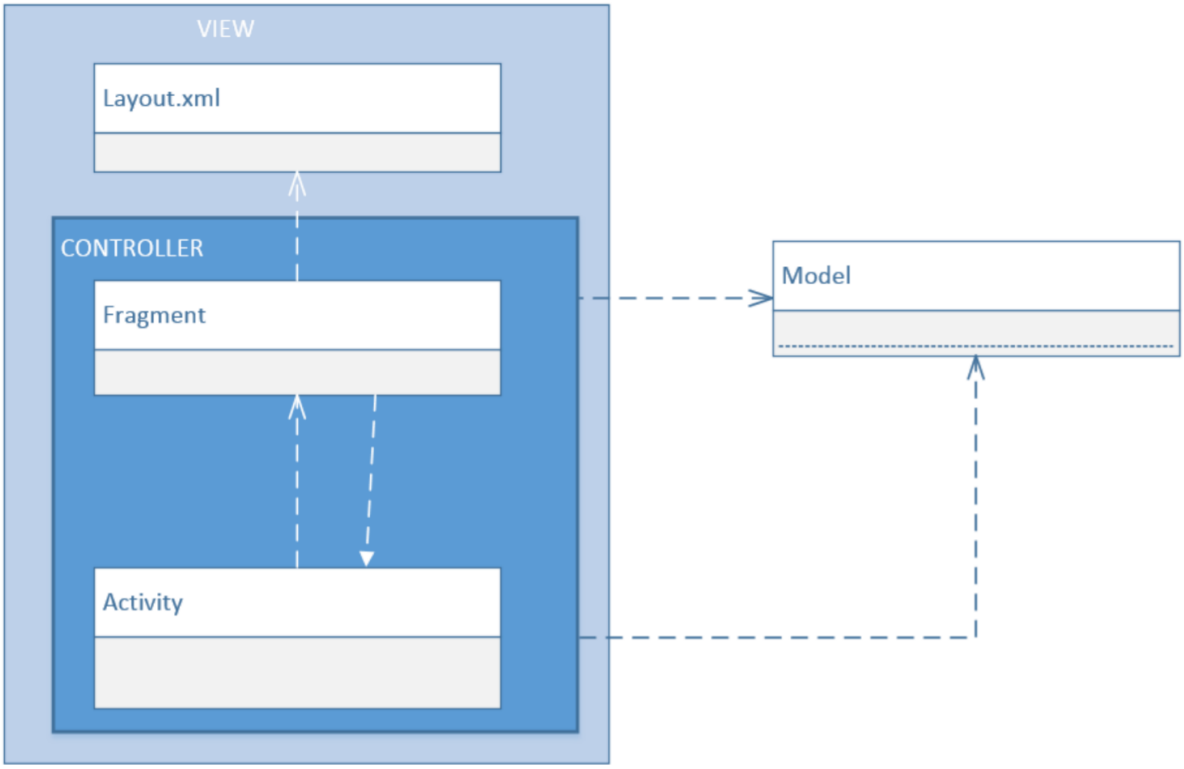


Ilustración 3 Modelo, Vista, Controlador, implementación en Android **(Lou, 2016)**.

2.3. Fase de iniciación

El propósito de esta fase es permitir el éxito de las siguientes fases del proyecto, mediante la preparación y verificación de todas las cuestiones fundamentales del desarrollo, a fin de que todos están en plena disposición de la aplicación de los requisitos seleccionados por el cliente. Los objetivos de esta fase son:

- Obtener una buena compresión global del producto para el equipo de desarrollo del proyecto, sobre los requisitos iniciales y la línea de la arquitectura.
- Preparar los requisitos físicos, técnicos y humanos, así como la comunicación con el cliente, los planes del proyecto y todas las cuestiones fundamentales de desarrollo a fin de que todo esté en plena disposición para la implementación.

2.3.1. Historias de usuario

Las historias de usuario son utilizadas en las metodologías de desarrollo ágil para la especificación de requisitos, permitiendo administrar estos requisitos sin elaborar gran cantidad de documentación, y responder rápidamente a los requerimientos cambiantes.

2.3.1.1. Descripción de historias de usuario

Las historias de usuario, sirven para describir lo que el usuario desea ser capaz de hacer. Se centran en el valor, que viene de usar el sistema en lugar de una especificación detallada de lo que el sistema debe hacer. Están concebidos como un medio para fomentar la colaboración (Lozano, 2016).

A continuación, se describe la historia de usuario (HU) para el requisito Cargar y guardar los datos. El resto de las historias de usuario se pueden observar en el Anexo I.

Tabla 4 HU1. Requisito 1: Cargar y guardar los datos.

Número: 1	**Nombre del Requisito:** Cargar y guardar los datos
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: 5	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

Al abrir la aplicación se cargan y guardan los datos en una base de datos creada en la memoria interna del dispositivo bajo el nombre de *datafv.db*.

1- Objetivo:

Cargar y guardar los datos para su posterior uso sin conexión.

2- Acciones para lograr el objetivo (precondiciones y datos):

Abrir la aplicación.

3- Comportamientos válidos y no válidos (flujo central y alternos):

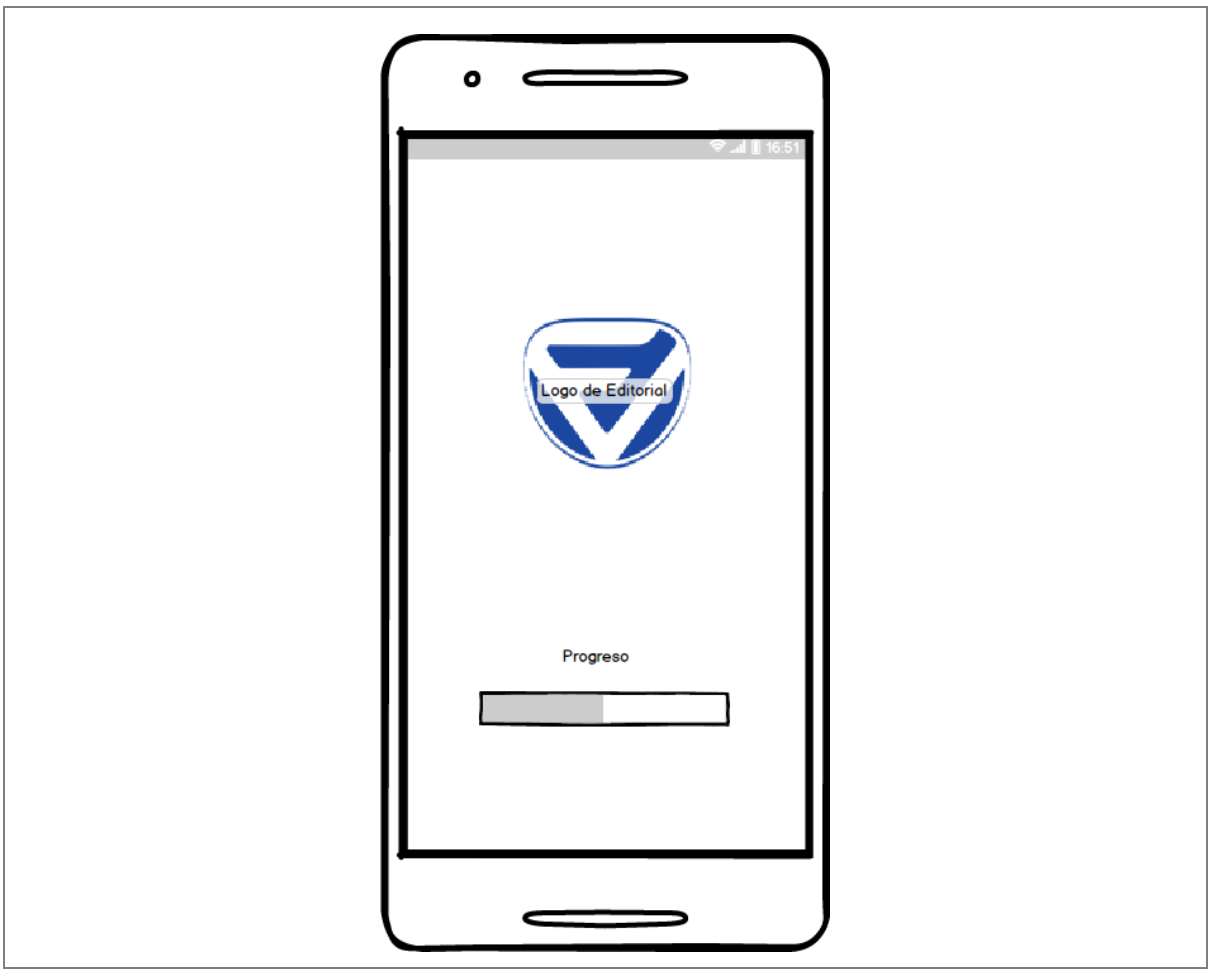
- Si el usuario es la primera vez que abre la aplicación y no se encuentra conectado se mostrará una alerta indicando que debe conectarse al menos una vez para poder guardar datos.

4- Flujo de la acción a realizar:

- Se muestra un *Splash* con el logo de la editorial y una barra de progreso indicando un proceso de carga.

Observaciones:

Prototipo de interfaz:



2.4. Patrones de diseño

Un patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre como asignar las responsabilidades a los objetos ante determinada categoría de problemas (Larman, 2003).

2.4.1. Aplicaciones de los patrones de diseño en Android

Cuando abordamos el desarrollo de una aplicación para Android, además de conocer el entorno de desarrollo para implementar la aplicación, es importante tener claro la interfaz de usuario y su comportamiento. En Android, cada uno de los patrones se basa en un prototipo físico en el que se encuentra la forma correcta de mostrar los datos, cómo ordenarlos, filtros típicos, la entrada de datos y la selección de estos (Rodríguez, 2011).

Patrones generales de *software* para asignar responsabilidades (*GRASP*):

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. A continuación, se explicarán algunos de los patrones utilizados directamente en la solución.

Experto: se basa en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (Larman, 2003). Este patrón se evidencia en la actividad principal, pues es la que recoge toda la información de todas las *activities* y *fragments* que se generan a partir de ella.

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

    controller = null;

    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar_1);
    setSupportActionBar(toolbar);

    data=SQLiteOperations.getInstance(getApplicationContext());

    downloadBooks();

    adapter = new BookAdapter(this, controller.getBooks());


    fab = (FloatingActionButton) findViewById(R.id.fab_refresh);
    fab.setOnClickListener((view) → {
```

Ilustración 4 Actividad principal de la aplicación, MainActivity.class (Captura de pantalla)

Creador: es el encargado de identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases (Larman, 2003). En la misma clase “MainActivity” de la aplicación podemos presenciar el uso de este patrón en donde se hace llamada a varios objetos en este caso *fragments*.

```
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    Fragment fragment = null;

    if (id == R.id.nav_home) {
        fragment = new FragmentHome();

    } else if (id == R.id.nav_books) {
        fragment = new FragmentBooks();

    } else if (id == R.id.nav_authors) {
        fragment = new FragmentEditorials();

    } else if (id == R.id.nav_editorials) {

    } else if (id == R.id.nav_resources) {
        fragment = new FragmentResources();
```

Ilustración 5 Clase MainActivity.class (Captura de pantalla)

Controlador: es utilizado como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que el controlador recibe los datos del usuario y los envía a las distintas clases (Larman, 2003). El uso del patrón controlador se ve reflejado en la clase *Controller* en el paquete *Models*, en donde se manejan los datos de todos los objetos del modelo.

```
public class Controller {
    private List<Editorial> editorials;
    private List<Book> books;
    private List<Author> authors;

    public Controller(List<Book> books, List<Author> authors, List<Editorial> editorials) {
        this.editorials = editorials;
        this.books = books;
        this.authors = authors;
    }

    public List<Editorial> getEditorials() { return editorials; }

    public List<Book> getBooks() { return books; }

    public List<Author> getAuthors() { return authors; }
}
```

Ilustración 6 Clase Controller.class (Captura de pantalla)

Patrones Banda de Cuatro *'Gang of Four'* (GoF):

Patrón *singleton*: tiene como objetivo asegurar que una clase determinada solo pueda poseer una instancia, proporcionando un método de clase único que la devuelva. Con este patrón la clase será la responsable de crear su propia instancia única. Este patrón se evidencia en la clase *"SQLiteOperations".*

```
public class SQLiteOperations {

    private static SQLiteController database;

    private static SQLiteOperations instance=new SQLiteOperations();

    private SQLiteOperations(){
    }

    public static SQLiteOperations getInstance(Context context){
        if(database==null){
            database=new SQLiteController(context, "datafv.db", null, 1);
        }
        return instance;
    }
```

Ilustración 7 Clase SQLiteOperations.class (Captura de pantalla)

Patrón adaptador: el objetivo de este patrón es convertir la interfaz de una clase existente en la interfaz esperada por los clientes existentes de modo que puedan trabajar de manera conjunta. El uso de adaptadores es ampliamente utilizado en Android para poblar elementos gráficos como listas con un conjunto de datos brindado por otra clase. En la aplicación se refleja el uso de este patrón en la clase *"BookItemsAdapter"* que se encuentra ubicada dentro del paquete *Adapters.*

```
public class BookItemsAdapter extends RecyclerView.Adapter<BookItemsAdapter.MyViewHolder>{
    private List<Book> bookList;
    private Context mContext;

    public BookItemsAdapter(Context mContext, List<Book> bookList){
        this.bookList=bookList;
        this.mContext=mContext;
    }

    @Override
    public BookItemsAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.general_card, parent, false);

