



UNIVERSIDAD SIMÓN BOLÍVAR  
DECANATO DE ESTUDIOS PROFESIONALES  
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

**DESARROLLO DE UNA APLICACIÓN MÓVIL DE CONTROL Y  
REPORTES DE GASTOS**

Por:  
SUSANA CHARARA CHARARA

Realizado con la asesoría de:  
Tutor Académico: PROF. XIOMARA CONTRERAS  
Tutor Industrial: LIC. LUIS AUGUSTO PEÑA PEREIRA

INFORME DE PASANTÍA LARGA  
Presentado ante la Ilustre Universidad Simón Bolívar  
como requisito parcial para optar al título de  
Ingeniero en Computación

**Sartenejas, SEPTIEMBRE de 2016**



UNIVERSIDAD SIMÓN BOLÍVAR  
DECANATO DE ESTUDIOS PROFESIONALES  
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

**DESARROLLO DE UNA APLICACIÓN MÓVIL DE CONTROL Y  
REPORTES DE GASTOS**

Por:  
SUSANA CHARARA CHARARA

Realizado con la asesoría de:  
Tutor Académico: PROF. XIOMARA CONTRERAS  
Tutor Industrial: LIC. LUIS AUGUSTO PEÑA PEREIRA

INFORME DE PASANTÍA LARGA  
Presentado ante la Ilustre Universidad Simón Bolívar  
como requisito parcial para optar al título de  
Ingeniero en Computación

**Sartenejas, SEPTIEMBRE de 2016**

## Resumen

# Introducción

En la actualidad, muchas empresas le ofrecen a sus empleados el beneficio de realizar gastos personales y posteriormente iniciar un proceso de reembolso. Este proceso puede tomar mucho tiempo y puede resultar difícil si se hace manualmente, pues implica que el trabajador deba guardar todas las facturas o recibos de los gastos implicados. También dificulta la organización de estos registros por parte de la empresa, porque al mantener todos los recibos en físico, estos se pueden extraviar.

Este problema llevó a la conclusión de que es necesario agilizar todo el proceso, y dada la facilidad de acceso que se tiene hoy en día a un dispositivo móvil inteligente, se decidió crear una aplicación que permita mantener el registro de gastos.

Actualmente existen aplicaciones que sirven para llevar un registro de gastos. Sin embargo, estas aplicaciones no satisfacen las necesidades de la empresa por diversas razones:

- En primer lugar, la empresa realiza el reembolso de gastos en ciertos rubros y para esto se debe especificar a qué categoría pertenece cada gasto. Las aplicaciones ya existentes pueden limitar el proceso al no contar con la posibilidad de poder asociar un gasto a un rubro cubierto por la empresa.
- Se desea crear archivos de reportes de los gastos registrados por los trabajadores. Se desea que estos reportes puedan ser personalizados y presenten una estructura particular.

- Se desea mantener toda la información referente a estos reportes de manera centralizada de manera que se pueda acceder a la misma de una manera más fácil.
- Si el proceso de reembolso sufre alguna modificación, se desea contar con una aplicación que tome en cuenta los nuevos cambios.

Por las razones expuestas anteriormente, se tomó la decisión de crear una aplicación móvil que permita llevar un registro de gastos. Esta aplicación debe contar con las siguientes funcionalidades:

- Registrar gastos con su fecha, monto, una breve descripción, fotos y categoría a la que pertenecen
- Organizar y consultar gastos con criterios de búsqueda pre-establecidos
- Crear reportes personalizados con los gastos y enviar dichos reportes a un servidor
- Enviar reportes mediante archivos con formato PDF a los supervisores
- Controlar el estado de aprobación de los reportes

En este informe se describe el proceso de desarrollo de una aplicación móvil para el registro de gastos.

El informe se estructura de la siguiente manera:

# Capítulo 1

## Entorno Empresarial

### 1.1. Descripción de la empresa

Digitalica Group, C.A. es una empresa que se dedica al desarrollo de soluciones móviles, aplicaciones empresariales y *middleware*. Se caracteriza por estar a la vanguardia, utilizando tecnologías innovadoras [1]. Sus principales servicios son:

- Consultoría en el área de desarrollo de aplicaciones móviles: Durante los últimos años de ha brindado apoyo en el área de soluciones móviles a empresas del sector educativo.
- Desarrollo de *software* y aplicaciones móviles para clientes. Se han desarrollado aplicaciones para medios de comunicación nacionales e internacionales, así como para empresas del sector educativo.
- Ofrecer *software* como un servicio: Actualmente, los productos que ofrece la empresa están enfocados en el área de noticias, medios de comunicación y reporte de gastos.

## 1.2. Misión

”Proveer soluciones y servicios de consultoría basados en tecnologías móviles, utilizando la mejores herramientas, metodologías y estándares de calidad en la industria” [1].

## 1.3. Visión

”Ser el líder en Latinoamérica en brindar soluciones innovadoras basadas en tecnologías móviles, con un equipo creativo con los conocimientos más sólidos en tecnologías de información para móviles, para competir en el mercado global” [1].

## 1.4. Estructura organizacional

La estructura organizacional de la empresa está comprendida principalmente por la Junta de Directores, como se muestra en la figura 1.1.

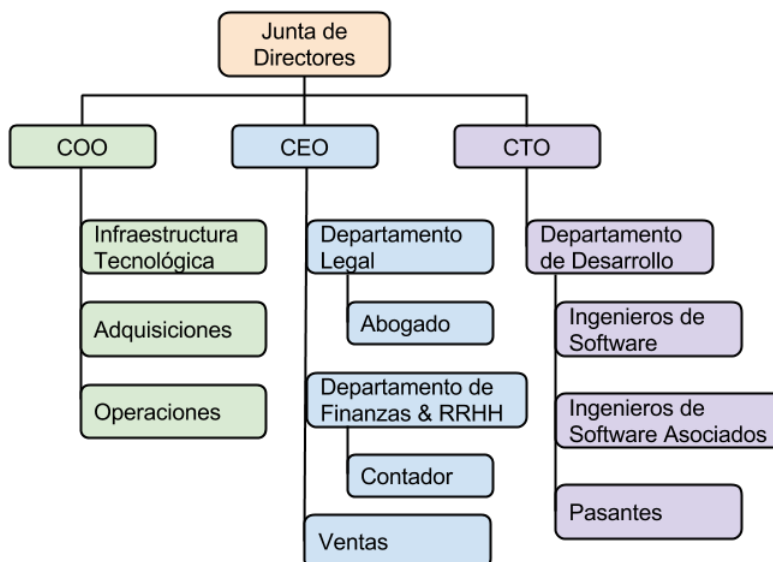


Figura 1.1: Estructura organizacional de la empresa

La Junta de Directores está compuesta por tres cargos: Director de Operaciones (COO por sus siglas en inglés), Director Ejecutivo (CEO por sus siglas en inglés) el Director de Tecnología (CTO por sus siglas en inglés).

El COO se encarga de gestionar las adquisiciones de la empresa y de mantener la infraestructura tecnológica.

El CEO se encarga de manejar la Dirección de Ventas, dirigir el Departamento Legal junto al abogado y dirigir el Departamento de Finanzas y Recursos Humanos junto al contador.

