# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

# IMPLEMENTACIÓN DE UN PROTOTIPO DE APLICACIÓN MÓVIL QUE PERMITA CONECTAR EMPLEADORES CON TRABAJADORES QUE PERTENECEN AL SECTOR INFORMAL EN LA CIUDAD DE QUITO

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN**

**MARLON PAÚL APOLO QUISHPE**

**DIRECTOR: SORAYA LUCÍA SINCHE MAITA, PhD.**

**DMQ, agosto 2023**

# CERTIFICACIONES

Yo, Marlon Paúl Apolo Quishpe declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

**Marlon Paúl Apolo Quishpe**

Certifico que el presente trabajo de integración curricular fue desarrollado por Marlon Paúl Apolo Quishpe, bajo mi supervisión.

_____

**Soraya Lucía Sinche Maita, PhD.**

**DIRECTOR**

# DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Marlon Paúl Apolo Quishpe

Soraya Lucía Sinche Maita, PhD.

# DEDICATORIA

A mi madre que me ha servido de inspiración.

A mis abuelitos que me han enseñado a seguir adelante.

A mis tíos que me han ayudado mucho cuando los he necesitado.

A mis primos para servirles de inspiración.

# AGRADECIMIENTO

A toda mi familia por su apoyo.

A mi tutor de tesis por su apoyo y por sus enseñanzas en las aulas.

A mis profesores de universidad por enseñarme que aprender se vuelve divertido.

A mí, por nunca rendirme a pesar de las miles de adversidades.

# ÍNDICE DE CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# RESUMEN

En el último boletín publicado por la ENEMDU con fecha 24 de julio de 2023, se puede evidenciar que en el Ecuador la tasa de empleo informal está alrededor del 52,7 % [1], en comparación al publicado en diciembre de 2022, donde se indica que el empleo informal estaba en el 50.6 % [2]. Lo que evidencia que en el Ecuador ha crecido la tasa de empleo informal en un 2.1 %.

Uno de los principales problemas de las personas que se encuentran en este segmento es la dificultad que tienen para encontrar trabajo. Por lo tanto, en el presente trabajo se propone un buscador de trabajadores, que permita ayudar a las personas que se encuentran en este sector de la población, a mejorar sus ingresos económicos brindando sus servicios a través de una aplicación móvil. La aplicación permite conectar trabajadores informales que pertenecen a la ciudad de Quito con potenciales empleadores, basada en la metodología ágil XP, utilizando el sistema operativo Android, y la plataforma para el desarrollo de aplicaciones móviles *Firebase*.

El presente trabajo incluye tres capítulos. El primer capítulo incluye una revisión de los fundamentos teóricos indispensables para el desarrollo del proyecto, abordando temáticas como el Sistema Operativo Android, IDE (*Integrated Development Environment*) Android Studio, arquitectura de la aplicación, lenguaje de programación, gestor de base de datos y la plataforma *Firebase*. En el segundo capítulo, se distinguen dos secciones: la planificación y la implementación del proyecto. En la planificación se detallan aspectos como entrevistas, requerimientos funcionales y no funcionales, historias de usuario, módulos de la aplicación, sketches de las interfaces gráficas. La implementación del proyecto abarca las fases del proyecto y el proceso de desarrollo de la aplicación móvil. Finalmente, en el tercer capítulo se presentan las pruebas, resultados, conclusiones y recomendaciones obtenidas como resultado del desarrollo del proyecto técnico.

**PALABRAS CLAVE:** Android, *Extreme Programming*, *Firebase*, buscador de trabajadores, buscador de oficios.

# ABSTRACT

In the latest newsletter published by ENEMDU on July 24, 2023, it can be observed that in Ecuador, the informal employment rate is approximately 52.7% [1], compared to the report published in December 2022, which indicated informal employment at 50.6% [2]. This demonstrates that the informal employment rate in Ecuador has increased by 2.1%.

One of the main challenges for individuals in this segment is the difficultly they face in finding employment. Therefore, this work proposes a worker search platform that aims to assist people within this population segment in improving their economic incomes by offering their services through a mobile application. The application facilitates the connection of informal workers in the city of Quito with potential employers, using Android operating system, the agile methodology XP, and the *Firebase* platform for mobile application development.

This work comprises three chapters. First Chapter includes a review of the essential theoretical foundations necessary for the project development. Topics covered include the Android operating system, Android Studio Integrated Development Environment (IDE), application architecture, programming language, database management, and the *Firebase* platform. In the Second Chapter, two sections are distinguished: project planning and project implementation. The project planning details aspects such as interviews, functional and non-functional requirements, user stories, application modules, and graphical interfaces sketches. The project implementation covers project phases, and the mobile application development process. Finally, Third Chapter presents the testing, results, conclusions, and recommendations derived from the technical project's development.