        return new BookItemsAdapter.MyViewHolder(itemView);
    }
```

Ilustración 8 Clase BookItemsAdapter.class (Captura de pantalla)

2.5. Conclusiones parciales

Luego de transitar por las fases de exploración e iniciación planteadas por la metodología Mobile-D, se concluye que:

- La representación del modelo del dominio sirvió de base para comprender el funcionamiento estructural entre las diferentes clases que integran el sistema.
- El establecimiento de niveles de prioridad para los requisitos funcionales impondrá la creación de los componentes más importantes, en las primeras iteraciones.
- Mediante las historias de usuario y los prototipos de interfaces se especificaron los requisitos funcionales de la aplicación.
- La definición de la línea de arquitectura permitirá el desarrollo del *software* estructurado y organizado.
- De esta forma, la modelación, el diseño y la generación de los artefactos correspondientes a la metodología Mobile-D, crearon las bases para la siguiente fase de desarrollo, que logra dar cumplimiento a los requisitos tanto funcionales como no funcionales descritos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

En el capítulo anterior se describió la propuesta del sistema y se presentaron los artefactos generados, facilitando al desarrollador, entender las funcionalidades que exige el cliente y garantizar una implementación exitosa. A continuación, se procederá a las fases correspondientes que propone la metodología de desarrollo de *software* seleccionada anteriormente.

3.2. Fase de producción

Tercera fase planteada por la metodología Mobile-D, que realiza todo el proceso de implementación de la propuesta de solución. El propósito de la fase de producción es implementar las funcionalidades requeridas por el producto, mediante la aplicación del ciclo de desarrollo iterativo e incremental.

3.2.1. Modelo de datos

El modelo de datos es el lenguaje orientado a describir la base de datos, permite almacenar, organizar, manipular cantidades de datos con cierta facilidad y describir los elementos de la realidad que intervienen en el problema dado y la forma en que se relacionan estos elementos entre sí. Cuando se utiliza una base de datos para gestionar información, se está plasmando una parte del mundo real en una serie de tablas, registros y campos ubicados en un ordenador (Aguilera, 2015).

Diagrama del modelo de datos

Se utilizará para mostrar los atributos de las entidades y realizar de forma eficiente, las consultas mediante los servicios web y con el que se creará la estructura de los datos de la aplicación (Aguilera, 2015).

A continuación, se representa el diagrama del modelo de datos a emplear por la propuesta de solución. Solo se representan las entidades que hace uso la aplicación, ya que las demás no son relevantes para la implementación.

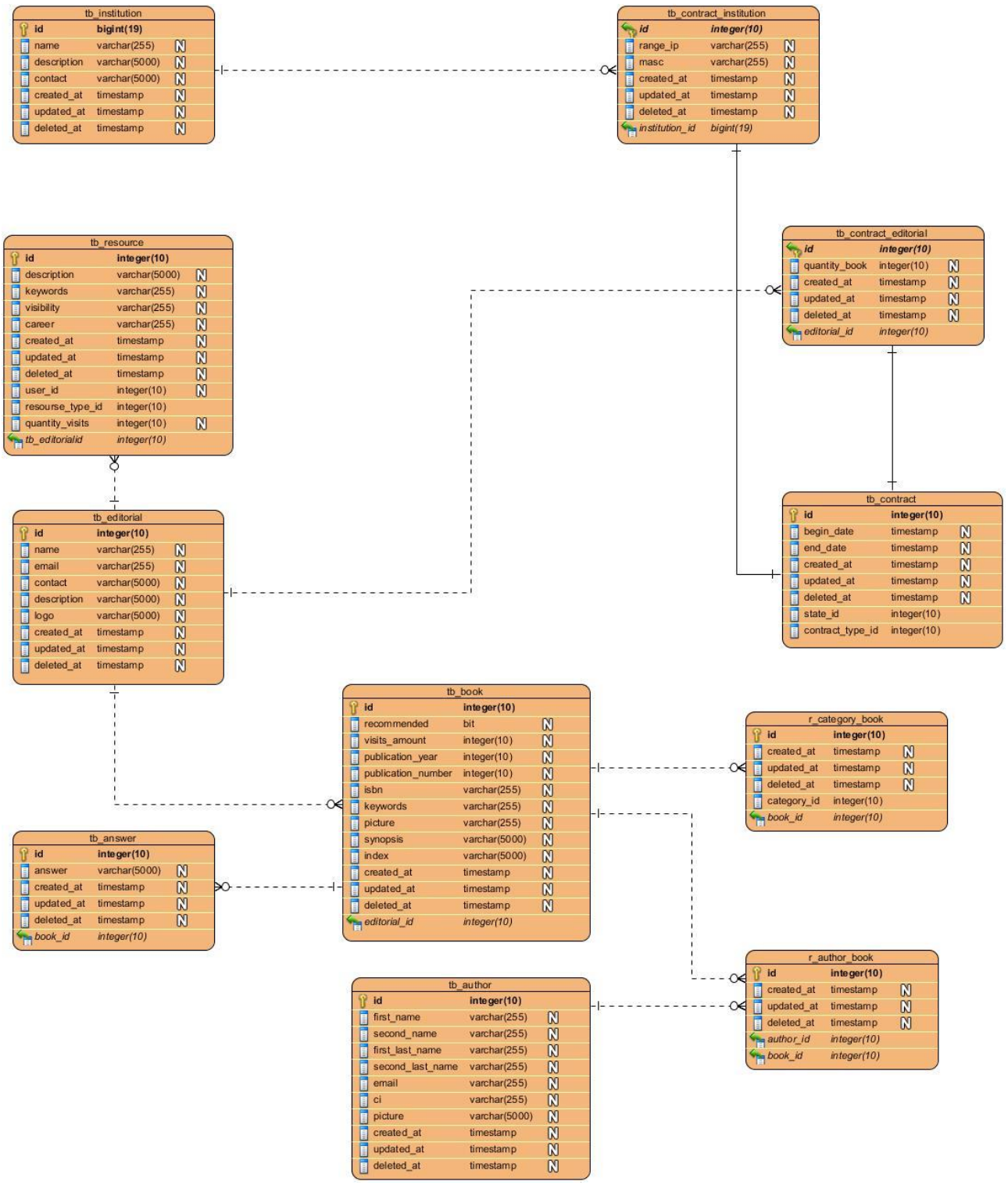


Ilustración 9 Diagrama Entidad-Relación del Modelo de Datos (Elaboración propia)

3.2.2. Tareas de ingeniería

Tareas de ingeniería (TI), relacionadas a la funcionalidad: Cargar y guardar datos.

El resto de las tareas de ingeniería para las correspondientes historias de usuario se encuentran en el Anexo II.

Tabla 5 TI1-HU1 Splash en el inicio de la aplicación (Elaboración propia).

Número tarea: 1		**Número historia:** 1
Nombre Tarea: Pantalla de bienvenida (Splash) en el inicio de la aplicación		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Realizar un SplashActivity, utilizando un ProgressDialog para el proceso de cargar los datos.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Tabla 6 TI2-HU1 Conectar y solicitar datos (Elaboración propia).

Número tarea: 2		**Número historia:** 1
Nombre Tarea: Conectar y solicitar datos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Conectar y solicitar los datos del servicio web REST utilizando la librería OKHTTP.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Tabla 7 TI3-HU1 Guardar los datos en la base de datos (Elaboración propia).

Número tarea: 3		**Número historia:** 1
Nombre Tarea: Guardar los datos en la base de datos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Guardar los datos recibidos del servicio web en el archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/01/2017	Definido	
15/01/2017	Implementación	
15/02/2017	Realizado	

3.2.3. Interfaces de la aplicación

Ilustración 10 Splash Activity usada para cargar y guardar los datos (Captura de pantalla).

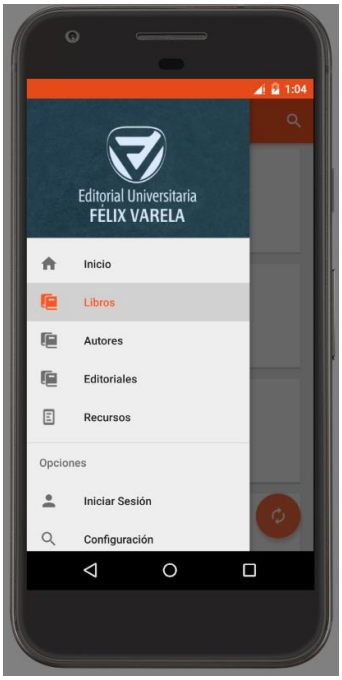


Ilustración 11 Vista del menú lateral (Captura de pantalla).

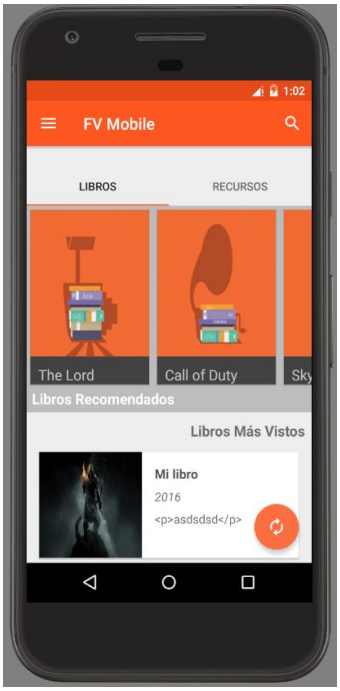


Ilustración 12 Pantalla principal mostrando los libros recomendados y más vistos (Captura de pantalla)

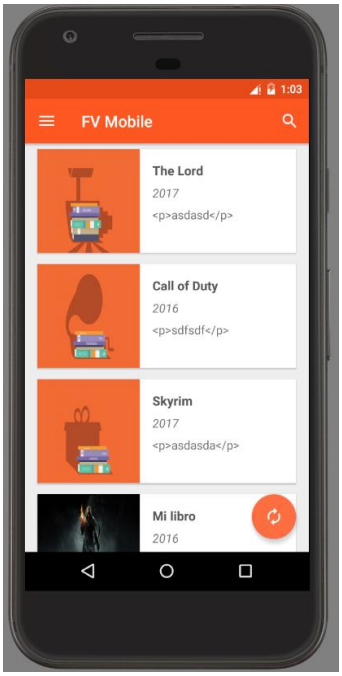


Ilustración 13 Listado de los libros (Captura de pantalla)

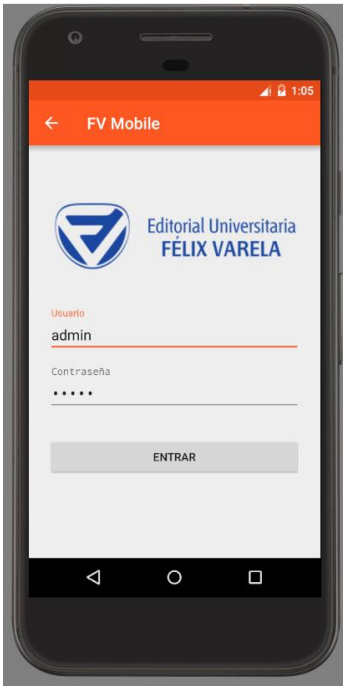


Ilustración 14 Activity iniciar sesión (Captura de pantalla).

3.2.4. Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema. Se utiliza como entrada principal en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño (Jacobson, y otros, 2000).

A continuación, se presenta el diagrama de despliegue del sistema:

Ilustración 15 Diagrama de despliegue de la aplicación (Elaboración propia).

3.3. Fase de estabilización

En esta etapa se llevan a cabo las últimas acciones de integración, donde se verificará el funcionamiento del sistema en conjunto. En esta fase se procede a la integración para vincular los módulos separados en una única aplicación. De toda la metodología, esta es la fase más importante, ya que es la que asegura la estabilización del desarrollo.

3.4. Fase de pruebas

Es la última fase planteada por la metodología Mobile-D, donde se proceden a realizar las pruebas, hasta tener una versión estable del producto según lo establecido por el cliente. Si es necesario se reparan errores, pero no se desarrolla nada nuevo. Una vez terminada esta fase se debería contar con una aplicación publicable y entregable al cliente.

3.4.1. Pruebas de *software*

Las pruebas del *software* consisten en contrastar las respuestas de una implementación de un sistema, a series de datos de prueba y examinar las respuestas y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido. (Drake, y otros, 2009)

De acuerdo con (Sommerville, 2009), es un proceso iterativo que se lleva a cabo en conjunto con la implementación. Las pruebas del sistema, siguen la finalización de la fase de implementación, permitiendo detectar defectos, verificar que los requisitos se hayan implementado correctamente, y comprobar la integración de cada uno de los componentes.

Descripción de los tipos de pruebas realizadas

A continuación, se muestran los niveles de pruebas empleados en la propuesta de solución planteadas por la metodología Mobile-D:

Pruebas unitarias: Son el proceso de probar los componentes del programa, como métodos u clases de objetos. En estas pruebas se utiliza un marco de automatización de pruebas (como JUnit) para escribir y ejecutar las pruebas del programa (Sommerville, 2009).

Estas pruebas se realizaron durante la fase de producción y conforme se desarrollaron las funcionalidades del sistema.

Pruebas de aceptación: es una descripción formal del comportamiento de un producto de *software*, con el objetivo de que sea posible automatizar la ejecución de dichas pruebas mediante una herramienta. Las pruebas de aceptación son similares a las pruebas unitarias, una prueba de aceptación generalmente tiene un resultado binario, pasa o falla. Un fallo sugiere, aunque no demuestra, la presencia de un defecto en el producto. Se pueden utilizar como un complemento a los documentos de especificación que contienen casos de uso o más texto narrativo (Agile-Alliance, 2006).