El CTO se encarga de dirigir el departamento de desarrollo, por lo que el equipo de desarrollo de *software* se comunica y le rinde cuentas a él.

## 1.5. Ubicación del pasante

Los pasantes se encuentran en el Departamento de Desarrollo, dirigido por el CTO, y donde se encuentran los especialistas de todas las áreas de desarrollo de *software*. Esto permite que los pasantes estén en constante contacto con personas que puedan asistirlos y solventar las dudas que puedan surgir durante el desarrollo.



# Capítulo 2

## Marco Teórico

En este capítulo se exponen los conceptos teóricos estudiados, tanto para entender las tecnologías utilizadas como para realizar el desarrollo de *software* requerido. Los conceptos estudiados formaron parte fundamental del proceso de diseño y desarrollo de la solución.

### 2.1. Patrones de diseño

Un patrón de diseño es una solución general y repetible a problemas que suelen presentarse en el proceso de diseño de software. Para que una solución pueda ser considerada como un patrón de diseño debe ser reutilizable, es decir, que se pueda aplicar a diferentes problemas de diseño en diferentes situaciones [2]. A continuación se describen los patrones utilizados en el diseño de la solución del proyecto.

#### 2.1.1. *Singleton*

El *singleton* es un patrón de diseño que asegura la existencia de una sola instancia de la clase que lo implementa [3]. Esto es útil cuando se necesita únicamente un objeto para coordinar las acciones en el sistema [4].

### 2.1.2. *Adapter*

Transforma la interfaz de una clase en otra interfaz que el cliente espera. Esto permite que una clase que no pueda utilizar la primera interfaz, sí pueda hacerlo a través de la otra [3].

### 2.1.3. *Facade*

Se utiliza para ofrecer una interfaz sencilla para operaciones complejas. Con este patrón se logra ofrecer una interfaz de alto nivel que hace que el sistema sea más fácil de usar para los clientes [3].

## 2.2. Patrones de arquitectura

Un patrón de arquitectura es una solución a problemas de arquitectura de *software*, permitiendo definir una estructura. La diferencia entre los patrones de diseño y los de arquitectura es que estos últimos tienen un nivel de abstracción mayor. [5]

### 2.2.1. Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de *software* que divide una aplicación en tres partes que están interconectadas: los datos y la lógica de negocio (modelo), la interfaz de usuario (vista) y el módulo encargado de gestionar las comunicaciones (controlador). Esto se utiliza para separar la representación de la información de la forma en que dicha información es presentada al usuario [6].

### 2.2.2. Modelo Vista Presentador (MVP)

Es una derivación del patrón Modelo Vista Controlador (MVC) que se utiliza generalmente para construir interfaces de usuario [7]. En este patrón, la interacción entre el modelo

y la vista se logra únicamente a través del presentador, mientras que en el MVC la vista en ocasiones puede comunicarse directamente con el modelo. Otra diferencia con el patrón MVC es que en este último las peticiones son recibidas por el controlador. Éste, se comunica con el modelo para pedir los datos y luego se encarga de mostrar la vista adecuada. En el patrón MVP las peticiones son recibidas por la vista y delegadas al presentador, que es quien se comunica con el modelo para obtener los datos [8].

## 2.3. Arquitectura cliente-servidor

Es un modelo de arquitectura de sistema distribuido, donde existe un conjunto de procesos que proporcionan servicios (servicios) a otros procesos (clientes). Por lo general, el servidor no conoce la identidad ni el número de sus clientes, pero el cliente sí conoce la identidad del servidor [9]. Dentro de esta arquitectura existen cuatro tipos [10]:

- Cliente activo, servidor pasivo: El cliente procesa toda la información.
- Cliente pasivo, servidor pasivo: Tanto el cliente como el servidor pasan información.
- Cliente pasivo, servidor activo: El servidor procesa toda la información y el cliente presenta los datos
- Cliente activo, servidor activo: Tanto el servidor como el cliente procesan la información.

## 2.4. Servicio web

Es un conjunto de funcionalidades diseñadas para permitir la comunicación entre diferentes dispositivos a través de una red, utilizando un conjunto de protocolos y estándares [11].

## **2.5. HTTP ((Hypertext Transfer Protocol))**

Es el protocolo de comunicación que permite la transferencia de recursos en la web. Un recurso es cualquier información que puede ser identificada por un URL. Existen diversos métodos que permiten realizar peticiones con el protocolo HTTP, entre los que están GET y POST [?].

### **2.5.1. HTTP GET**

Es el método más común de HTTP, que permite pedir datos de un recurso en específico [?].

### **2.5.2. HTTP POST**

Es uno de los diferentes métodos de peticiones que soporta el protocolo HTTP, que permite enviar datos a un recurso, los cuales deben ser procesados. Se requiere que los datos sean enviados dentro del cuerpo del mensaje [?].

Para enviar estos datos dentro de una petición POST, existen diferentes tipos de codificación. Entre estos tipos esta Multipart, que permite enviar mensajes con varias partes combinadas en un solo cuerpo [?].

# Capítulo 3

## Marco Tecnológico

En este capítulo se definen las diferentes herramientas utilizadas a lo largo de todo el proyecto. Dichas herramientas permitieron el desarrollo del *software* planteado como solución al problema.

### 3.1. Herramientas, entornos y lenguajes

#### 3.1.1. Android

Es un sistema operativo de código abiertos basado en el sistema operativo Linux, utilizado principalmente para dispositivos móviles [13]. Es la plataforma para la cual se desarrolló la aplicación. s

#### 3.1.2. Java

Es un lenguaje de programación de alto nivel orientado a objetos, que puede correr virtualmente en cualquier computadora [14]. Es el lenguaje en el que están desarrolladas la mayoría de las aplicaciones de Android [15]. Se utilizó tanto para la programación de la

aplicación móvil como del servidor.

### **3.1.3. MySQL**

Es un sistema de manejo de base de datos relacional basado en el Lenguaje de Consulta Estructurado (SQL por sus siglas en inglés) [16]. Se utilizó para la gestión de la base de datos del servidor.

### **3.1.4. SQLite**

Es un sistema de manejo de base de datos relacional basado en el Lenguaje de Consulta Estructurado (SQL por sus siglas en inglés) que forma parte del programa principal, en lugar de ser un proceso independiente como ocurre con los sistemas de gestión de base de datos cliente-servidor [17].

### **3.1.5. Maven**

Es una herramienta que se permite compilar y manejar proyectos de *software* basados en Java. Se encarga de conectar las dependencias de los paquetes [18].

### **3.1.6. Jetty**

Es un servidor HTTP que puede funcionar independientemente por sus propios medios, o que puede correr dentro de otra aplicación [19].

### **3.1.7. Android Studio**

Es el entorno de desarrollo integrado (IDE por su siglas en inglés) oficial para el desarrollo de aplicaciones nativas para Android [20].

### 3.1.8. Eclipse

Es un entorno desarrollo integrado (IDE por sus siglas en inglés) utilizado principalmente para desarrollar aplicaciones en Java [21].

### 3.1.9. Git

Es un sistema de control de versiones distribuido (DVCS por sus siglas en inglés). Es una herramienta que permite gestionar las distintas versiones de un proyecto de *software* [22].

## 3.2. *Frameworks y librerías*

### 3.2.1. AppFuse

Es un *framework* utilizado para la creación de aplicaciones web en la máquina virtual de Java (JVM por sus siglas en inglés). Dentro de AppFuse, se pueden integrar otras tecnologías como Bootstrap, Maven, Hibernate, Spring, Tapestry, etc [23].

### 3.2.2. Hibernate

Es una herramienta de mapeo objeto-relacional (ORM por sus siglas en inglés) para Java que permite el mapeo de objetos de Java a bases de datos relacionales [24].

### 3.2.3. Spring

Es un *framework* para el desarrollo de aplicaciones en Java.

### 3.2.4. Tapestry

Es un *framework* para el desarrollo de aplicaciones web en Java, Groovy o Scala. [25].

### **3.2.5. Gson**

Es una librería de Java que permite la conversión de objetos a JSON y viceversa [26] .

### **3.2.6. Retrofit**

Es una librería de Android que permite realizar peticiones HTTP mediante una interfaz de Java [27].



# Capítulo 4

## Marco Metodológico

A continuación se describe el procedimiento seguido para el desarrollo del proyecto. Se decidió utilizar el marco de trabajo Scrum, dado que éste es utilizado por la empresa para el desarrollo de *software*.

### 4.1. Scrum

Scrum es un marco de trabajo que se basa en el desarrollo iterativo e incremental de un producto, en lugar del modelo clásico de planificación y ejecución completa [30]. Se caracteriza por ser una metodología ligera, fácil de entender y difícil de dominar, que permite entregar incrementos de producto potencialmente productivos [29].

#### 4.1.1. Roles

En Scrum el desarrollo se realiza por uno o más equipos de trabajo dentro de los cuales existen tres roles: dueño del producto, facilitador de Scrum y el equipo de desarrollo [30].

### **Dueño del producto (*Product owner*)**

Es el representante de los clientes. Dentro del equipo de Scrum, es el líder principal del producto y el responsable de decidir qué funcionalidades serán desarrolladas y la prioridad que tendrá cada una de ellas. Debe comunicar al resto de los involucrados en el proyecto una visión clara de lo que se quiere lograr. Tiene la obligación de asegurar que siempre se entregue un producto con el máximo de valor, por lo que debe colaborar con el resto del equipo para responder cualquier duda que surja [30]. Este rol fue asumido por el Ing. Chi Wang Zhong.

### **Facilitador de Scrum (*Scrum master*)**

Actúa como facilitador tanto para el dueño del producto como para el equipo de desarrollo. Es el encargado de ayudar al resto del equipo a entender y cumplir con los principios y prácticas de Scrum. También tiene la responsabilidad de eliminar cualquier impedimento que el equipo no sea capaz de resolver y que afecte su productividad [30]. Este rol fue asumido por el Lic. Luis Augusto Peña.

### **Equipo de desarrollo**

Es el encargado de desarrollar el producto. Es un equipo que está compuesto por arquitectos, programadores, probadores, administradores de base de datos, diseñadores de interfaces, entre otros. Son los responsables de diseñar, desarrollar y probar el producto [30]. El equipo de desarrollo estuvo integrado únicamente por la pasante Susana Charara.

#### **4.1.2. Actividades**

En Scrum, el trabajo se desarrolla en iteraciones de una duración máxima de un mes, llamadas *sprints*. Al final de cada iteración, se debe haber desarrollado una parte del producto final, la cual debe ser completamente funcional. Dentro de cada iteración existe una

serie de eventos o actividades que se llevan a cabo: planeación de la iteración, ejecución de la iteración, reuniones diarias, revisión de la iteración y retrospectiva de la iteración [30].

### **Planeación de la iteración (*Sprint planning*)**

Para determinar qué funcionalidades del producto final son las más importantes y próximas a desarrollar, el equipo de trabajo (dueño del producto, facilitador de Scrum y el equipo de desarrollo) realizan una reunión llamada *sprint planning* o planeación de la iteración [30].

Durante la reunión, el dueño del producto y el equipo de desarrollo establecen una meta que debe ser cumplida para el final de la iteración. De acuerdo a esta meta, el equipo de desarrollo decide de una manera realista qué incrementos del producto final pueden entregarse al terminar la iteración [30].

### **Ejecución de la iteración (*Sprint execution*)**

Luego de la ejecución de la iteración, el equipo de desarrollo desarrolla todas las tareas acordadas en la reunión. Esto es lo que se conoce como la ejecución de la iteración [30].

### **Reunión diaria (*Daily scrum*)**

Cada día dentro de la ejecución de la iteración, los miembros del equipo de desarrollo se reúnen durante un máximo de 15 minutos con el fin de informar qué se hizo el día anterior, qué se tiene planificado realizar el presente día y qué impedimentos se han presentado durante el desarrollo de su trabajo [30].

### **Revisión de la iteración (*Sprint review*)**

Al final de cada iteración, ocurre un evento que se conoce como *sprint review* o revisión de la iteración. El objetivo de esta actividad es revisar el incremento y realizar las adaptaciones necesarias al producto [30].

## **Retrospectiva de la iteración (*Sprint retrospective*)**

Esta actividad ocurre generalmente luego del *sprint review* y antes del próximo *sprint planning*. Es una oportunidad para que todo el equipo se reúna para discutir qué ha funcionado y qué no acerca de Scrum. Al final de la retrospectiva, el equipo de Scrum habrá discutido qué acciones deberán tomarse para mejorar la dinámica en las próximas iteraciones [31].

### **4.1.3. Artefactos**

Dentro de Scrum existen dos herramientas o artefactos que permiten mantener un seguimiento del proyecto: el documento de historias de usuario y la lista de tareas de la iteración. [31].

#### **Documento de historias de usuario (*Product backlog*)**

Es una lista de los requerimientos funcionales del producto ordenados según su importancia. El dueño del producto es el responsable de definir qué elementos serán incluidos en esta lista y de colocarlos según su prioridad, de manera que los elementos de mayor valor o prioridad aparezcan al principio de la lista, y los de menos valor al final de la misma [31].

#### **Lista de tareas de la iteración (*Sprint backlog*)**

Es una lista donde se presenta un subconjunto de los elementos del *product backlog* divididos en tareas más pequeñas [31].

# Capítulo 5

## Desarrollo de la aplicación

En este capítulo se describe el trabajo realizado para el desarrollo de la aplicación. Este proceso se dividió en tres fases: investigación, diseño y desarrollo del sistema que incluye tanto la aplicación móvil como el componente servidor.

### 5.1. Investigación

El objetivo de esta primera fase era familiarizarse con el entorno de trabajo.

En primer lugar se estudiaron diversos patrones de diseño. Para esto, se investigó acerca de patrones ampliamente utilizados en el desarrollo de aplicaciones para Android y en la programación orientada a objetos en general. Esto permitió estructurar la aplicación de una manera más ordenada y entendible, de manera que se pueda extender fácilmente.

Luego de esto, se investigó acerca del desarrollo de aplicaciones para Android. Se aprendió a trabajar con el IDE Android Studio, así como los componentes y herramientas para el desarrollo bajo el sistema operativo.