A continuación, se representa el caso de prueba de aceptación para el requisito funcional; Cargar y guardar los datos. El resto de las pruebas de aceptación se encuentran en el Anexo III.

Tabla 8 Prueba de aceptación para el requisito: Cargar y guardar los datos

Caso de prueba de aceptación
ID: 1
Historia: Cargar y guardar los datos
Aprobada/ID Defecto:
Descripción: 1. La pantalla debe mostrarse como el diseño. 2. Conectarse al servidor para obtener los datos

3. Debe mostrar una alerta si no se pudo conectar con el servidor
Resultado Esperado: 1. Inspección visual 2. Se muestra un mensaje 3. Si, se muestra la alerta
Pasa: 1. Si, se muestra como el diseño 2. Si, se muestra el mensaje 3. Si, se muestra la alerta

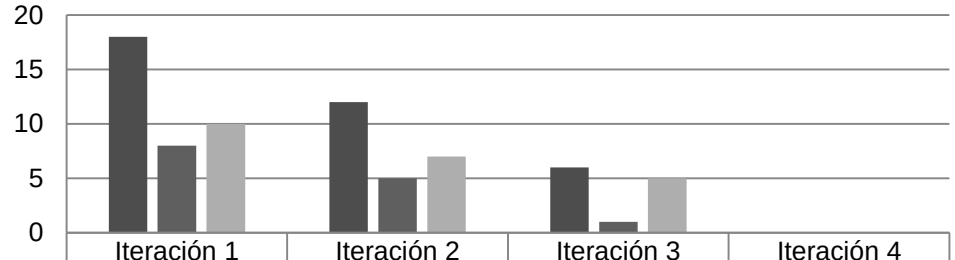
Resultados de las pruebas

Una vez ejecutadas las pruebas definidas para la aplicación fueron obtenidas un conjunto de no conformidades (NC) en cada iteración. Estas NC fueron clasificadas de acuerdo con su prioridad en significativas (S) y no significativas (NS).

Pruebas Unitarias

En las pruebas unitarias automatizadas se detectaron 18 NC, durante la implementación de las funcionalidades en la primera iteración, de ellas 8 S y 10 NS. Para la segunda iteración fueron detectadas 12 NC (5 S y 7NS) y para la tercera iteración fueron detectadas 6 NC (1 S y 5 NS). En la cuarta iteración no se detectaron NC.

Gráfica. Resultados de las pruebas unitarias

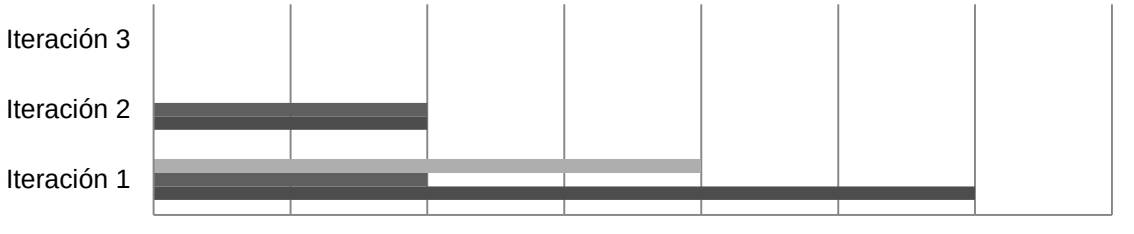


	Iteración 1	Iteración 2	Iteración 3	Iteración 4
NC	18	12	6	0
S	8	5	1	0
NS	10	7	5	0

Pruebas de aceptación

En las pruebas de aceptación realizadas con el cliente durante la primera iteración y primer encuentro se arrojaron 3 NC de estas 1 fue definida como S y 2 como NS. En la segunda iteración y encuentro con el cliente se detectó 1 NC siendo esta S. En la tercera iteración no fueron encontradas NC, siendo aprobada la aplicación por el cliente.

Gráfica. Resultados de las pruebas de aceptación



	Iteración 1	Iteración 2	Iteración 3
NS	2	0	0
S	1	1	0
NC	3	1	0

Conclusiones parciales

Luego de finalizar las fases de Producción, Estabilización y Prueba planteadas por la metodología Mobile-D se concluye que:

- En la fase de producción, se representó el modelo de datos relacional empleado en la propuesta de solución, junto a las interfaces realizadas en la primera iteración.
- Además, se representó el diagrama de despliegue de la aplicación, separando en nodos los dispositivos físicos que permitirán la completa ejecución del *software*.
- Se realizaron las pruebas unitarias correspondientes a cada funcionalidad y las pruebas funcionales para detectar y corregir las no conformidades del sistema.
- Se realizaron las pruebas de aceptación en conjunto con el cliente, permitiendo comprobar la interconexión del sistema en el entorno de la EUFV.

- De manera general en este capítulo podemos decir que las pruebas fueron satisfactorias, permitiendo corregir los errores que presentaba el sistema, y contribuyendo a obtener un producto de calidad.
- Realizadas las pruebas de *software* se debe contar con una aplicación publicable y entregable para el cliente.

CONCLUSIONES GENERALES

- El análisis de las diferentes metodologías y herramientas disponibles, posibilitó una adecuada selección de la estructura tecnológica usada en el desarrollo de aplicaciones móviles.
- El diseño de la propuesta de solución permitió la creación de la base para la implementación como siguiente fase de desarrollo, lográndose dar cumplimiento a los requisitos tanto funcionales como no funcionales descritos.
- Se implementó una aplicación móvil que permite el acceso de los clientes de la plataforma de recursos de la Editorial Universitaria Félix Varela en entornos de baja conectividad y acceso poco frecuente a la red.
- Las pruebas de software realizadas al sistema, detectaron un total de 42 no conformidades, donde 36 fueron detectadas en pruebas unitarias y 6 en pruebas de aceptación, estas fueron resueltas en cuatro iteraciones.

RECOMENDACIONES

Para perfeccionar y ampliar las funcionalidades de la solución, se propone realizar las siguientes recomendaciones:

- Actualizar la aplicación cada cierto tiempo y a medida que se realizan cambios en la plataforma de recursos Félix Varela.
- Implementar funciones de interés para el usuario como puede ser el seguimiento de lectura, utilizado en Kobo.

BIBLIOGRAFÍA

ABC. 2015. ABC Tecnología. *ABC.* [En línea] Diario ABC, S.L., 16 de Febrero de 2015. [Citado el: 5 de Mayo de 2017.] http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html.

Abrahamsson, Pekka, y otros. 2004. *Mobile-D: An Agile Approach for Mobile Application Development.* Finland : University of Oulu, 2004.

Agile-Alliance. 2006. [En línea] 2006. [Citado el: 28 de Mayo de 2017.] https://www.agilealliance.org/glossary/acceptance.

Aguilera, Sergio. 2015. *Diagrama Entidad Relación.* Cátedras : Universidad de Belgrano, 2015.

Alegsa, Leandro. 2016. Alegsa. [En línea] 28 de Junio de 2016. http://www.alegsa.com.ar/Dic/ios.php.

Alfonzo Rivas, Elizbeth. 2011. *Uso de Herramientas case para el modelado de negocios.* Medellin, Colombia : s.n., 2011.

Alto Nivel. 2010. Alto Nivel. *Alto Nivel.* [En línea] 20 de Octubre de 2010. http://www.altonivel.com.mx/sistemas-operativos-para-moviles/.

Amaya Balaguera, Yohn Daniel. 2013. *Agile methodologies in the development of applications for mobile devices.* Colombia : Universidad Pedagógica y Tecnológica de Colombia, 2013.

Android Developers. 2016. Android Developers. [En línea] 2016. https://developer.android.com/.

Bahit, Eugenia. 2011. *El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.* 2011.

Barry, Douglas K. 2016. Service Architecture. [En línea] Barry & Associates, Inc., 2016. http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html.

Castillo, Carlos. 2008. *Sistemas de Información.* 2008.

CCM. 2017. CCM. [En línea] Enero de 2017. http://es.ccm.net/contents/304-lenguajes-de-programacion.

Referencias Bibliográficas

Cubadebate. 2016. Cubadebate. *Cubadebate.* [En línea] 2016. http://www.cubadebate.cu/noticias/2016/02/08/2020-mas-personas-con-moviles-que-con-electricidad-y-agua/.

Developers, Android. 2017. [En línea] 5 de Mayo de 2017.

DRAE. 2017. Diccionario de la Real Academia Española. [En línea] 2017. http://dle.rae.es/?id=ENm66Ur.

Drake, J.M. y López, Patricia. 2009. Ingeniería de Programación. *Ingeniería de Programación.* 2009.

Eclipse. 2017. Eclipse.org. [En línea] 2017. https://www.eclipse.org/.

Editorial Universitaria Félix Varela. 2013. Editorial Universitaria Félix Varela. *Editorial Universitaria Félix Varela.* [En línea] 2013. http://www.epfv.com.cu/?q=quienes_somos.

Eguaras, Maria. 2014. Maria Eguaras. *Consultoria Digital.* [En línea] 8 de Diciembre de 2014. [Citado el: 9 de Mayo de 2017.] http://marianaeguaras.com/editorial-no-se-relaciona-solo-con-libros-sino-con-contenidos/.

Fitzgerald, G. y Avison, David E. 2006. *Information system .* s.l. : McGraw-Hill Education, 2006.

Freeman, E. and Freeman, E. 2004. *Head First Design Patterns.* 2004.

Gomez Medina, Jhoan Sebastian y Hernandez Zuleta, David Felipe. 2016. *Metodología Para El Desarrollo De Aplicaciones Móviles "Mobile-D".* Armenia : Universidad de Quindio, 2016.

González, Felipe Luis Martínez. 2011. *Aplicaciones para dispositivos móviles.* Valencia : Universidad Politécnica , 2011.

IBM. 2006. GSInnova. [En línea] 2006. http://www.rational.com.ar/herramientas/herramientasrational.html.

IBM, Rest. 2016. *Using the REST API on the XGS.* 2016.