Por último, se realizó una aplicación de prueba para poner en práctica algunos conceptos básicos en el desarrollo de aplicaciones para Android. Con esto se aprendió la estructura

general de una aplicación, así como el flujo de ejecución.

## 5.2. Análisis y diseño de la solución

En esta fase tuvo lugar dos etapas fundamentales para el desarrollo: análisis del problema y diseño de la solución.

Para comprender mejor el problema presentado, se realizó una reunión con el dueño del producto (*product owner*). En esta reunión se realizó el primer levantamiento y refinamiento de requerimientos. A partir de estos requerimientos, se diseñó una arquitectura base que diera solución al problema.

### 5.2.1. Análisis del problema

Como se mencionó anteriormente, se necesita una aplicación móvil que permita a los trabajadores de la compañía registrar gastos personales que posteriormente serán reembolsados. El reporte de estos gastos debe ser enviado a un servidor web. También se requiere una aplicación web que permita aprobar o rechazar los reportes enviados.

La aplicación móvil debe permitir la creación de cuentas con diferentes monedas. Para cada una de estas cuentas, se pueden registrar gastos, a los cuales se puede asociar un monto y una fecha. Además, se pueden tomar fotos de las facturas o recibos de dichos gastos.

Dado que la empresa realiza el reembolso de gastos en determinadas áreas, es necesario que estos gastos puedan asociarse a categorías que los identifiquen. Además, se debe permitir la creación de nuevas categorías en caso de que en algún momento la empresa decida incluir nuevos rubros.

También se quiere poder realizar consultas de los gastos registrados en un rango de fechas y generar un reporte de dichos gastos.

Por último, se quiere que los supervisores de la compañía puedan tener acceso a los reportes de gastos mensuales de los trabajadores. Los supervisores deben poder aprobar o rechazar dichos reportes.

### 5.2.2. Diseño de la solución

#### Arquitectura base

Dados los requerimientos anteriores, se diseñó una arquitectura base para solucionar el problema. Se dividió el sistema según sus componentes: servidor web, dispositivos móviles y cliente web. Como se muestra en la figura 5.1, es una arquitectura cliente-servidor, donde el componente móvil y la aplicación web juegan el papel de cliente. En este caso, la aplicación móvil actúa como un cliente activo, pues es quien manejará los datos relevantes. El servidor solo recibirá esta información, sin poder modificarla. Esto corresponde a una arquitectura del tipo cliente activo-servidor pasivo, descrita en el capítulo 2.

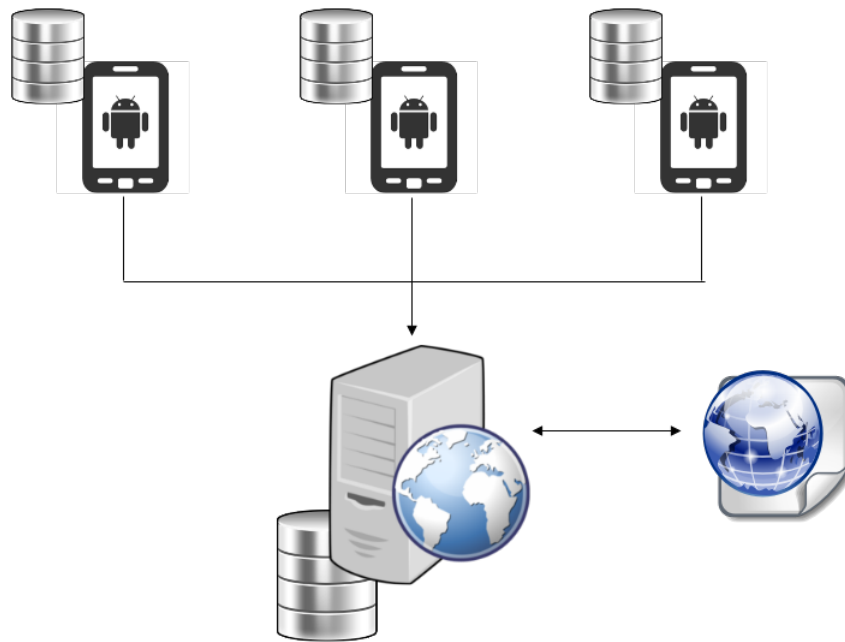


Figura 5.1: Arquitectura base de la solución

Por medio de la aplicación móvil se podrá hacer el registro de los gastos personales. El servidor permitirá la revisión de los reportes de gastos, así como su aprobación o rechazo.

Toda la información relacionada a los gastos (fecha, monto, categorías, fotos) será creada y guardada localmente en cada dispositivo móvil. Cada dispositivo podrá enviar al servidor su reporte de gastos en forma de un archivo PDF. La comunicación entre la aplicación móvil y el servidor se hace por medio de servicios web que presta este último.

Estos reportes deberán ser guardados en la base de datos del servidor. Además, a través de una aplicación web, se podrá aprobar o rechazar los reportes recibidos.

### Arquitectura de *software*

Para los componentes mencionados anteriormente, se definieron los patrón de arquitectura de *software* sobre los cuales se crearon. Para el servidor se utilizó el patrón Modelo Vista Controlador (MVC) y para la aplicación móvil el Modelo Vista Presentador (MVP).

### Estructura de datos

Para dar lugar al desarrollo del sistema, fue necesario diseñar un modelo de datos que permita mantener la información necesaria en el dispositivo móvil, y otro para mantener información en el servidor.

En la figura 5.2 se puede observar el diagrama de clases que representa el modelo de datos de la aplicación móvil.

A continuación se describirán cada una de estas clases, así como sus atributos:

- ***Account***: representa un conjunto de registros de movimientos de dinero. Almacena información de la moneda en la que estarán los montos de los registros asociados a la misma.