Infobae. 2016. Infobae. [En línea] 2016. http://www.infobae.com/.../las-aplicaciones-en-ios-fallan-tres-veces-mas-que-en-android/.

Informajoven. 2006. Informajoven. *Informajoven.* [En línea] 13 de Marzo de 2006. http://www.informajoven.org/info/informacion/I_12_4.asp.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Java. 2017. Java. *Java.* [En línea] 2017. [Citado el: 24 de 01 de 2017.] http://www.Java.com/.

Julian, Guillermo. 2013. GENBETA. [En línea] 15 de Mayo de 2013. [Citado el: 24 de 01 de 2017.] https://www.genbeta.com/movil/android-studio-el-nuevo-ide-de-google-para-desarrollar-en-android.

Juventud Rebelde. 2015. Ministerio de Comunicación. *MINCOM.* [En línea] 2015. http://www.mincom.gob.cu/?q=node/1008.

Kulesz, Octavio. 2016. *Informacion del mundo editorial.* [En línea] 13 de Agosto de 2016. http://valordecambio.com/11863/.

Larman, Craig. 2003. *UML y Patrones. Segunda edición. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2003.

Larman, Craig. 2003. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 2003.

Lou, Tian. 2016. A comparison of Android Native App Architecture. *A comparison of Android Native App Architecture.* Espoo : Aalto University, 2016.

Lozano, Angel. 2016. Angel Lozano. *Experto en Metodologias de Desarrollo de Software.* [En línea] Angel Lozano, 28 de febrero de 2016. http://www.angellozano.com/requisitos-del-sistema-vs-casos-uso-vs-historias-usuario/.

Marquina, Julián. 2015. Julián Marquina. [En línea] 8 de Octubre de 2015. http://www.julianmarquina.es/16-aplicaciones-para-leer-libros-en-tus-dispositivos-moviles/.

Molina Rivera, Yeicy Juliana, Sandoval Cardona, Jonathan y Toledo Franco, Santiago Alberto. 2012. *Sistema Operativo Android: Características y Funcionalidad para Dispositivos Móviles.* Pereira : Universidad Tecnológica de Pereira, 2012.

OASIS. 2006. OASIS Web Services. [En línea] 2006. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

Open Handset Alliance. 2009. [En línea] 2009. http://www.openhandsetalliance.com/android_overview.html.

PC.net. 2016. PC.net. [En línea] 2016. http://pc.net/glossary/definition/ide.

Pimienta, Pedro. 2014. deIdeaAapp. [En línea] 5 de Mayo de 2014. https://deideaaapp.org/tipos-de-aplicaciones-moviles-y-sus-caracteristicas/.

Popkin Software and Systems. 2000. *Modelado de sistemas con UML.* 2000.

Pozo, Julián David Morillo. 2015. *Introducción a los Dispositivos Moviles.* Catalunya : Universida Oberta de Catalunya, 2015.

Pressman, Roger S. 2003. *Ingeniería de Software, un enfoque práctico. Quinta Edición.* Nueva York : McGraw-Hill, 2003.

Rahimian, V. y Ramsin, R. 2008. *Designing and agile methodology for mobile software development: A hybrid method engineering approach.* Marrakech : Second International Conference on Research Challenges in Information Science, 2008.

Realm. 2017. Realm. [En línea] 2017. [Citado el: 22 de Abril de 2017.] https://news.realm.io/news/realm-object-centric-present-day-database-mobile-applications/.

Roberth G. Figueroa, Camilo J. Solís. 2016. *Metodologías Tradicionales vs. Metodologías Ágiles.* 2016.

Rodríguez, Txema. 2011. Genbeta:dev. [En línea] 2 de Abril de 2011. [Citado el: 5 de Enero de 2017.] https://m.genbetadev.com/androidpatterns.

Sánchez, Alberto. 2015. Indismatic. [En línea] 10 de noviembre de 2015. http://www.indismatic.es/cadena-ser-rioja-baja/diferencias-android-ios-windows-phone/.

Soler Pérez, V. 2008. *El uso de las TIC como herramienta didáctica en la escuela.* España : Centro educativo de Sevilla, 2008.

Sommerville, Ian. 2005. *Ingeniería del software. Séptima edición.* s.l. : Pearson Addison Wesley, 2005.

—. 2009. Software Engineering. *Software Engineering 9th edition.* Massachusetts, United States of America : Pearson Education, Inc, 2009.

SQLite. 2017. SQLite. [En línea] 2017. [Citado el: 24 de 01 de 2017.] https://sqlite.org/about.html.

TechTarget. 2015. TechTargert. [En línea] 2015. http://www.techtarget.com/.

The Computer Language Company. 2015. The Computer Language Company. [En línea] 2015. http://lookup.computerlanguage.com/host_app/search?cid=C999999&term=smartphone&lookup.x=33&lookup.y=14.

Vázquez, Sergio. 2016. Digital-Editorial. [En línea] 16 de Enero de 2016. http://digital-editorial.com/la-app-kindle/.

—. 2016. Digital-Editorial. [En línea] 17 de Enero de 2016. http://digital-editorial.com/las-aplicaciones-ereader-2/.

—. 2015. Digital-Editorial. [En línea] 29 de Diciembre de 2015. http://digital-editorial.com/la-app-google-play-books/.

Vazquez, Sol Beatriz. 2011. Tecnologia e Informatica. [En línea] 24 de Enero de 2011. https://solvasquez.wordpress.com/2011/01/24/definicion-de-sistema-operativo/.

Villavicencio, Cesar. 2014. Herramienta Case "DIA". [En línea] 6 de junio de 2014. http://cesarvillavicencio.blogspot.com/.

Visual Paradigm. 2017. Visual Paradigm. [En línea] 2017. [Citado el: 24 de 01 de 2017.] https://www.visual-paradigm.com/.

W3C. 2000. W3C. [En línea] 2000. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

Referencias Bibliográficas

ANEXO I: HISTORIAS DE USUARIO

HU1. Requisito 1: Cargar y guardar los datos.

Número: 1	**Nombre del Requisito:** Cargar y guardar los datos
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: 5	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

Al abrir la aplicación se cargan y guardan los datos en una base de datos creada en la memoria interna del dispositivo bajo el nombre de *datafv.db*.

1- Objetivo:

Cargar y guardar los datos para su posterior uso sin conexión.

2- Acciones para lograr el objetivo (precondiciones y datos):

Abrir la aplicación.

3- Comportamientos válidos y no válidos (flujo central y alternos):

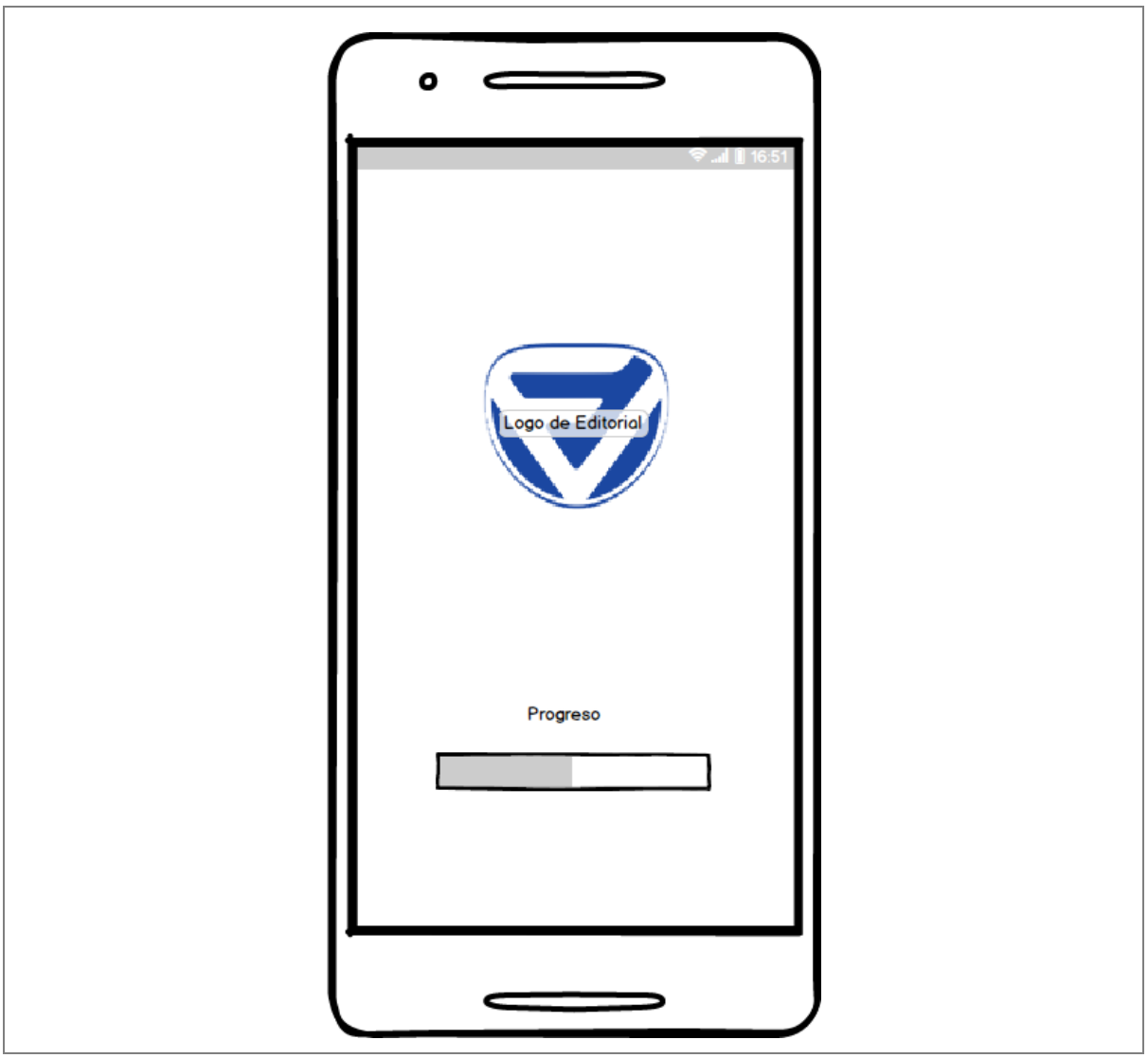
- Si el usuario es la primera vez que abre la aplicación y no se encuentra conectado se mostrará una alerta indicando que debe conectarse al menos una vez para poder guardar datos.

4- Flujo de la acción a realizar:

- Se muestra un *Splash* con el logo de la editorial y una barra de progreso indicando un proceso de carga.

Observaciones:

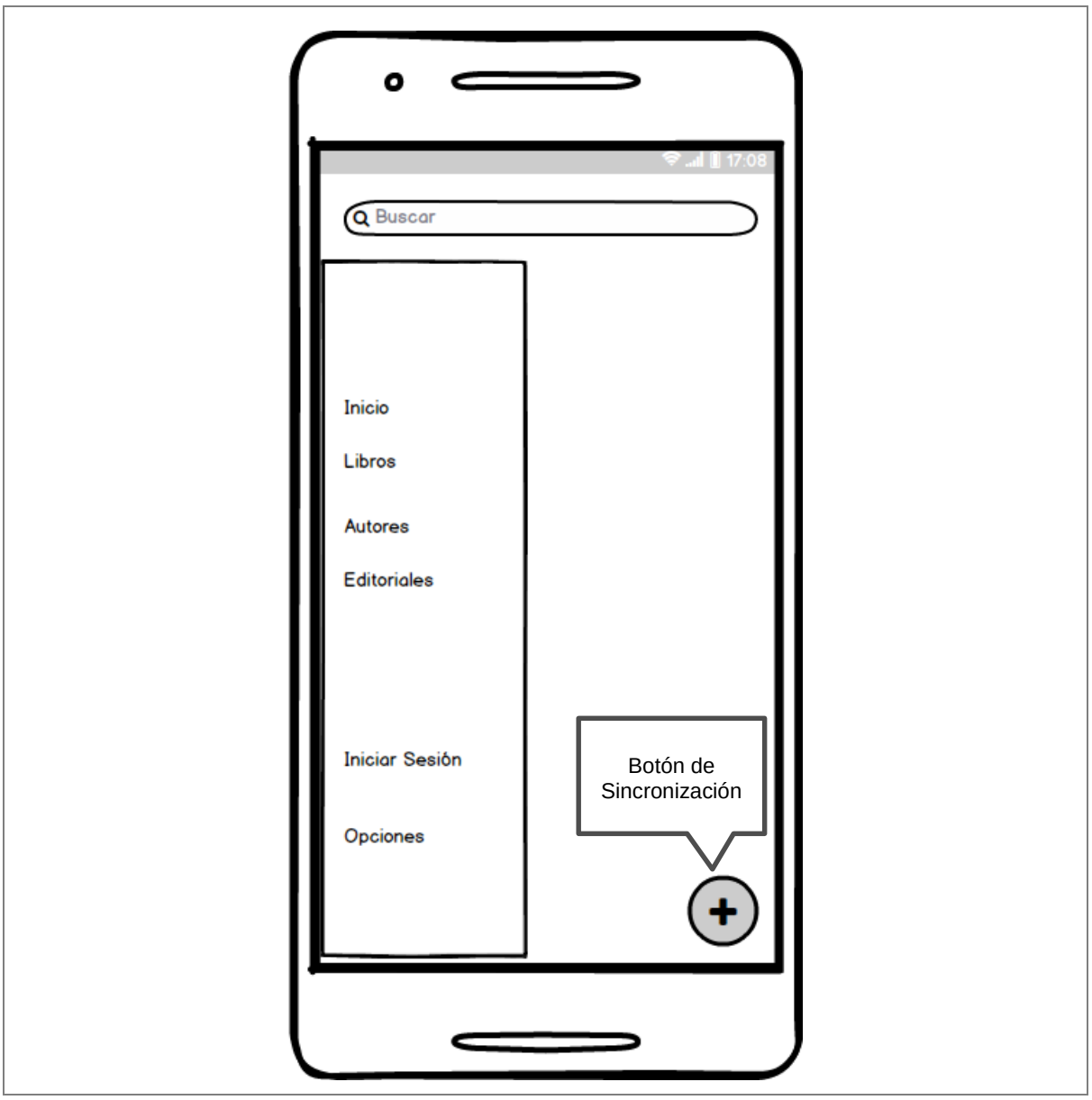
Prototipo de interfaz:



HU2. Requisito 2: Actualizar datos manualmente.

Número: 2	**Nombre del Requisito:** Actualizar datos manualmente
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

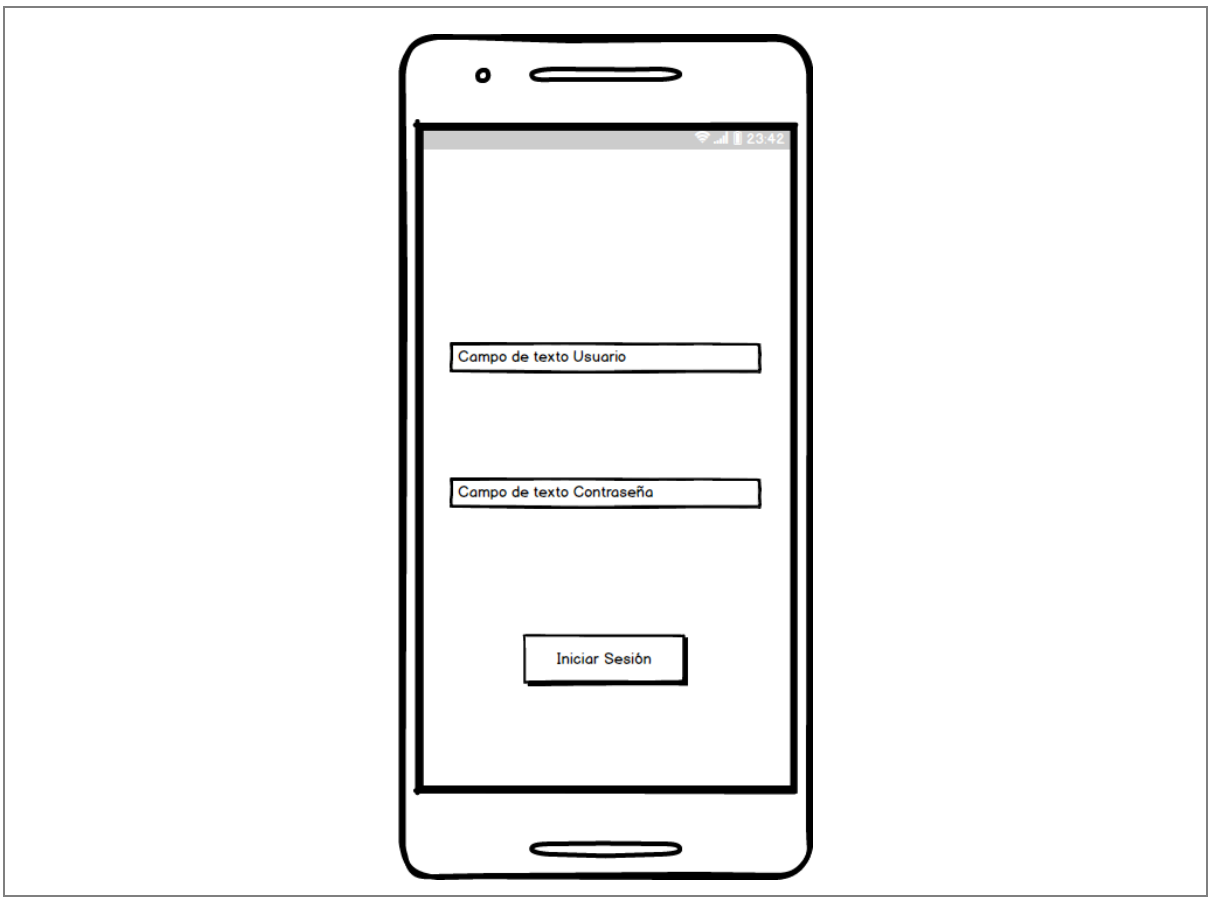
Descripción: El usuario en este caso actualiza manualmente los datos. **1- Objetivo:** Actualizar los datos de la aplicación con los nuevos datos de la plataforma de recursos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para actualizar hay que: - Estar conectado a una conexión de red. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Pulsar sobre el botón de sincronizar en la esquina inferior derecha.
Observaciones:
Prototipo de interfaz:



HU3. Requisito 3: Iniciar sesión.

Número: 3	**Nombre del Requisito:** Iniciar sesión
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 2da
Prioridad: 5	**Tiempo Estimado:** N/A

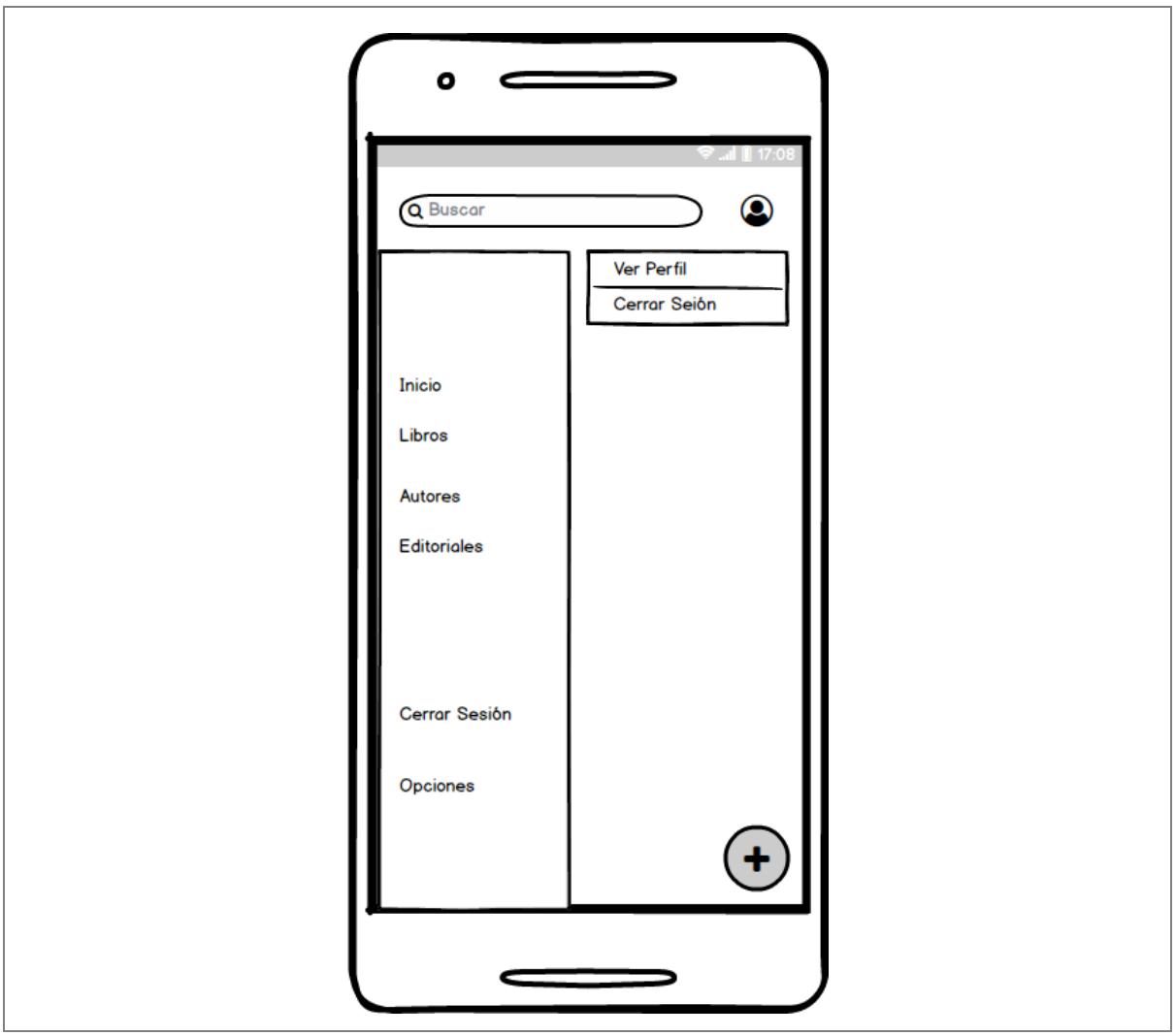
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: El usuario puede iniciar sesión en el sistema de la EUFV con su usuario y contraseña. Una vez autenticado podrá permanecer indefinidamente o hasta que decida cerrar su sesión. **1- Objetivo:** Iniciar sesión en la aplicación de la plataforma de la EUFV. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para iniciar sesión hay que: - Deslizar el panel de opciones de la izquierda. - Pulsar sobre ‘Iniciar sesión’. - Ingresar usuario y contraseña. - Acto seguido, presionar en ‘Entrar’. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - Debe permitir permanecer con la sesión abierta hasta que el usuario lo decida - Cuando el usuario ingresa su usuario y contraseña correctamente, automáticamente mostrará la vista inicial de la aplicación con su perfil de usuario en la parte superior derecha.	
Observaciones:	
Prototipo de interfaz:	



HU4. Requisito 4: Cerrar sesión

Número: 4	**Nombre del Requisito:** Cerrar sesión
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: 5	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

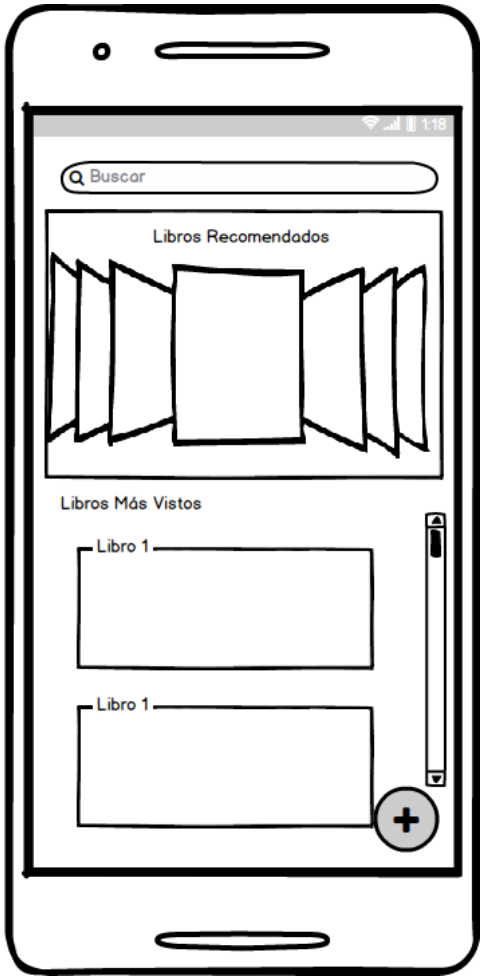
Descripción: El usuario, que previamente inicio sesión en la aplicación puede cerrar la sesión cuando desee. **1- Objetivo:** Cerrar la sesión iniciada previamente en la aplicación de la plataforma de la EUFV. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para cerrar sesión hay que: - Deslizar el panel de opciones de la izquierda. - Pulsar sobre ‘Cerrar sesión’. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - Debe permitir cerrar sesión al usuario previamente autenticado.
Observaciones:
Prototipo de interfaz:



HU5. Requisito 5: Listar libros recomendados.

Número: 5	**Nombre del Requisito:** Listar libros recomendados
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad:	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

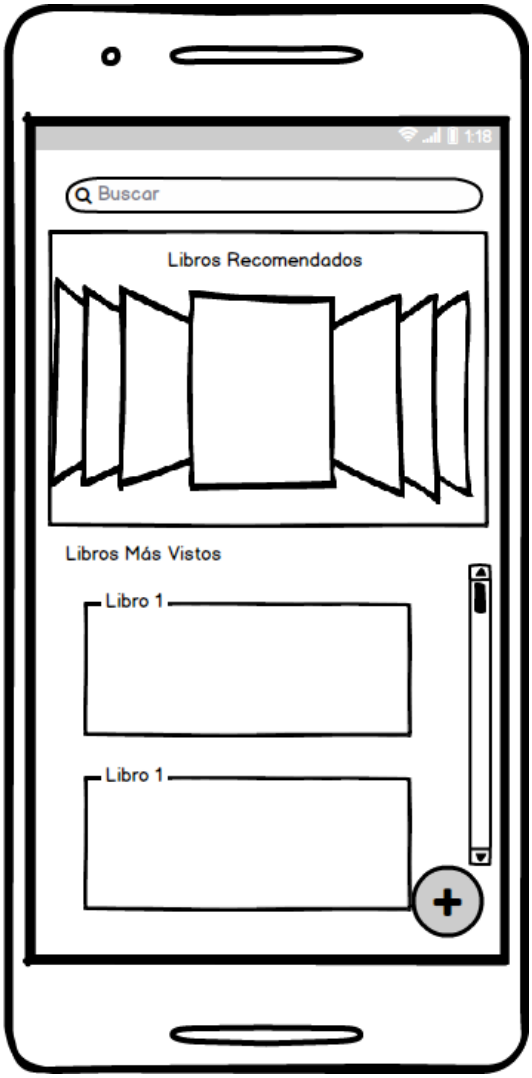
Descripción: El usuario en este caso podrá visualizar los libros recomendados por la plataforma. **1- Objetivo:** Listar los libros recomendados por la plataforma **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los libros recomendados hay que: - Estar conectado a una conexión de red. - Ejecutar la aplicación **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación - Al cargar la aplicación el sistema muestra el listado de los libros recomendados
Observaciones:
Prototipo de interfaz:



HU6. Requisito 6: Listar libros más vistos.

Número: 6	**Nombre del Requisito:** Listar libros más vistos
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

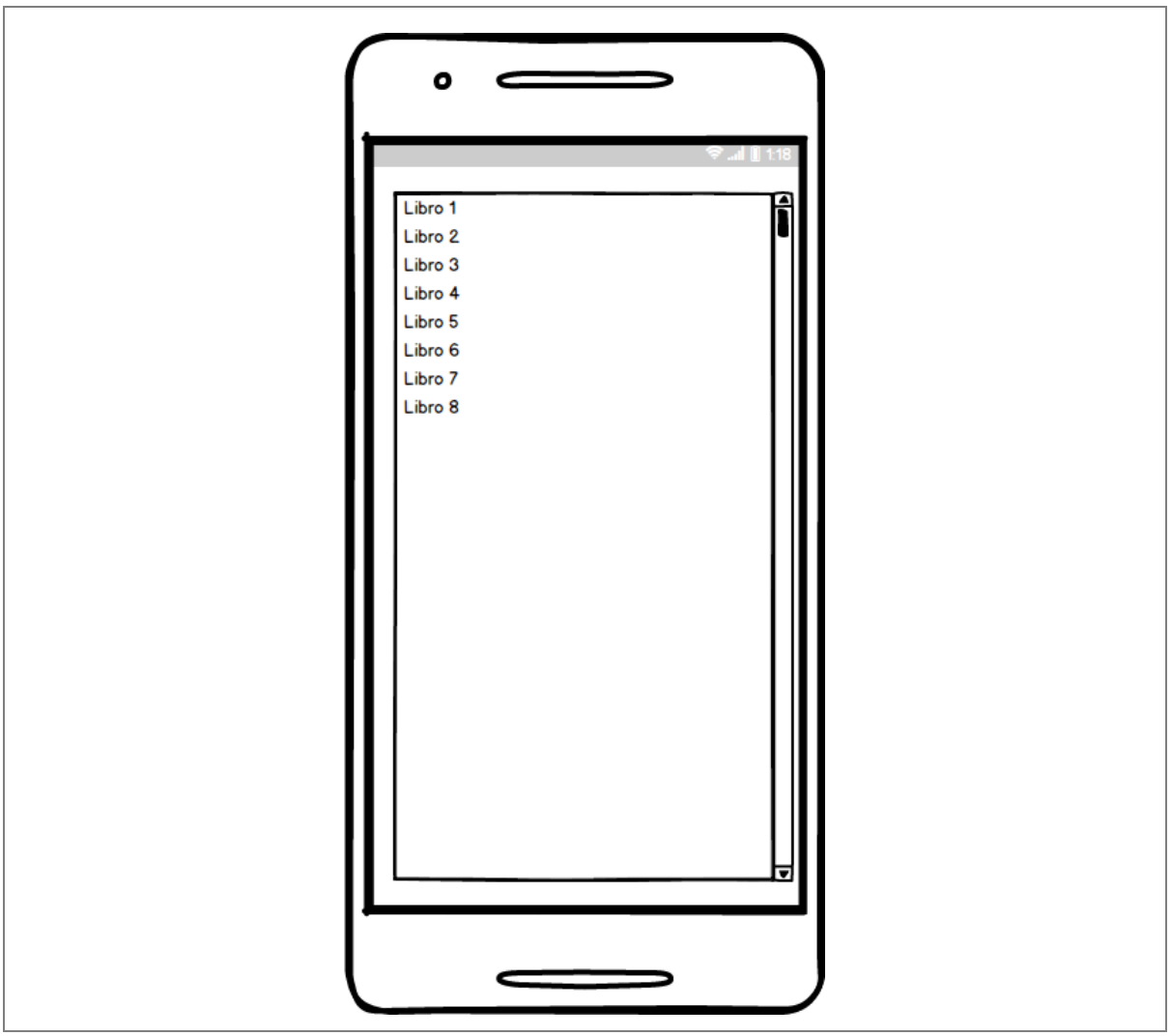
Descripción: El usuario en este caso podrá visualizar los más vistos por la plataforma. **1- Objetivo:** Listar los libros más vistos de la plataforma. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los libros más vistos hay que: - Estar conectado a una conexión de red. - Ejecutar la aplicación. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Al cargar la aplicación el sistema muestra el listado de los libros más vistos.
Observaciones:
Prototipo de interfaz:



HU7. Requisito 7: Listar todos los libros.

Número: 7	**Nombre del Requisito:** Listar todos los libros
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción: El usuario en este caso podrá visualizar los todos los libros existentes en la plataforma. **1- Objetivo:** Listar los libros más vistos de la plataforma. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar todos los libros más vistos hay que: - Estar conectado al menos la primera vez a una conexión de red para poder descargar los datos. - Pulsar sobre la opción de Libros **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación - Pulsar sobre la opción del menú Libros
Observaciones:
Prototipo de interfaz:



HU8: Requisito 8: Listar todos los autores.

Número: 8	**Nombre del Requisito:** Listar todos los autores
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Media	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

El usuario en este caso podrá visualizar los todos los libros existentes en la plataforma.

1- Objetivo:

Listar todos los autores de la plataforma.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para listar los autores hay que:

- Estar conectado a una conexión de red al menos una vez para descargar los datos.

- Abrir el menú que se encuentra a un costado de la pantalla.

- Pulsar sobre Autores.

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

- El usuario abre la aplicación.

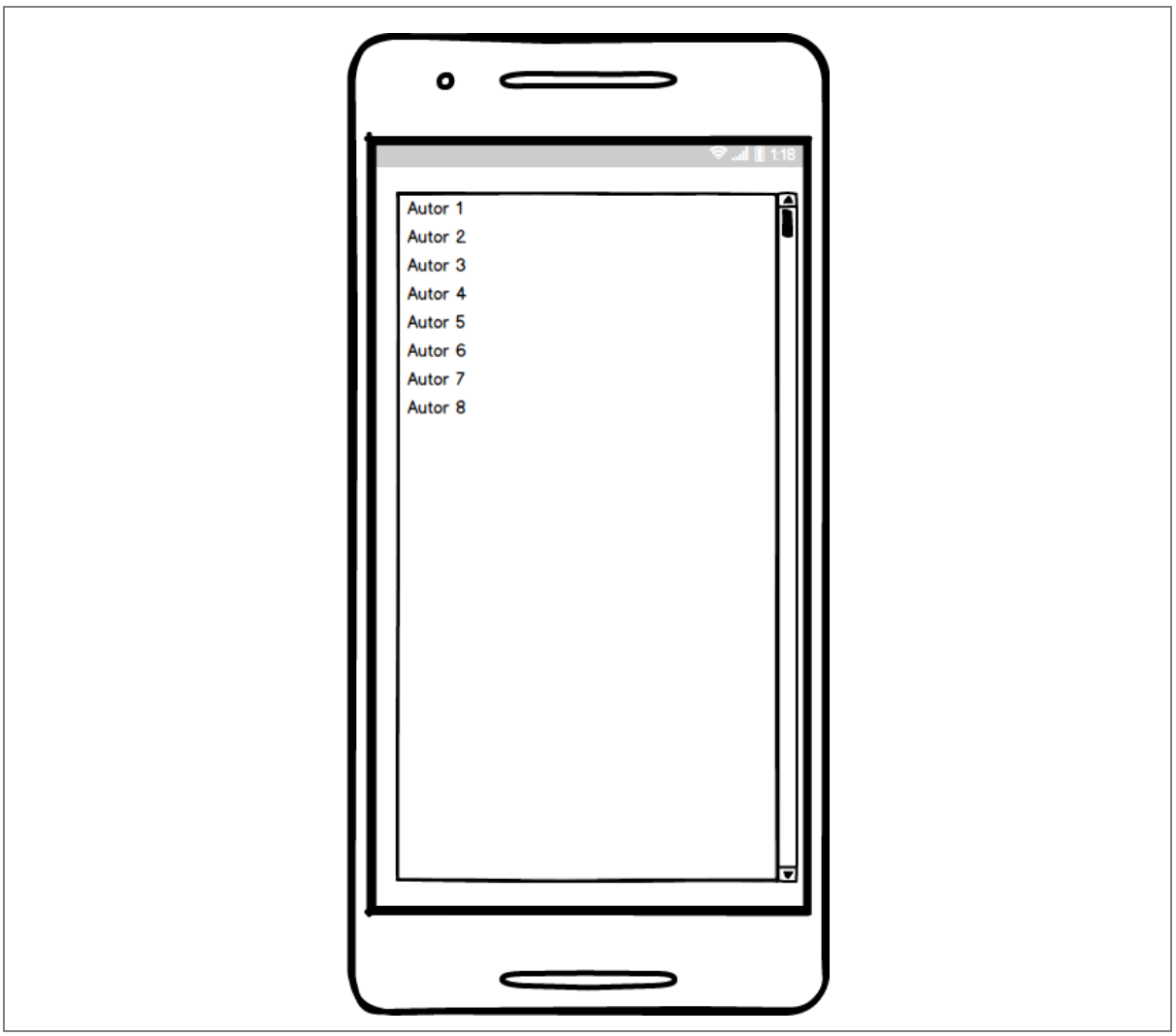
- Al cargarse los datos se mostrará la pantalla principal.

- Acto seguido abrir el menú al lado izquierdo deslizándolo o simplemente pulsando sobre el icono superior izquierdo.

- Pulsar sobre Autores para mostrar el listado de los autores.

Observaciones:

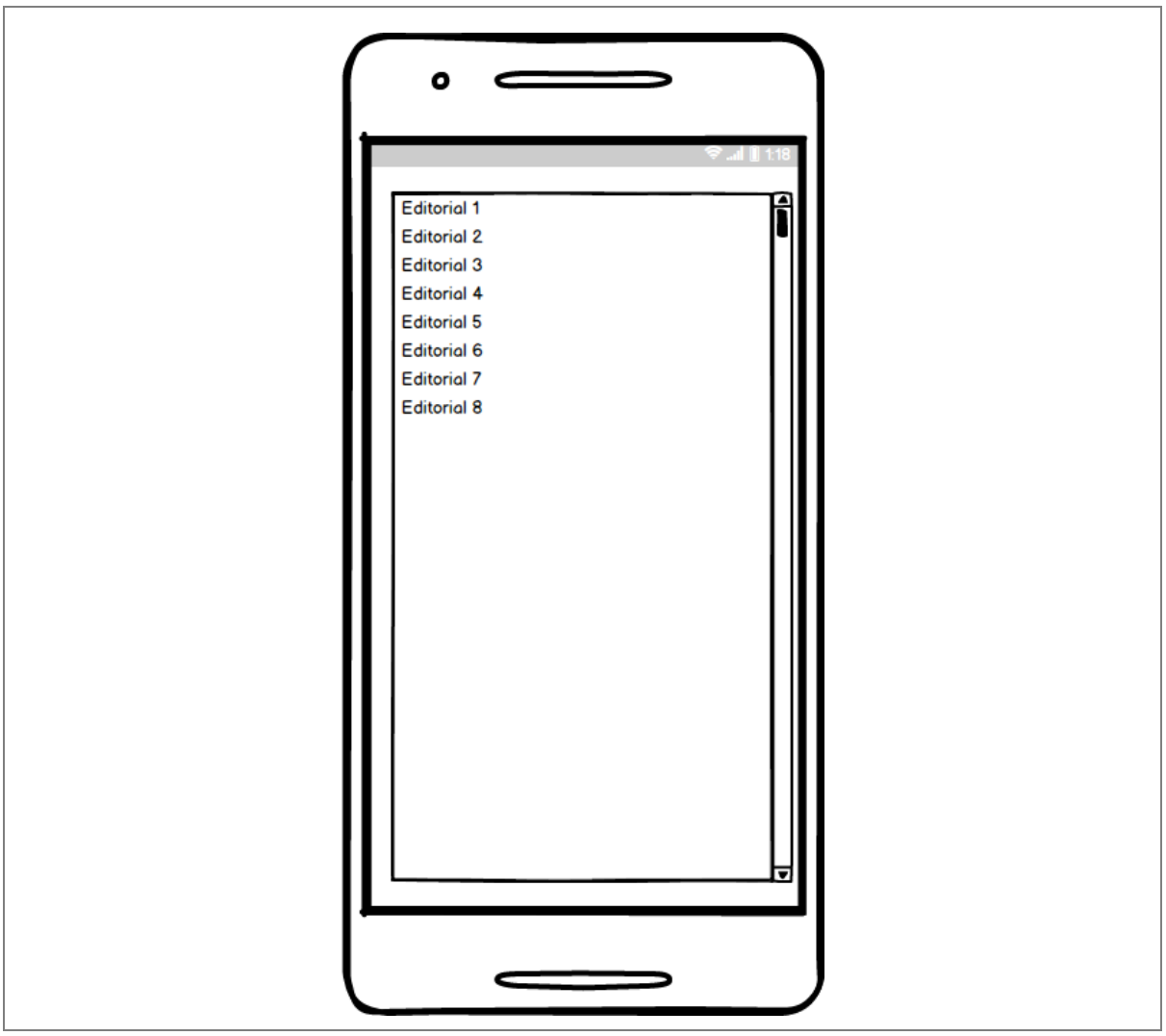
Prototipo de interfaz:



HU9. Requisito 9: Listar todas las editoriales.

Número: 9	**Nombre del Requisito:** Listar todas las editoriales
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

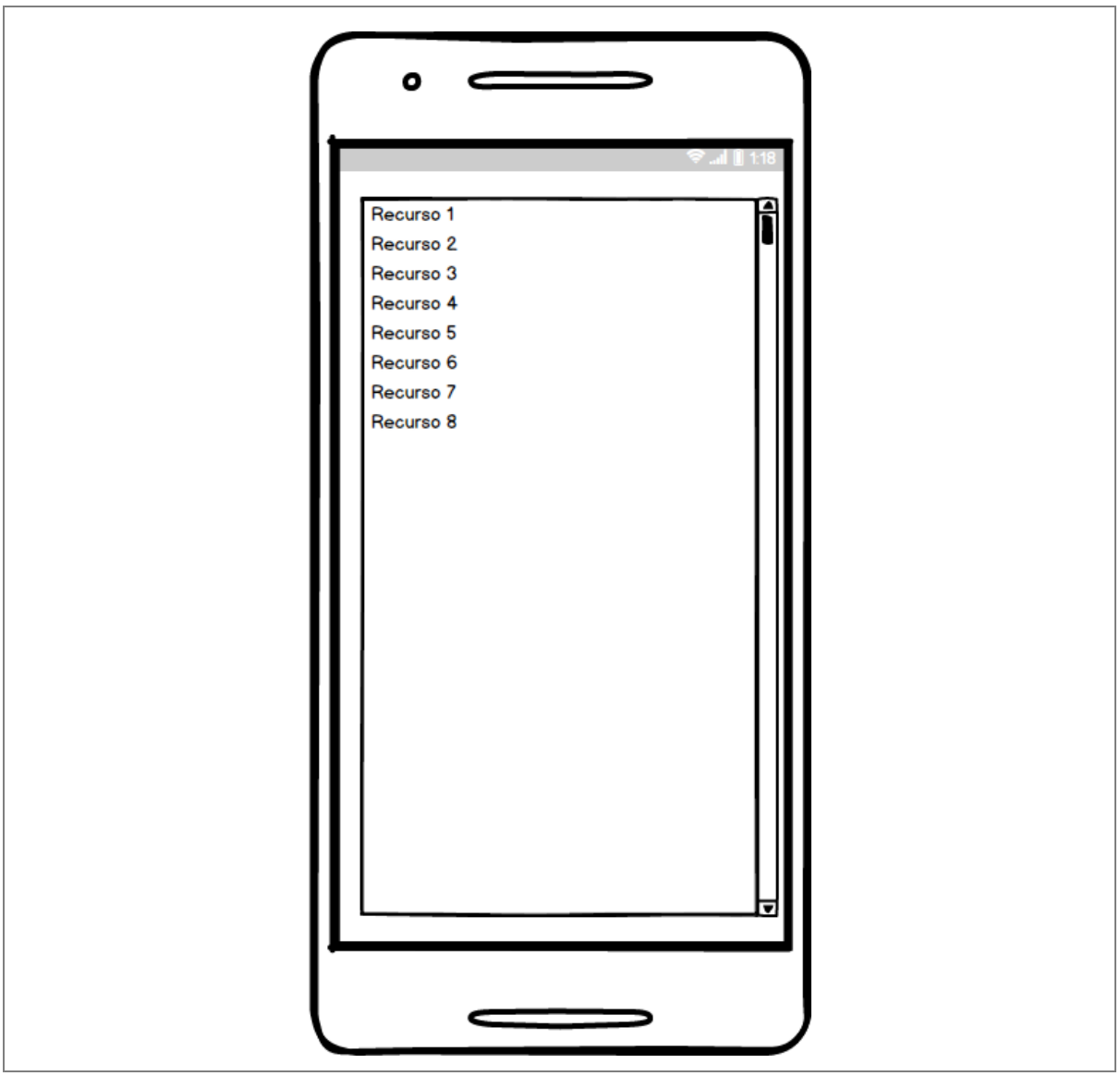
Descripción: El usuario en este caso podrá visualizar las editoriales existentes en la plataforma. **1- Objetivo:** Listar las editoriales de la plataforma. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar las editoriales hay que: - Estar conectado a una conexión de red al menos una vez para descargar los datos. - Abrir el menú que se encuentra a un costado de la pantalla. - Pulsar sobre Editoriales **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Al cargarse los datos se mostrará la pantalla principal. - Acto seguido abrir el menú al lado izquierdo deslizándolo o simplemente pulsando sobre el icono superior izquierdo. - Pulsar sobre Editoriales para mostrar el listado de los autores.
Observaciones:
Prototipo de interfaz:



HU10. Requisito 10: Listar los recursos.

Número: 10	**Nombre del Requisito:** Listar los recursos
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción: El usuario en este caso podrá visualizar los recursos existentes en la plataforma. **1- Objetivo:** Listar los recursos de la plataforma. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los recursos hay que: - Estar conectado a una conexión de red al menos una vez para descargar los datos. - Abrir el menú que se encuentra a un costado de la pantalla. - Pulsar sobre Recursos. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Al cargarse los datos se mostrará la pantalla principal. - Acto seguido abrir el menú al lado izquierdo deslizándolo o simplemente pulsando sobre el icono superior izquierdo. - Pulsar sobre Recursos para mostrar el listado de los recursos.
Observaciones:
Prototipo de interfaz:



HU11. Requisito 11: Mostrar detalles de libro.

Número: 11	**Nombre del Requisito:** Mostrar detalles de libro
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

El usuario en este caso podrá visualizar los detalles de un libro en particular existente en la plataforma.

1- Objetivo:

Mostrar los detalles del libro.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para lograr esto hay que:

- Estar registrado en la plataforma pulsando sobre Iniciar Sesión.

- Abrir el menú que se encuentra a un costado de la pantalla.

- Pulsar sobre Libros

- Luego pulsar sobre el libro que desee visualizar para ver sus detalles.

3- Comportamientos válidos y no válidos (flujo central y alternos):

-Si no estás autenticado se mostrará la pantalla de Iniciar Sesión para poder hacerlo.

-Si estas autenticado previamente se mostrarán los detalles del libro.

4- Flujo de la acción a realizar:

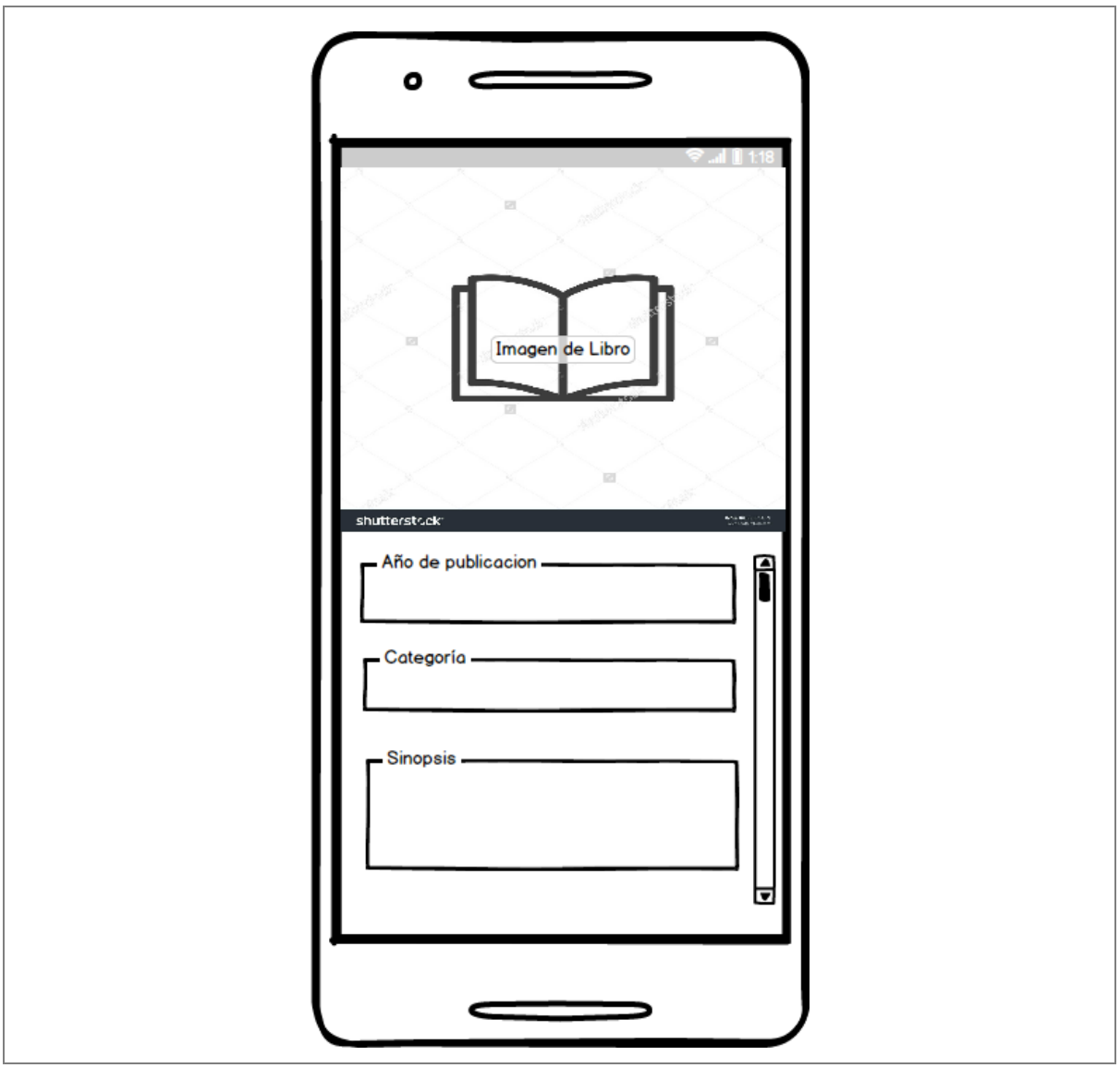
- El usuario abre la aplicación

- Pulsar sobre la opción del menú Libros

- Pulsar sobre un libro en especifico

Observaciones:

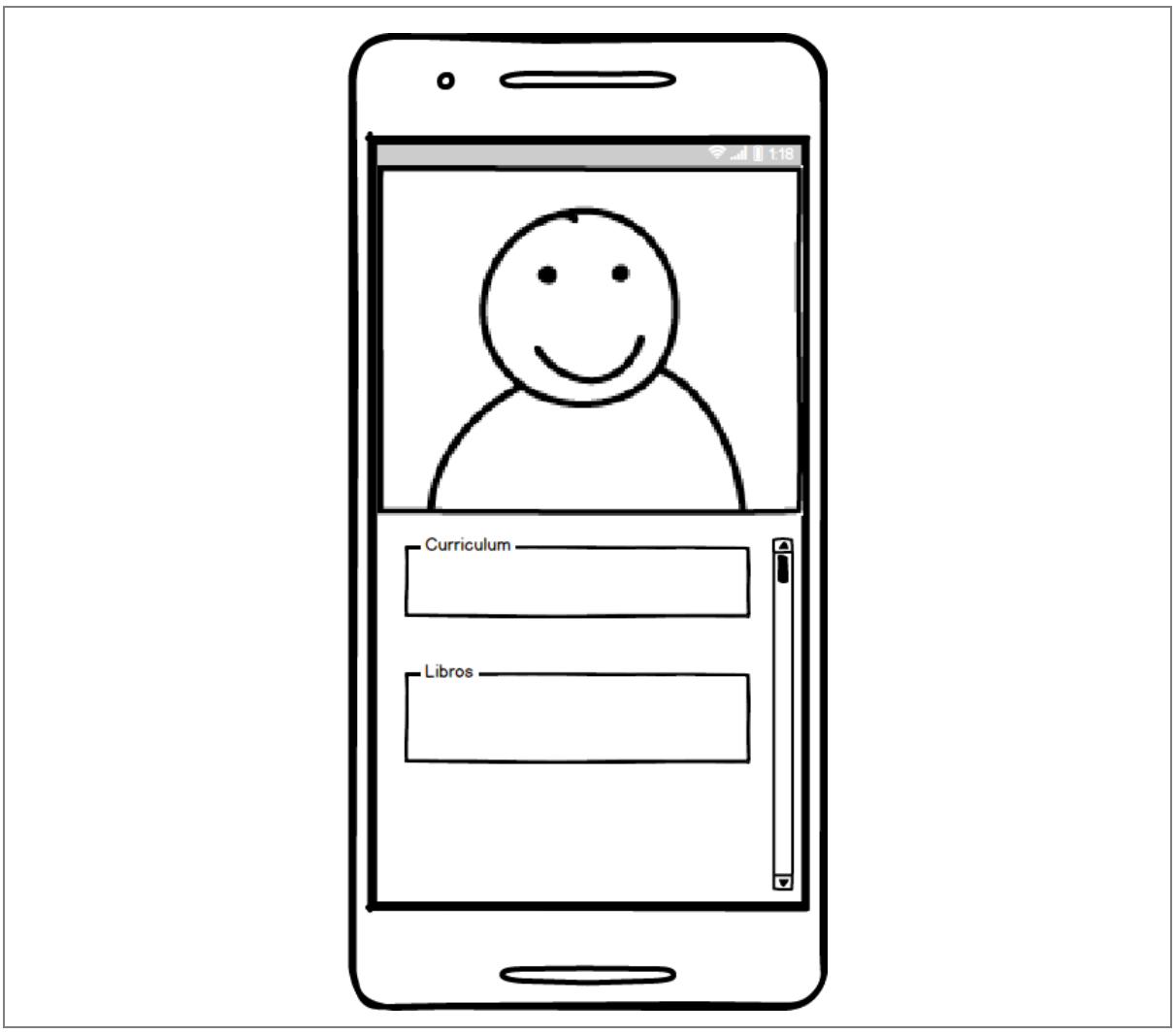
Prototipo de interfaz:



HU12. Requisito 12: Mostrar detalles de autor

Número: 12	**Nombre del Requisito:** Mostrar detalles de autor
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

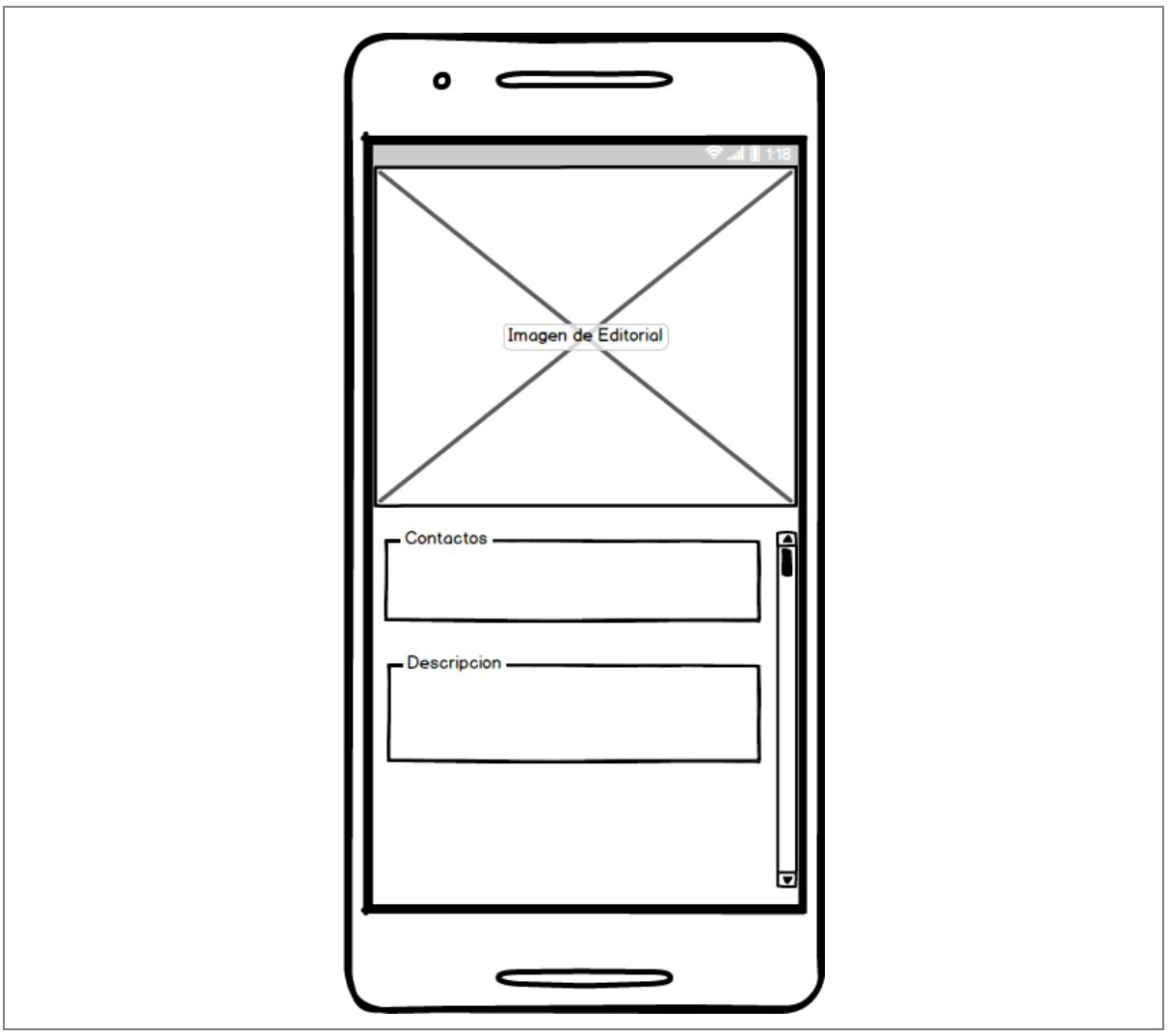