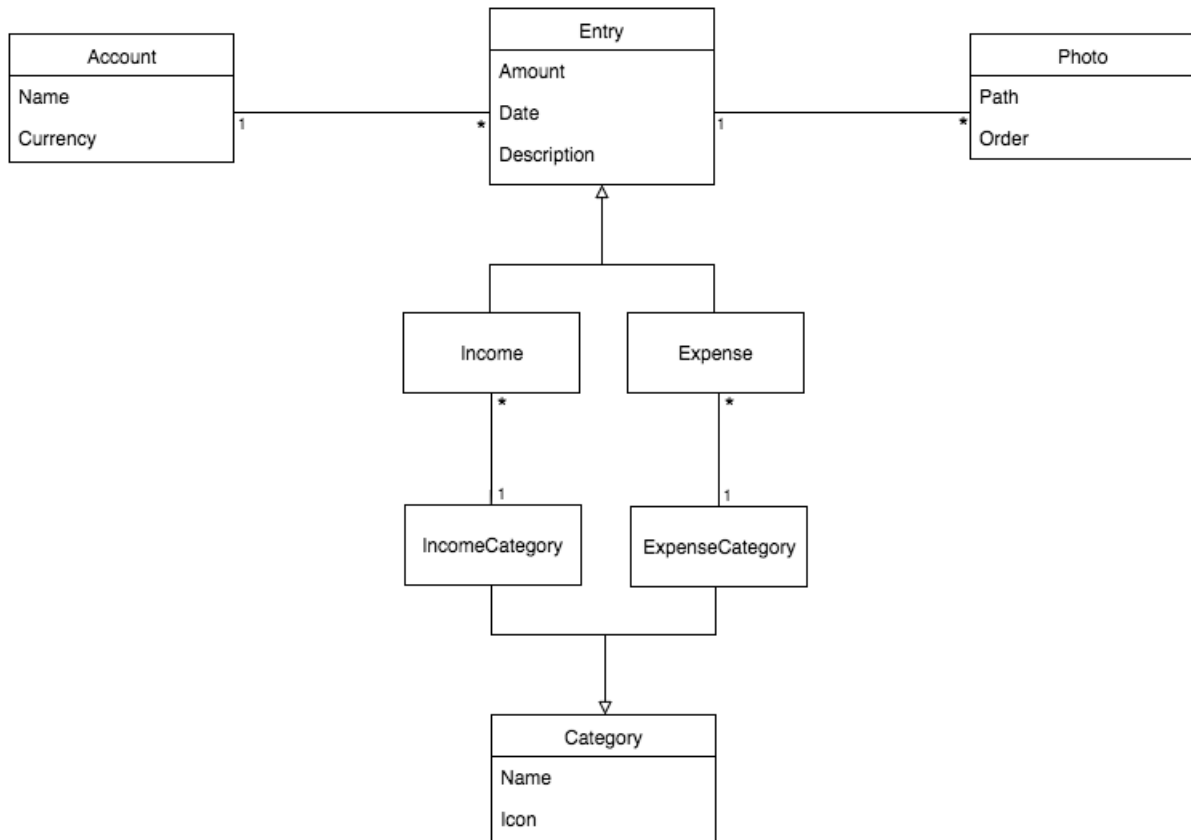


Figura 5.2: Diagrama de clases de la aplicación móvil

- **Entry**: representa un registro dentro de una cuenta, y puede ser de dos tipos: ingreso o gasto. Almacena información del monto, fecha y descripción.
- **Category**: representa las clases a las que puede pertenecer un *entry*. Puede ser de dos tipos: categoría de ingresos o categoría de gastos. Almacena información general (nombre) y un ícono que la representa.
- **Photo**: representa las fotos asociadas a un *entry*. Almacena información del *entry* al que pertenece, y la ruta del archivo dentro del dispositivo móvil.

Por otra parte, en la figura 5.3 se puede observar la estructura de datos del servidor. En este caso, se tienen dos clases: *user* y *report*.

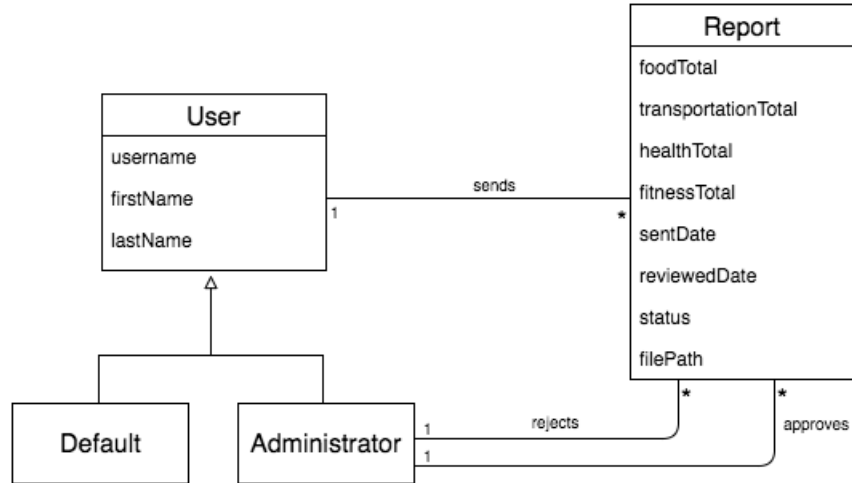


Figura 5.3: Diagrama de clases del servidor

### 5.3. Desarrollo

Durante esta fase se implementó progresivamente la versión alfa del prototipo funcional.

Se creó un componente móvil y un servidor para dar solución al problema de registro y control de gastos. Para cumplir con los requerimientos del componente móvil, se desarrolló una aplicación para el sistema operativo Android, y se trabajó con el IDE Android Studio.

Se implementaron vistas para crear, editar y eliminar gastos. Para estos gastos, se debe guardar su monto, fecha, descripción, fotos de recibos y categoría a la que pertenecen. Posteriormente, se decidió extender la funcionalidad de creación de gastos para permitir también el manejo de ingresos.

También se crearon vistas para el manejo de categorías, que incluyen la creación, edición y eliminación de las mismas.

Luego de la creación de las principales funcionalidades del sistema, mencionadas en los párrafos anteriores, se introdujo el concepto de cuentas”. Hasta el momento, los gastos estaban asociados a una única cuenta por defecto. Por esta razón, también se creó una funcionalidad para agregar nuevas cuentas, y poder crear nuevos gastos dentro de dichas cuentas.

Como se explicó en la sección anterior, es necesario guardar toda esta información en la base de datos local de cada dispositivo. Para ello, se utilizó la librería SQLite.

Además, también se crearon dos vistas para reportes de gastos: una para reportes de balance general, y otra para reporte de gastos por categorías. La vista de reporte de balance general muestra una lista de gastos e ingresos filtrados por fecha. La vista de reporte por categorías muestra una lista de categorías con la suma total de los gastos e ingresos de dichas categorías, filtradas por fecha.

Para mantener la lógica separada de la interfaz gráfica, se utilizó el patrón de arquitectura MVP o Modelo Vista Presentador, descrito en el capítulo 2. De esta manera, toda acción en la vista que requiera realizar operaciones sobre la base de datos, se realizó por medio de una capa intermedia: el presentador.

En la figura 5.4 se muestran capturas de pantallas de algunas de las funcionalidades de la aplicación móvil

Por otra parte, se creó un servidor al que se pueda enviar información desde la aplicación móvil.

Para permitir el acceso a la aplicación móvil, se creó un servicio web que permita hacer la autenticación del usuario. La aplicación envía al servidor un JSON con los datos del usuario (nombre de usuario y contraseña) por medio de una petición HTTP POST.

Se creó un servicio web que recibe un archivo PDF que contiene el reporte de gastos. Además del archivo, recibe la suma de gastos por cada categoría y el total de gastos. Estos datos se enviaron por medio de una petición HTTP Post Multipart, compuesta por un archivo y un JSON. Para la conversión de objetos de Java a JSON, y viceversa, se utilizó la librería GSON tanto en el servidor como en la aplicación móvil. Para persistir esta información, se utilizó Hibernate y MySQL como manejador de base de datos.

También se creó una aplicación web que permita aprobar o rechazar los reportes enviados

al servidor. Además de esto, la herramienta utilizada para el desarrollo del servidor trae consigo algunas funcionalidades por defecto: iniciar sesión y el manejo de usuarios (agregar, modificar y eliminar usuarios).

En la figura 5.5 se muestran capturas de pantallas de las funcionalidades del servidor.