Descripción: El usuario en este caso podrá visualizar los detalles de un autor existente en la plataforma. **1- Objetivo:** Mostrar los detalles del autor. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para lograr esto hay que: - Estar registrado en la plataforma pulsando sobre Iniciar Sesión. - Abrir el menú que se encuentra a un costado de la pantalla. - Pulsar sobre Autores. - Luego pulsar sobre el autor que desee para para ver sus características. **3- Comportamientos válidos y no válidos (flujo central y alternos):** -Si no estás autenticado se mostrará la pantalla de Iniciar Sesión para poder hacerlo. -Si estas autenticado previamente se mostrarán los detalles del autor. **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Pulsar sobre la opción del menú Autores. - Pulsar sobre un autor en específico.
Observaciones:
Prototipo de interfaz:



HU13. Requisito 13: Mostrar detalles de editorial.

Número: 13	**Nombre del Requisito:** Mostrar detalles de editorial
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción: El usuario en este caso podrá visualizar los todos los libros existentes en la plataforma. **1- Objetivo:** Mostrar los detalles del autor. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para lograr esto hay que: - Estar registrado en la plataforma pulsando sobre Iniciar Sesión. - Abrir el menú que se encuentra a un costado de la pantalla. - Pulsar sobre Editoriales. - Luego pulsar sobre la editorial que desee para para ver sus características. **3- Comportamientos válidos y no válidos (flujo central y alternos):** -Si no estás autenticado se mostrará la pantalla de Iniciar Sesión para poder hacerlo. -Si estas autenticado previamente se mostrarán los detalles de la editorial. **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Pulsar sobre la opción del menú Editoriales. - Pulsar sobre una editorial en específico.
Observaciones:
Prototipo de interfaz:



HU14. Requisito 14: Mostrar detalles de recurso.

Número: 14	**Nombre del Requisito:** Mostrar detalles de recurso
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

El usuario en este caso podrá visualizar los detalles de un recurso en particular existente en la plataforma.

1- Objetivo:

Mostrar los detalles del libro.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para lograr esto hay que:

- Estar registrado en la plataforma pulsando sobre Iniciar Sesión.

- Abrir el menú que se encuentra a un costado de la pantalla.

- Pulsar sobre Recursos

- Luego pulsar sobre el recurso que desee visualizar para ver sus detalles.

3- Comportamientos válidos y no válidos (flujo central y alternos):

-Si no estás autenticado se mostrará la pantalla de Iniciar Sesión para poder hacerlo.

-Si estas autenticado previamente, se mostrarán los detalles del recurso.

4- Flujo de la acción a realizar:

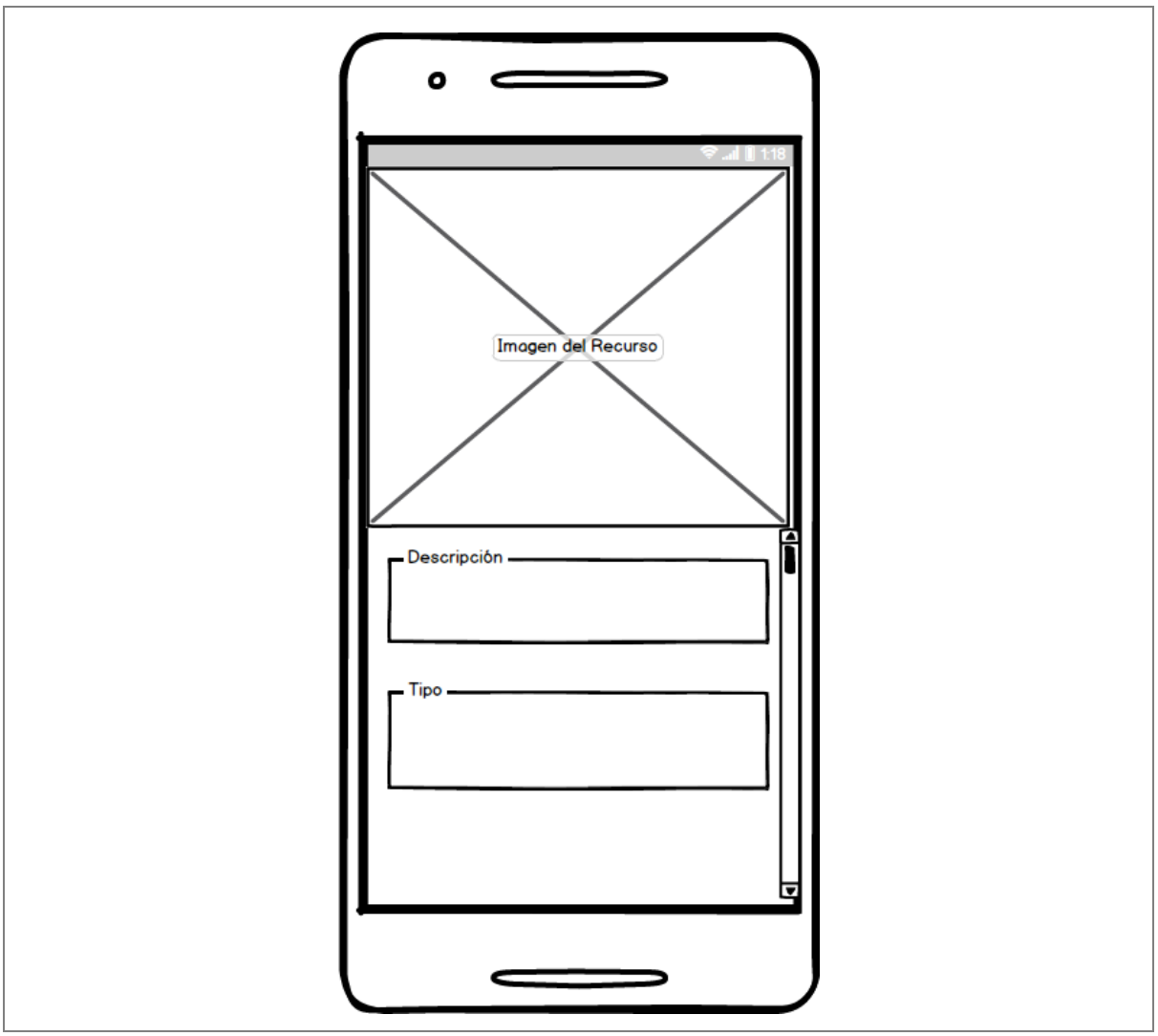
- El usuario abre la aplicación.

- Pulsar sobre la opción del menú Recursos.

- Pulsar sobre un recurso en específico.

Observaciones:

Prototipo de interfaz:



HU15. Requisito 15: Visualizar libro.

Número: 15	**Nombre del Requisito:** Visualizar libro
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción: El usuario en este caso podrá visualizar un libro en formato PDF desde la aplicación **1- Objetivo:** Visualizar el libro que el usuario desee leer. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar un libro hay que: - Estar conectado a una conexión de red. - Estar autenticado previamente en la plataforma. **3- Comportamientos válidos y no válidos (flujo central y alternos):** **4- Flujo de la acción a realizar:** - El usuario abre la aplicación. - Pulsar sobre el libro que desee visualizar. - Le mostrará los detalles, luego presionar en el icono que aparece en la esquina inferior derecha, de la imagen del libro.
Observaciones:
Prototipo de interfaz:

HU16. Requisito 16: Descargar recurso.

Número: 16	**Nombre del Requisito:** Descargar Recursos
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

El usuario tiene la opción de descargar los recursos siempre que la plataforma y el acceso lo permita

1- Objetivo:

Descargar el recurso que el usuario desee.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para descargar el recurso hay que:

- Estar conectado a una conexión de red.

- Estar autenticado previamente en la plataforma.

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

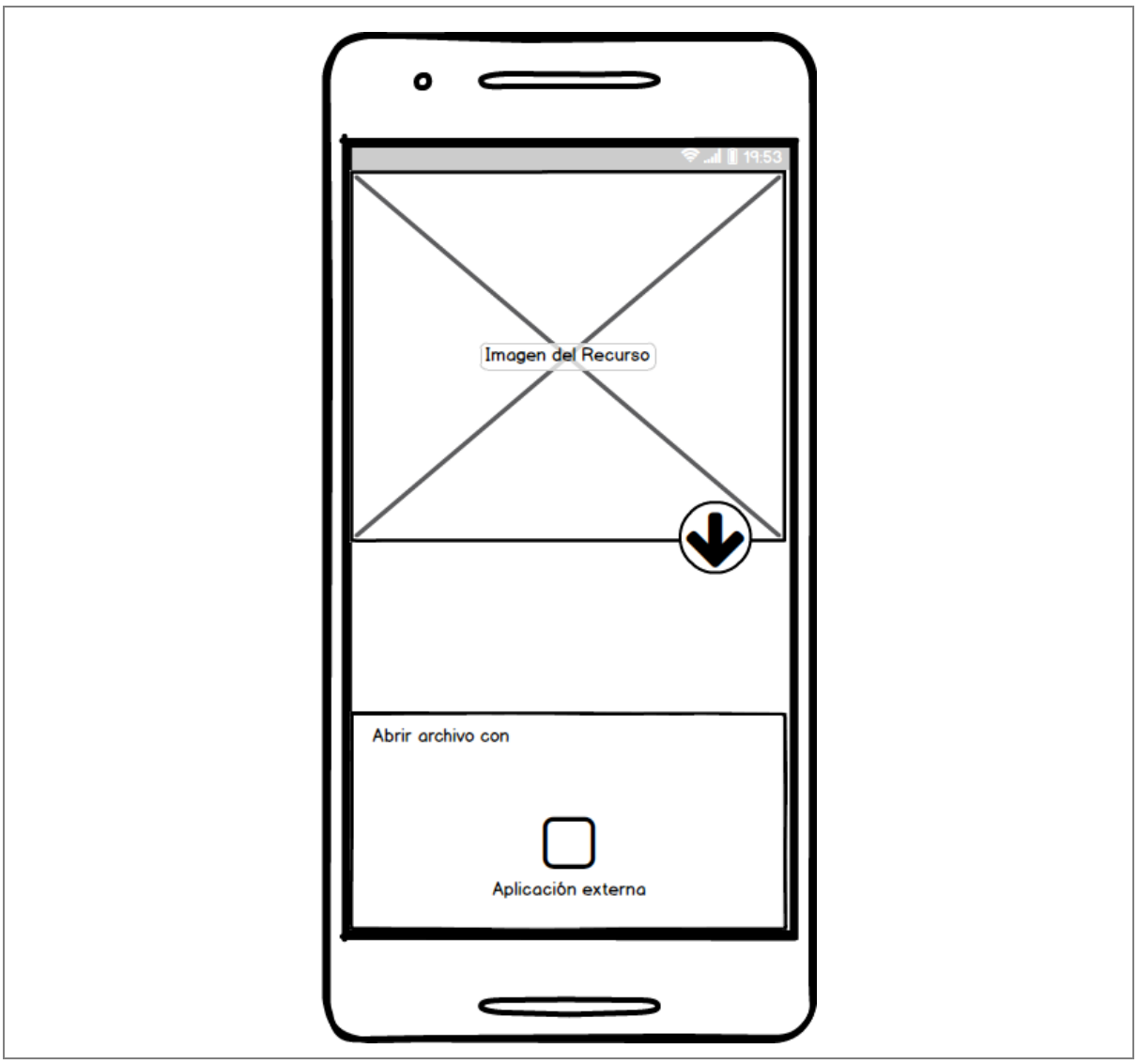
- El usuario abre la aplicación.

- Pulsar sobre el recurso que desee.

- Le mostrará los detalles, luego presionar en el icono que aparece en la esquina inferior derecha, de la imagen del libro.

Observaciones:

Prototipo de interfaz:



HU17. Requisito 17: Mostrar perfil de usuario.

Número: 17	**Nombre del Requisito:** Mostrar perfil de usuario
Programador: Diosbel E. Lima Sánchez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

El usuario en este caso podrá visualizar los detalles relacionados a su perfil e identidad.

1- Objetivo:

Mostrar perfil del usuario que inicio sesión en la aplicación.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para lograr esto hay que:

- Estar previamente autenticado.

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

- El usuario abre la aplicación.

- Después de haber iniciado sesión se mostrará un botón.

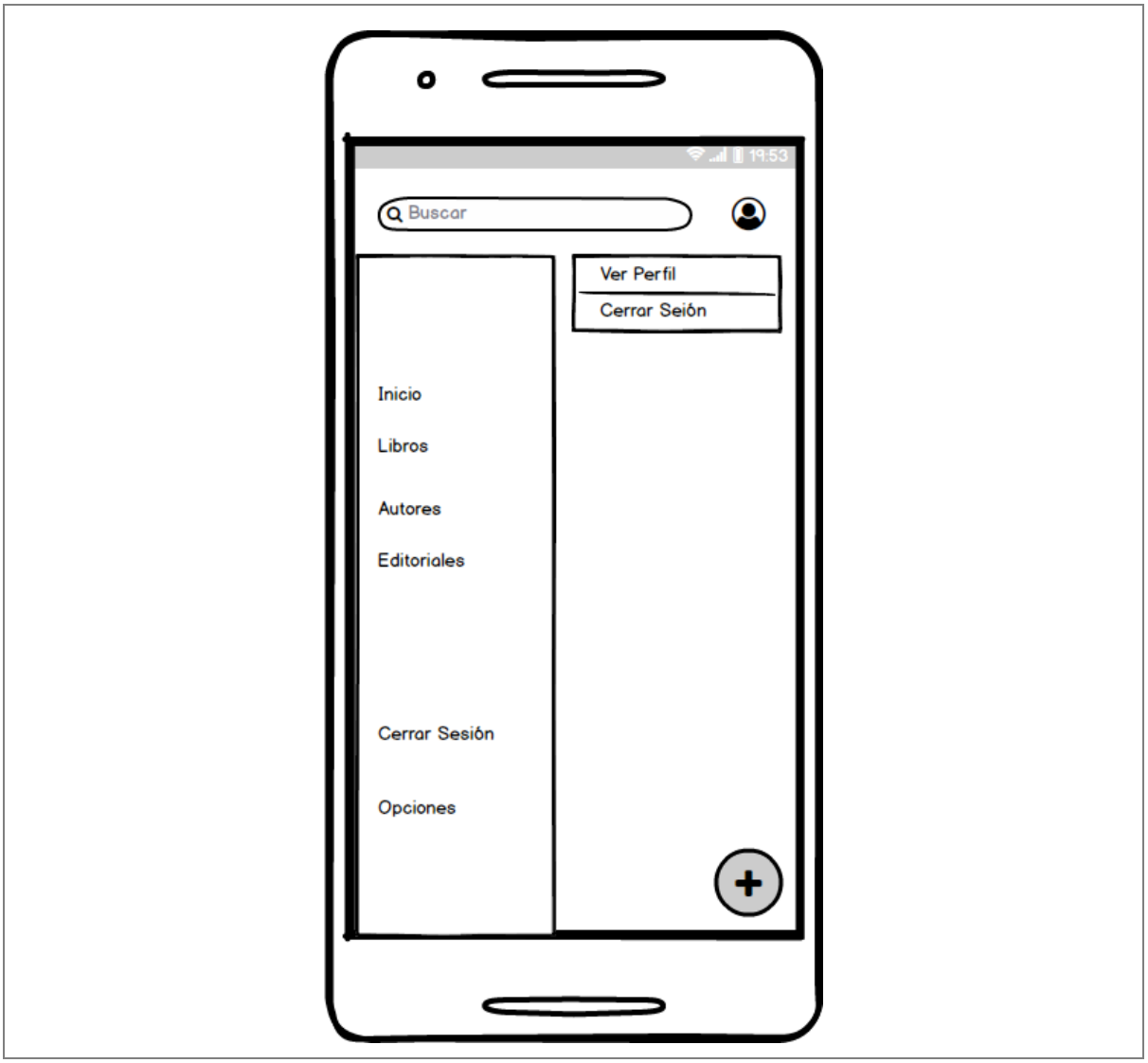
- Al pulsar sobre él, se deslizará un menú en el que se muestran las opciones de “Ver perfil” y “Cerrar sesión”.

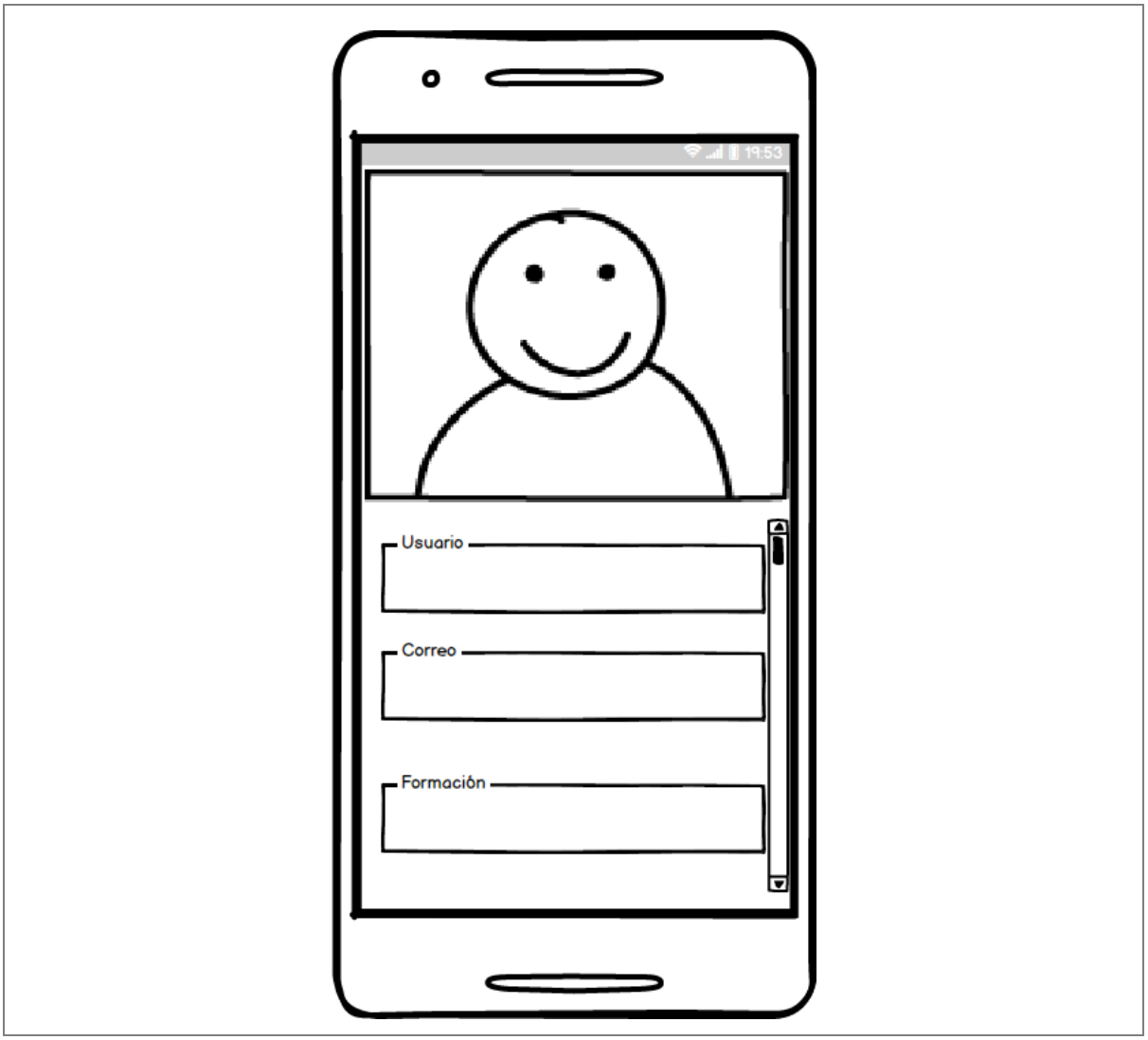
- Pulsar sobre “Ver perfil”

Observaciones:

Prototipo de interfaz:

Anexo I


19:53
Buscar
Ver Perfil
Cerrar Seiôn
Inicio
Libros
Autores
Editoriales
Cerrar Sesiôn
Opciones


19:53
Usuario
Correo
Formación

ANEXO II: TAREAS DE INGENIERÍA

Número tarea: 1	**Número historia:** 1	
Nombre Tarea: Pantalla de bienvenida en el inicio de la aplicación		
Tipo de tarea: Nuevo	**Prioridad:** Media	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Realizar un SplashActivity, utilizando un ProgressDialog para el proceso de cargar los datos.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 2	**Número historia:** 1	
Nombre Tarea: Conectar y solicitar datos		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Conectar y solicitar los datos del servicio web REST utilizando la librería OKHTTP.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 3	**Número historia:** 1
Nombre Tarea: Guardar los datos en la base de datos	
Tipo de tarea: Nuevo	**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez	

Descripción		
Guardar los datos recibidos del servicio web en el archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/01/2017	Definido	
15/01/2017	Implementación	
15/02/2017	Realizado	

Número tarea: 4	**Número historia:** 2
Nombre Tarea: Actualización de los datos manualmente	
Tipo de tarea: Nuevo	**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez	
Descripción	
Actualizar los datos que se encuentran el archivo datafv.db, por los nuevos y más recientes de la plataforma de recursos.	

Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 5	**Número historia:** 3
Nombre Tarea: Interfaz para la autenticación	
Tipo de tarea: Nuevo	**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez	
Descripción	
Realizar la interfaz de autenticación, mostrando los campos para ingresar usuario y contraseña	

Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	

15/03/2017	Realizado	

Número tarea: 6		**Número historia:** 3
Nombre Tarea: Inicia la sesión del usuario utilizando las credenciales		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Inicia la sesión con los datos ingresados por el usuario en la interfaz de autenticación		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 7		**Número historia:** 3