El desarrollo del componente servidor se realizó con el IDE Eclipse. Se utilizó el *framework* AppFuse, que integra a su vez otros *frameworks* que facilitan el desarrollo web. En este caso, se trabajó en conjunto con Spring y Tapestry.

Se utilizó Spring para la implementación de los servicios web que permiten recibir datos desde la aplicación móvil. Estos servicios se implementaron mediante una clase *Manager*, la cual accede a los POJO's mediante un DAO. Además, Spring también cuenta con el *framework* Hibernate para la persistencia de los datos, utilizando como manejador de base de datos MySQL.

Para crear la aplicación web, se utilizó Tapestry. Con Jetty se proveen los servicios, mediante el protocolo HTTP, que permiten la comunicación con la aplicación web y la aplicación móvil.

Por otra parte, a través de la librería Retrofit, la aplicación se comunica con el servidor web utilizando el protocolo HTTP. La comunicación con la base de datos del dispositivo y con los servicios web se realizan en la capa del presentador. Por su parte, en la vista se maneja toda la interfaz gráfica a través de la cual el usuario interactúa con la aplicación, y toda la lógica necesaria para la comunicación con la capa de modelo y el presentador.

El proyecto se desarrolló bajo el marco de trabajo de *Scrum*, descrito en el capítulo 1. Durante la ejecución de las iteraciones, se implementaron las funcionalidades planificadas, así como las pruebas necesarias para validarlas.

A continuación se presentarán las iteraciones (*sprints*), durante las cuales se desarrollaron las funcionalidades descritas en los párrafos anteriores. Para cada iteración, se mencionarán

sus objetivos, resultados y una breve descripción de las actividades realizadas.

### **5.3.1. Iteración 1**

#### **Objetivos**

- Mostrar información del balance de una cuenta
- Permitir la creación de un nuevo gasto

#### **Resultados**

- Se creó una vista para mostrar el total de ingresos,gastos y balance general de una cuenta.
- Se creó una vista para permitir la creación de un nuevo gasto.

#### **Actividades**

En la primera parte de la iteración se implementó el modelo de datos del dispositivo móvil.

Se creó la vista principal de la aplicación móvil, donde se muestra el total de ingresos y el de gastos, así como el balance general. Para esto, fue necesario crear consultas a la base de datos que permitieran obtener el total de ingresos/gastos de una cuenta. También se realizó una consulta que permite obtener el balance total de una cuenta.

Por otro lado se implementó una vista que permite crear un nuevo gasto, con su fecha, monto y descripción.

Como se expuso anteriormente, la comunicación entre el modelo y la vista se hace a través de una capa intermedia: el presentador. Por esta razón, también fue necesaria la creación de un presentador para cada vista que permita hacer consultas a la base de datos.

### 5.3.2. Iteración 2

#### Objetivos

- Mostrar la lista de categorías
- Permitir la creación de un nuevo ingreso
- Permitir guardar fotos de un *entry*
- Asociar un ingreso/gasto a una categoría

#### Resultados

- Se creó una vista para mostrar la lista de categorías a las que puede pertenecer un gasto.
- Se adaptó y reutilizó la vista existente para crear un gasto para permitir la creación de un ingreso
- Se agregó la funcionalidad para tomar fotos relacionadas a un ingreso/gasto
- Se agregó la funcionalidad para poder asociar un ingreso/gasto a una categoría existente

#### Actividades

En esta iteración se creó la vista para mostrar las categorías presentes en el dispositivo.

Se creó una consulta para guardar en la base de datos una nueva categoría con su nombre y un ícono que la represente. Además, se creó una consulta para guardar en la base de datos una lista de categorías por defecto.

También se adaptó la vista existente para la creación de un gasto, de manera que se pueda reutilizar esta vista para la creación de un ingreso.

Por otra parte, se implementó la funcionalidad para tomar y guardar fotos asociadas a un ingreso/gasto. Para esto se tuvo que crear un método que permita abrir la aplicación de la cámara del dispositivo desde la aplicación móvil. Luego, se tuvo que adaptar la vista para la creación de un ingreso/gasto para mostrar las miniaturas de las fotos tomadas. Se puso un límite por parte de la empresa para que el máximo de fotos que se puedan guardar por ingreso/gasto sean 3.

Por último, se modificó la vista para la creación de un ingreso/gasto para que se permitiera asociar una categoría. Se tenía como requerimiento guardar las cuatro categorías que se utilizaron más recientemente. Para esto, se tuvo que crear los métodos necesarios para persistir en el dispositivo móvil las categorías más recientes. Dado que esta información no tiene una estructura compleja, se decidió utilizar un objeto que provee la plataforma de Android, llamado *SharedPreferences*, que permite guardar un conjunto de pares clave-valor. En este caso, se guardó una lista con las categorías más recientes.

### 5.3.3. Iteración 3

#### Objetivos

- Mostrar la lista de ingresos/gastos del mes actual
- Mostrar los detalles de los ingresos/gastos existentes
- Permitir editar y borrar un ingreso/gasto existente
- Implementar una calculadora para guardar el monto de un ingreso/gasto

#### Resultados

- Se creó la vista para mostrar una lista con los gastos y otra con los ingresos del mes actual.

- Se agregó la funcionalidad para ver los detalles de un ingreso/gasto ya existente, y poder editarlo
- Se agregó la funcionalidad para eliminar ingresos/gastos
- Se creó la interfaz para usar la calculadora que permita ingresar el monto de un ingreso/gasto

## Actividades

En primer lugar, se adaptó la vista principal de la aplicación para mostrar el total de ingresos y gastos únicamente durante el mes actual. Para esto, se tuvo que modificar la consulta a la base de datos para obtener el total de ingresos/gastos dado un rango de fecha.

Se implementó la funcionalidad para navegar a la lista de ingresos o de gastos del mes desde la vista principal de la aplicación. Para esto, se creó una vista general donde se pueda mostrar una lista de ingresos/gastos, con su descripción (o categoría, en caso de que no tenga descripción), su fecha y su monto.

Por otra parte, se creó la funcionalidad para eliminar ingresos/gastos existentes desde la vista mencionada en el párrafo anterior.

También se creó la funcionalidad para editar un ingreso/gasto existente. Para esto no fue necesario crear una vista nueva, sino que se adaptó la que se tenía para la creación de un nuevo ingreso/gasto, de manera que se muestre la información relacionada a dicho *entry* (fecha, monto, categoría, descripción y fotos).

Por último, se tenía como requerimiento mostrar una calculadora para ingresar el monto de un ingreso/gasto, en lugar de un teclado numérico común. Por esta razón, se tuvo que modificar la vista de creación de un ingreso/gasto para incluir la nueva funcionalidad.

Para este último requerimiento fue necesario crear un teclado virtual personalizado, pues el teclado numérico del dispositivo no incluye los operadores matemáticos básicos. Se creó el



formato del teclado, el cual incluye las teclas presentes en una calculadora básica. También se tuvo que manejar el uso de las teclas: detectar qué tecla fue presionada y realizar una acción en base a esta. Se utilizó una librería para evaluar fórmulas dada una cadena de caracteres.