Nombre Tarea: Modificar elementos después de iniciada la sesión		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Modificar el botón de iniciar sesión por el de cerrar sesión en el NavigationDrawer y agregar un botón para el perfil de usuario en la barra de acciones (Toolbar).		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 8	**Número historia:** 4
Nombre Tarea: Cerrar sesión del usuario	

Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Implementar el cierre de sesión del servidor de la PRFV		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 9		**Número historia:** 4
Nombre Tarea: Modificar elementos después de cerrar sesión		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Modificar el botón de cerrar sesión por el de iniciar sesión en el NavigationDrawer y ocultar el perfil de usuario en la barra de acciones (Toolbar).		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 10		**Número historia:** 5
Nombre Tarea: Interfaz para los libros recomendados		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará el RecyclerView y CardView para mostrar los libros más vistos		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	

15/03/2017	Realizado	

Número tarea: 11		**Número historia:** 5
Nombre Tarea: Rellenar el adaptador de libros recomendados		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se rellena el adapter creado para mostrar los libros recomendados, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 12		**Número historia:** 6
Nombre Tarea: Interfaz para libros más vistos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará el RecyclerView y CardView para mostrar los libros más vistos		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 13		**Número historia:** 6
Nombre Tarea: Rellenar el adaptador de libros más vistos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		

Descripción		
Se rellena el *Adapter* creado para mostrar los libros más vistos, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 14	**Número historia:** 7	
Nombre Tarea: Interfaz para mostrar los libros		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará el RecyclerView y CardView para mostrar los libros de la PRFV		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 15	**Número historia:** 7	
Nombre Tarea: Rellenar el adaptador de libros		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se rellena el *Adapter* creado para mostrar los libros, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 16		**Número historia:** 8
Nombre Tarea: Interfaz para mostrar los autores		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará el RecyclerView y CardView para mostrar los autores de la PRFV		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 17		**Número historia:** 8
Nombre Tarea: Rellenar el adaptador de autores		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se rellena el *Adapter* creado para mostrar los autores, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 18		**Número historia:** 9
Nombre Tarea: Interfaz para mostrar las editoriales		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		

Se creará el *RecyclerView* y *CardView* para mostrar las editoriales de la PRFV		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 19		**Número historia:** 9
Nombre Tarea: Rellenar el adaptador de editoriales		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se rellena el *Adapter* creado para mostrar las editoriales, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 20		**Número historia:** 10
Nombre Tarea: Interfaz para mostrar los recursos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará el RecyclerView y CardView para mostrar los recursos disponibles de la PRFV		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 21		**Número historia:** 10
Nombre Tarea: Rellenar el adaptador de recursos		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se rellena el *Adapter* creado para mostrar los recursos, con los datos del archivo de base de datos datafv.db		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 22		**Número historia:** 11
Nombre Tarea: Interfaz para mostrar los detalles de un libro		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará la interfaz con todos los campos o atributos que puede tener un libro		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 23		**Número historia:** 11
Nombre Tarea: Mostrar los detalles de un libro		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		

Se cargan los datos del archivo datafv.db y se muestran en la interfaz		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 24	**Número historia:** 12	
Nombre Tarea: Interfaz para mostrar los detalles de un autor		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará la interfaz con todos los campos o atributos que puede tener un libro		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 25	**Número historia:** 12	
Nombre Tarea: Mostrar los detalles de un autor		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se cargan los datos del archivo datafv.db y se muestran en la interfaz		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 26	**Número historia:** 13

Nombre Tarea: Interfaz para mostrar los detalles de una editorial		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará la interfaz con todos los campos o atributos que puede tener un libro		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 27	**Número historia:** 13	
Nombre Tarea: Mostrar los detalles de una editorial		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se cargan los datos del archivo datafv.db y se muestran en la interfaz		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 28	**Número historia:** 14	
Nombre Tarea: Interfaz para mostrar los detalles de un recurso		
Tipo de tarea: Nuevo	**Prioridad:** Alta	
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se creará la interfaz con todos los campos o atributos que puede tener un recurso.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	

15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 30		**Número historia:** 14
Nombre Tarea: Mostrar los detalles de un recurso.		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Se cargan los datos del archivo datafv.db y se muestran en la interfaz.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 31		**Número historia:** 15
Nombre Tarea: Creación de un botón para visualizar libro.		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Creación del botón para visualizar el libro		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 32		**Número historia:** 15
Nombre Tarea: Abrir libro con aplicación externa.		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		

Descripción		
Con el uso de URI, abrir una aplicación externa que permita visualizar contenido en formato PDF		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 33	**Número historia:** 16
Nombre Tarea: Crear botón para descargar recurso	
Tipo de tarea: Nuevo	**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez	
Descripción	
Se creará el botón para descargar recurso, e implementará el onClickListener para descargar el recurso.	

Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

Número tarea: 34	**Número historia:** 16
Nombre Tarea: Descargar recurso en carpeta	
Tipo de tarea: Nuevo	**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez	
Descripción	
Implementación de un método que guarde el recurso dentro de una carpeta con el nombre de /download_fvmobile/.	

Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	

15/03/2017	Realizado	

Número tarea: 35		**Número historia:** 16
Nombre Tarea: Abrir recurso automáticamente		
Tipo de tarea: Nuevo		**Prioridad:** Alta
Programador responsable: Diosbel E. Lima Sánchez		
Descripción		
Abrir recurso automáticamente después haber sido descargado, mediante la utilización de URI en Android.		