Para la calculadora, se tenía que mostrar el símbolo decimal de acuerdo con el país para el cual está configurado el teléfono, ya que en algunos países se usa la coma (,) como separador y en otros el punto (.). Por esta razón, se tuvo que obtener el país de configuración del teléfono para decidir qué símbolo decimal mostrar.

#### **5.3.4. Iteración 4**

##### **Objetivos**

- Permitir el manejo de nuevas cuentas

##### **Resultados**

- Se creó una vista para crear una nueva cuenta
- Se implementó la funcionalidad para listar las cuentas del usuario
- Se implementó la funcionalidad para cambiar de cuenta
- Se implementó la funcionalidad para editar el nombre de una cuenta existente
- Se implementó la funcionalidad para eliminar cuentas existentes
- Se implementó la funcionalidad para cambiar el estado de una cuenta: activa o archivada
- Se implementó la funcionalidad para ver la lista de ingresos/gastos de una cuenta desde su creación hasta la fecha actual

## Actividades

En la ejecución de las iteraciones anteriores se estuvo trabajando con una sola cuenta creada por defecto en la aplicación. Para esta iteración, se decidió agregar la funcionalidad para poder manejar nuevas cuentas.

Para esto, se creó la vista que permite agregar una nueva cuenta, con su nombre y la moneda en la que van a estar sus ingresos/gastos.

Se creó una vista para mostrar la lista de cuentas guardadas. Se implementó la funcionalidad para cambiar el estado de una cuenta: activa o archivada. La lista de cuentas se muestra según su estado, es decir, se muestra una lista para la lista de cuentas activas y una para las cuentas archivadas.

Además, se implementaron las funcionalidades para editar el nombre de una cuenta y eliminar cuentas ya existentes.

También se creó la funcionalidad para cambiar la cuenta que se está mostrando actualmente en el dispositivo.

Por último, se adapto la vista existente para mostrar la lista de ingresos/gastos, de manera que se pueda mostrar todos los ingresos/gastos (en una misma lista) asociados a la cuenta que se muestra actualmente.

### 5.3.5. Iteración 5

#### Objetivos

- Permitir el manejo de las fotos de un ingreso/gasto
- Permitir el manejo de las categorías

## Resultados

- Se agregó la funcionalidad para ver las fotos de un ingreso/gasto
- Se agregó la funcionalidad para borrar las fotos de un ingreso/gasto
- Se agregó la funcionalidad para cambiar la ubicación en el dispositivo de las fotos tomadas de un ingreso/gasto
- Se agregó la funcionalidad para crear una categoría
- Se agregó la funcionalidad para eliminar una categoría existente
- Se agregó la funcionalidad para editar una categoría existentes

## Actividades

La primera parte de la iteración se dedicó al manejo de las fotos. En primer lugar se creó la vista para poder ver en pantalla completa las fotos de un ingreso/gasto (esto se hace desde la vista de creación de un ingreso/gasto). También se implementó la funcionalidad para eliminar una foto.

Por otra parte, se creó la vista de configuraciones de la aplicación. En esta vista se agregó una opción para cambiar la ubicación en el dispositivo en la que se guardan los archivos de las fotos: memoria interna o memoria extraíble.

La segunda parte del *sprint* se dedicó al manejo de las categorías. Se creó la interfaz para agregar una nueva categoría. Por último, se agregaron las funcionalidades para editar y eliminar categorías existentes.

### 5.3.6. Iteración 6

#### Objetivos

- Mostrar un reporte con la lista de ingresos/gastos asociados a una cuenta, en un rango de fecha dado
- Mostrar un reporte con el balance total por categorías asociadas a una cuenta, en un rango de fecha dado
- Permitir el envío de los reportes en un archivo PDF a otras personas

#### Resultados

- Se implementó la funcionalidad para listar los ingresos/gastos de una cuenta en el rango de fecha ingresado por el usuario
- Se implementó la funcionalidad para listar el balance total de las categorías de una cuenta en el rango de fecha ingresado por el usuario
- Se creó la funcionalidad para compartir el reporte con la lista de *entries*, incluyendo las fotos, de la cuenta en el rango de fecha ingresado.
- Se creó la funcionalidad para compartir el reporte con la lista de *entries*, sin incluir las fotos, de la cuenta en el rango de fecha ingresado.
- Se creó la funcionalidad para compartir el reporte con el balance total de las categorías de la cuenta en el rango de fecha ingresado.

#### Actividades

Esta iteración se dedicó exclusivamente al manejo de los reportes. Se tenía como requerimiento el manejo de dos tipos de reportes: reporte de balance general y reporte por categorías.

Para la generación de ambos reportes, se puede especificar una cuenta y un rango de fechas. El reporte de balance general incluye la lista de todos los ingresos/gastos de la cuenta escogida, dentro del rango de fechas establecido. El reporte por categorías incluye únicamente el monto total de todos los ingresos/gastos (es decir, la suma de ingresos y gastos) divididos por categorías, de la cuenta y rango de fechas escogidos.

Se creó una vista para mostrar el reporte de balance general, que muestra una lista con los ingresos/gastos asociados, sus montos, fechas y descripción (o categoría, en caso de que la descripción sea nula). También se creó una vista para el reporte por categoría, donde se muestra una lista con el nombre de cada categoría y la suma total de los ingresos y gastos correspondientes.

Luego, se creó la funcionalidad para generar un archivo PDF con la información de los reportes descritos anteriormente. Para esto se usó una librería que provee la plataforma de Android para la creación de archivos PDF.

Se tenía como requerimiento que el usuario pueda escoger generar un archivo con el reporte de balance general incluyendo las fotos de los ingresos/gastos asociados, o sin incluirlas.

Luego se implementaron los métodos necesarios para crear un archivo PDF con el reporte del balance general. Esta funcionalidad luego se adaptó para permitir también la creación del reporte por categorías. Para la escritura del PDF, se tuvo que lidiar manualmente con la paginación.

Por último, se creó la funcionalidad para compartir este archivo PDF con otras personas a través de otras aplicaciones instaladas en el dispositivo móvil. Dentro de estas aplicaciones se incluyen las de correo electrónico y mensajería móvil que soporten el envío de archivos.

Durante la creación del archivo PDF se presentaron diferentes dificultades. En primer lugar, las librerías existentes en Java para generar archivos con dicha extensión hacen uso de otras librerías, las cuales no son soportadas directamente en la plataforma de Android.

Por esta razón, en un principio se decidió utilizar una adaptación de una librería de Java para poder ser utilizada en Android. Sin embargo, no se encontró documentación suficiente que permitiera entender su uso. Por esta razón, se decidió finalmente utilizar una librería nativa de Android que facilita la creación de archivos PDF, pero que sólo está disponible para versiones de Android a partir de la 4.4. Esta decisión se apoyó en el hecho de que aproximadamente el 78,5 % de los usuarios de Android utilizan una versión mayor o igual a 4.4 [32].

### **5.3.7. Iteración 7**

#### **Objetivos**

- Crear un servicio web para permitir la autenticación de un usuario
- Crear un servicio web para recibir un reporte de gastos
- Crear una aplicación web para mostrar los reportes recibidos
- Permitir la aprobación o rechazo a través de la aplicación web de los reportes recibidos

#### **Resultados**

- Se diseñó e implementó el modelo de datos del servidor web
- Se creó un servicio al cual la aplicación móvil se puede conectar para la autenticación de usuarios
- Se creó un servicio a través del cual la aplicación móvil puede enviar reportes de gastos
- Se creó una aplicación web, a través de la cual se permite la creación de nuevos usuarios y el manejo de reportes (listar, aprobar y rechazar reportes recibidos)

## Actividades

En esta iteración se creó el servidor web al cual la aplicación móvil puede enviar peticiones.

En primer lugar, se creó el proyecto en AppFuse, con Tapestry como *framework* para el desarrollo de la aplicación web. Utilizando Hibernate con MySQL, se crearon las tablas necesarias para implementar el modelo de datos mostrado en la figura 5.3.

Luego, se creó el servicio web para la autenticación de un usuario.

Una vez creado, se implementó en la aplicación móvil una funcionalidad que permite iniciar sesión en el dispositivo. Para esto fue necesario crear una interfaz y los métodos necesarios para conectarse al servicio web para la autenticación del usuario.

Por otra parte, se creó un servicio web para recibir un reporte de gastos. Era necesario enviar tanto el archivo PDF con la información de los gastos, como un resumen de los totales por las categorías de interés para la empresa. Por esta razón, la comunicación entre la aplicación móvil y el servidor para el envío de reportes se hizo a través de una petición HTTP POST Multipart, que contiene el archivo PDF y un JSON con el resumen mencionado anteriormente.

También se creó un módulo para el envío de correos electrónicos. Esto se hizo para notificar al supervisor cuando un nuevo reporte llega al servidor, así como para enviar el archivo PDF adjunto al correo.

Luego de esto se creó la página web a través de la cual un supervisor puede ver la lista de reportes recibidos, así como aprobarlos o rechazarlos.

Para esto, se creó una vista que muestre la lista de reportes pendientes por revisión (es decir, que aún no han sido aprobados ni rechazados). En esta vista se agregó una opción para poder descargar el archivo PDF de cada reporte, así como opciones para aprobarlo/rechazarlo.

Finalmente, se adaptó la vista mencionada anteriormente para poder mostrar una lista de los reportes ya revisados (es decir, que ya fueron aprobados o rechazados).

### 5.3.8. Iteración 8



## Capítulo 6

### Conclusiones y Recomendaciones

# Bibliografía

- [1] “Digitalica — the technology partner of your disruptive ideas.” <http://digitalicagroup.com/>, consultado el 30 de julio de 2016.
- [2] “Patrón de diseño.” [https://es.wikipedia.org/wiki/Patr%C3%B3n\\_de\\_dise%C3%B1o](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o), consultado el 9 de julio de 2016.
- [3] E. Freeman, E. Freeman, K. Sierra, and B. Bates, *Head First: Design Patterns*. O'Reilly, 1 ed., 2004.
- [4] “Singleton pattern.” [https://en.wikipedia.org/wiki/Singleton\\_pattern](https://en.wikipedia.org/wiki/Singleton_pattern), consultado el 10 de julio de 2016.
- [5] “Patrones de arquitectura.” [https://es.wikipedia.org/wiki/Patrones\\_de\\_arquitectura](https://es.wikipedia.org/wiki/Patrones_de_arquitectura), consultado el 26 de julio de 2016.
- [6] “Modelo-vista-controlador.” <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>, consultado el 10 de julio de 2016.
- [7] “Model-view-presenter.” <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter>, consultado el 10 de julio de 2016.

- [8] “Differences between mvc and mvp for beginners.” <http://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners>, consultado el 10 de julio de 2016.
- [9] D. Garlan and M. Shaw, *An introduction to Software Engineer*. World Scientific, 1 ed., 1993.
- [10] “Tipos de arquitectura cliente servidor.” <https://www.ibiblio.org/pub/Linux/docs/LuCaS/Manuales-LuCAS/doc-curso-salamanca-LAMP/lamp-teoria-html/ch01s02.html>, consultado el 2 de agosto de 2016.
- [11] “Web service.” [https://en.wikipedia.org/wiki/Web\\_service](https://en.wikipedia.org/wiki/Web_service), consultado el 2 de agosto de 2016.
- [12] “Post(http).” [https://en.wikipedia.org/wiki/POST\\_%28HTTP%29](https://en.wikipedia.org/wiki/POST_%28HTTP%29), consultado el 2 de agosto de 2016.
- [13] “What is android os?.” <https://www.techopedia.com/definition/14873/android-os>, consultado el 30 de julio de 2016.
- [14] “What is java? definition and meaning.” <http://www.businessdictionary.com/definition/Java.html>, consultado el 26 de julio de 2016.
- [15] “Android software development.” [https://en.wikipedia.org/wiki/Android\\_software\\_development](https://en.wikipedia.org/wiki/Android_software_development), consultado el 30 de julio de 2016.
- [16] “What is mysql?.” <http://searchenterpriselinux.techtarget.com/definition/MySQL>, consultado el 26 de julio de 2016.
- [17] “Sqlite.” <https://es.wikipedia.org/wiki/SQLite>, consultado el 30 de julio de 2016.

- [18] “Maven - introduction.” <https://maven.apache.org/what-is-maven.html>, consultado el 30 de julio de 2016.
- [19] “Jetty.” <https://es.wikipedia.org/wiki/Jetty>, consultado el 26 de julio de 2016.
- [20] “Android studio.” [https://es.wikipedia.org/wiki/Android\\_Studio](https://es.wikipedia.org/wiki/Android_Studio), consultado el 26 de julio de 2016.
- [21] “Eclipse - the eclipse foundation open source community website.” <http://www.eclipse.org/home>, consultado el 30 de julio de 2016.
- [22] S. Chacon and B. Straub, *Pro Git: Everything you ned to know about Git*. Apress, 2 ed., 2014.
- [23] “Home - appfuse - appfuse conclucence.” <http://appfuse.org/display/APF/Home>, consultado el 26 de julio de 2016.
- [24] “Hibernate.” <https://es.wikipedia.org/wiki/Hibernate>, consultado el 26 de julio de 2016.
- [25] “Apache tapestry home page.” <http://tapestry.apache.org/index.html>, consultado el 30 de julio de 2016.
- [26] “Github - google/gson.” <https://github.com/google/gson>, consultado el 26 de julio de 2016.
- [27] “Retrofit.” <http://square.github.io/retrofit/>, consultado el 30 de julio de 2016.
- [28] “Scrum (software development).” [https://en.wikipedia.org/wiki/Scrum\\_%28software\\_development%29](https://en.wikipedia.org/wiki/Scrum_%28software_development%29), consultado el 7 de julio de 2016.
- [29] “Scrum basics.” <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf>, consultado el 7 de julio de 2016.

- [30] K. Rubin, *Essential Scrum: A practical guide to the most popular agile process*. Addison-Wesley Professional, 1 ed., 2012.
- [31] “Artefactos en scrum: claves para una organización diaria.” <http://www.desarrolloweb.com/articulos/artefactos-scrum.html>, consultado el 7 de julio de 2016.
- [32] “Dashboards — android developers.” <https://developer.android.com/about/dashboards/index.html?hl=es>, consultado el 29 de julio de 2016.