Fecha	**Estado**	**Comentario**
15/02/2017	Definido	
15/02/2017	Implementación	
15/03/2017	Realizado	

ANEXO III: PRUEBAS DE ACEPTACIÓN

PA1. Requisito: Cargar y guardar los datos.

Caso de prueba de aceptación
ID: 1
Historia: Cargar y guardar los datos
Aprobada/ID Defecto:
Descripción: 4. La pantalla debe mostrarse como el diseño. 5. Conectarse al servidor para obtener los datos 6. Debe mostrar una alerta si no se pudo conectar con el servidor
Resultado Esperado: 4. Inspección visual 5. Mostrar progreso 6. Mostrar alerta en caso de no encontrar la conexión
Pasa: 4. Si, se muestra como el diseño. 5. Si, se muestra el progreso. 6. Si, se muestra la alerta.

PA2. Requisito: Actualizar datos manualmente

Caso de prueba de aceptación
ID: 2
Historia: Actualizar datos manualmente.

Aprobada/ID Defecto:
Descripción: 7. La pantalla debe mostrarse como el diseño. 8. Conectarse al servidor para obtener los datos. 9. Debe mostrar una alerta si no se pudo conectar con el servidor.
Resultado Esperado: 7. Inspección visual. 8. Mostrar progreso. 9. Mostrar alerta en caso de no encontrar la conexión.
Pasa: 7. Si, se muestra como el diseño 8. Si, se muestra el progreso 9. Si, se muestra la alerta

PA3. Requisito: Iniciar sesión

Caso de prueba de aceptación
ID: 3
Historia: Iniciar sesión.
Aprobada/ID Defecto:
Descripción: 10. La pantalla debe mostrarse como el diseño. 11. Conectarse al servidor para autenticar al usuario.

12. Debe mostrar una alerta si no se pudo conectar con el servidor.
Resultado Esperado: 10. Inspección visual. 11. Mostrar un progreso del intento de conexión. 12. Mostrar alerta en caso de no conexión.
Pasa: 10. Si, se muestra como el diseño. 11. Si, se muestra el mensaje. 12. Si, se muestra la alerta.

PA4. Requisito: Cerrar sesión.

Caso de prueba de aceptación
ID: 4
Historia: Cerrar sesión.
Aprobada/ID Defecto:
Descripción: 13. La pantalla debe mostrarse como el diseño. 14. Conectarse al servidor para des-autenticar al usuario. 15. Debe mostrar una alerta si no se pudo conectar con el servidor.
Resultado Esperado: 13. Inspección visual. 14. Mostrar un progreso del intento de conexión. 15. Mostrar alerta en caso de no conexión.
Pasa: 13. Si, se muestra como el diseño.

14. Si, se muestra el progreso. 15. Si, se muestra la alerta.

PA5. Requisito: Listar los libros recomendados.

Caso de prueba de aceptación
ID: 5
Historia: Listar los libros recomendados.
Aprobada/ID Defecto:
Descripción: 16. La pantalla debe mostrarse como el diseño. 17. Muestra los libros recomendados por la editorial.
Resultado Esperado: 16. Inspección visual. 17. Mostrar los libros recomendados.
Pasa: 16. Si, se muestra como el diseño. 17. Si, se muestran los libros.

PA6. Listar los libros más vistos.

Caso de prueba de aceptación
ID: 6
Historia: Listar los libros más vistos.
Aprobada/ID Defecto:
Descripción:

18. La pantalla debe mostrarse como el diseño. 19. Deben mostrarse los libros más vistos de la editorial.
Resultado Esperado: 18. Inspección visual. 19. Mostrar los libros más vistos.
Pasa: 18. Si, se muestra como el diseño. 19. Si, se muestran los libros.

PA7. Listar los libros.

Caso de prueba de aceptación
ID: 7
Historia: Listar los libros de la editorial.
Aprobada/ID Defecto:
Descripción: 20. La pantalla debe mostrarse como el diseño. 21. Deben mostrarse los libros de la editorial.
Resultado Esperado: 20. Inspección visual. 21. Mostrar todos los libros de la editorial.
Pasa: 20. Si, se muestra como el diseño. 21. Si, se muestran los libros.

PA8. Listar todos los autores.

Caso de prueba de aceptación
ID: 8
Historia: Listar los autores de la editorial.
Aprobada/ID Defecto:
Descripción: 22. La pantalla debe mostrarse como el diseño. 23. Deben mostrarse los autores de la editorial.
Resultado Esperado: 22. Inspección visual. 23. Mostrar todos los autores de la editorial.
Pasa: 22. Si, se muestra como el diseño. 23. Si, se muestran los autores.

PA9. Listar las editoriales.

Caso de prueba de aceptación
ID: 9
Historia: Listar las editoriales que pertenecen a la plataforma.
Aprobada/ID Defecto:
Descripción: 24. La pantalla debe mostrarse como el diseño. 25. Deben mostrarse las editoriales.
Resultado Esperado:

24. Inspección visual. 25. Mostrar todos las editoriales.
Pasa: 24. Si, se muestra como el diseño. 25. Si, se muestran las editoriales.

PA10. Requisito: Listar los recursos.

Caso de prueba de aceptación
ID: 10
Historia: Listar los recursos que pertenecen a la plataforma.
Aprobada/ID Defecto:
Descripción: 26. La pantalla debe mostrarse como el diseño. 27. Deben mostrarse los recursos.
Resultado Esperado: 26. Inspección visual. 27. Mostrar todos los recursos.
Pasa: 26. Si, se muestra como el diseño. 27. Si, se muestran las editoriales.

PA11. Requisito: Mostrar los detalles del libro.

Caso de prueba de aceptación
ID: 11

Historia: Mostrar los detalles del libro.
Aprobada/ID Defecto:
Descripción: 28. La pantalla debe mostrarse como el diseño. 29. Deben mostrarse los detalles de un libro en específico.
Resultado Esperado: 28. Inspección visual. 29. Mostrar los detalles del libro.
Pasa: 28. Si, se muestra como el diseño. 29. Si, se muestran todos los detalles.

PA12. Requisito: Mostrar los detalles del autor.

Caso de prueba de aceptación
ID: 12
Historia: Mostrar los detalles del autor.
Aprobada/ID Defecto:
Descripción: 30. La pantalla debe mostrarse como el diseño. 31. Deben mostrarse los detalles de un autor en específico.
Resultado Esperado: 30. Inspección visual. 31. Mostrar los detalles de autor.
Pasa:

30. Si, se muestra como el diseño.
31. Si, se muestran los detalles.

PA13. Requisito: Mostrar los detalles de la editorial.

Caso de prueba de aceptación
ID: 13
Historia: Mostrar los detalles de la editorial.
Aprobada/ID Defecto:
Descripción: 32. La pantalla debe mostrarse como el diseño. 33. Deben mostrarse los detalles de una editorial en específico.
Resultado Esperado: 32. Inspección visual. 33. Mostrar los detalles de la editorial.
Pasa: 32. Si, se muestra como el diseño. 33. Si, se muestran los detalles.

PA14. Requisito: Mostrar los detalles de recurso.

Caso de prueba de aceptación
ID: 14
Historia: Mostrar los detalles de recurso.
Aprobada/ID Defecto:
Descripción:

34. La pantalla debe mostrarse como el diseño. 35. Deben mostrarse los detalles de un recurso en específico.
Resultado Esperado: 34. Inspección visual. 35. Mostrar los detalles del recurso.
Pasa: 34. Si, se muestra como el diseño. 35. Si, se muestran los detalles.

PA15. Requisito: Visualizar libro.

Caso de prueba de aceptación
ID: 15
Historia: Visualizar Libro.
Aprobada/ID Defecto:
Descripción: 36. La pantalla debe mostrarse como el diseño. 37. Debe mostrar el libro en formato PDF.
Resultado Esperado: 36. Inspección visual. 37. Mostrar libro en formato PDF.
Pasa: 36. Si, se muestra como el diseño. 37. Si, se muestra el libro.

PA16. Requisito: Descargar recurso.

Caso de prueba de aceptación
ID: 16
Historia: Descargar recurso.
Aprobada/ID Defecto:
Descripción: 38. La pantalla debe mostrarse como el diseño. 39. Debe permitir descargar el recurso si este es libre y puede ser descargado.
Resultado Esperado: 38. Inspección visual. 39. Descargar el recurso del tipo especificado.
Pasa: 38. Si, se muestra como el diseño. 39. Si, se descarga el recurso.

PA17. Requisito: Mostrar perfil de usuario.

Caso de prueba de aceptación
ID: 17
Historia: Mostrar perfil de usuario.
Aprobada/ID Defecto:
Descripción: 40. La pantalla debe mostrarse como el diseño. 41. Debe mostrarse el perfil del usuario.

Resultado Esperado: 40. Inspección visual. 41. Mostrar el perfil de usuario.
Pasa: 40. Si, se muestra como el diseño. 41. Si, se muestra el perfil de usuario.