**KEYWORDS:** Android, Extreme Programming, Firebase, worker finder, employer finder.

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En el año 2022 los estudios realizados por la ENEMDU indican que aproximadamente el 50.6 % de trabajadores laboran de manera informal en Ecuador [2]. Las empresas solicitan cumplir una serie de requisitos cada vez más exigentes para poder acceder a una plaza de trabajo, en la mayoría de los casos es necesario tener un título técnico o de tercer nivel. Además, los trabajadores informales no cuentan con un directorio o plataforma digital donde buscarlos.

En Ecuador se pueden encontrar aplicaciones móviles gratuitas como Jobra y Aora. Las dos aplicaciones son herramientas que buscan facilitar la contratación de un trabajador y mejorar la relación trabajador-empleador. Jobra permite encontrar talentos en su aplicación [3], mientras que Aora permite cotizar un posible trabajador ideal [4].

Las aplicaciones móviles se han convertido en grandes herramientas de comunicación y entretenimiento para los seres humanos.

En el presente trabajo de integración curricular se propone el desarrollo de un prototipo de aplicación móvil que permite conectar empleadores con trabajadores. La aplicación contiene herramientas de comunicación como mensajes, notificaciones, llamadas y videollamadas para que los usuarios registrados puedan comunicarse entre sí. También cuenta con herramientas de validación de datos que poseen algoritmos de *Machine Learning* para el registro de cada Trabajador. Adicionalmente contiene un sistema de calificaciones que permitirá categorizar a los diferentes trabajadores registrados en la aplicación según su desempeño.

## 1.1 Objetivo general

Implementar un prototipo de aplicación móvil que permita conectar empleadores con trabajadores que pertenecen al sector informal en la ciudad de Quito.

## 1.2 Objetivos específicos

1. Analizar los fundamentos teóricos y herramientas que se van a utilizar para el desarrollo del proyecto.
2. Diseñar los módulos a ser utilizados en el prototipo de la aplicación móvil.
3. Desarrollar los módulos diseñados para el prototipo de aplicación móvil.
4. Analizar los resultados en base a pruebas de funcionamiento del prototipo.

## 1.3 Alcance

El presente Trabajo de Integración Curricular tiene como propósito el desarrollo de un prototipo de aplicación móvil para smartphones con sistema operativo Android. La aplicación se basa en la metodología ágil XP (*Extreme Programming*), el IDE (*Integrated Development Environment*) Android Studio 4.0, el SDK (*Software Development Kit*) de *Google Firebase* para la comunicación con la base de datos NoSQL (*Not Only Structured Query Language*) y uso de los servicios de *Google Firebase*, la herramienta para la comunicación en tiempo real *Agora* y la plataforma como servicio *Heroku*.

La metodología XP permite el desarrollo de cualquier tipo de software, incluyendo las aplicaciones móviles y puede ser aplicada a un equipo conformado por una o muchas personas; además, permite una retroalimentación continua gracias a la estrecha relación que existe entre el equipo de desarrollo y el cliente. Por lo tanto, es perfecta para proyectos cambiantes e imprecisos.

De manera general el prototipo cuenta con el registro de al menos 10 oficios entre los cuales se tienen: carpintero, cerrajero, personal de limpieza, conductor, albañil, pintor, entre otros. Los nuevos trabajadores deberán pasar por un proceso de registro, en el cual se solicita su récord policial como requisito obligatorio, que se valida con la ayuda de ML (*Machine Learning*) *kit* de *Firebase*, el cual permite mediante algoritmos de *machine learning* el reconocimiento de texto dentro de imágenes o documentos PDF (*Portable Document Format*) [5], con esta herramienta se recupera el texto necesario para validación de los antecedentes penales del usuario y su posterior acceso a la aplicación. Además, se comprueba que la dirección de correo electrónico, nombres, apellidos concuerden con los proporcionados por cada usuario, y gracias a *Firebase Authentication* solo usuarios que cuenten con las credenciales de autenticación generadas en el proceso de registro podrán utilizar la aplicación.

La aplicación móvil cuenta con tres roles: Trabajador, Empleador y Administrador.

- **Trabajador:** usuario que ofrece sus servicios dentro de la aplicación. El usuario debe registrarse en la aplicación para poder atender a las notificaciones o mensajes que le envíen sus posibles empleadores. Además, puede visualizar el desempeño que ha tenido en todos sus trabajos, y modificar su información personal de ser necesario.
- **Empleador:** utiliza la aplicación como buscador de trabajadores. El empleador puede enviar notificaciones e incluso mensajes a los diferentes trabajadores

registrados en la aplicación; para que, de esa manera, se pueda facilitar la comunicación y posible contratación de los servicios de un trabajador.

- **Administrador:** encargado de realizar el control de documentos y de gestionar la información de los usuarios registrados. Además, tiene control directo sobre el CRUD (*Create Read Update Delete*) de los usuarios en la base de datos.

Cada uno de los roles cuenta con una interfaz diferente.

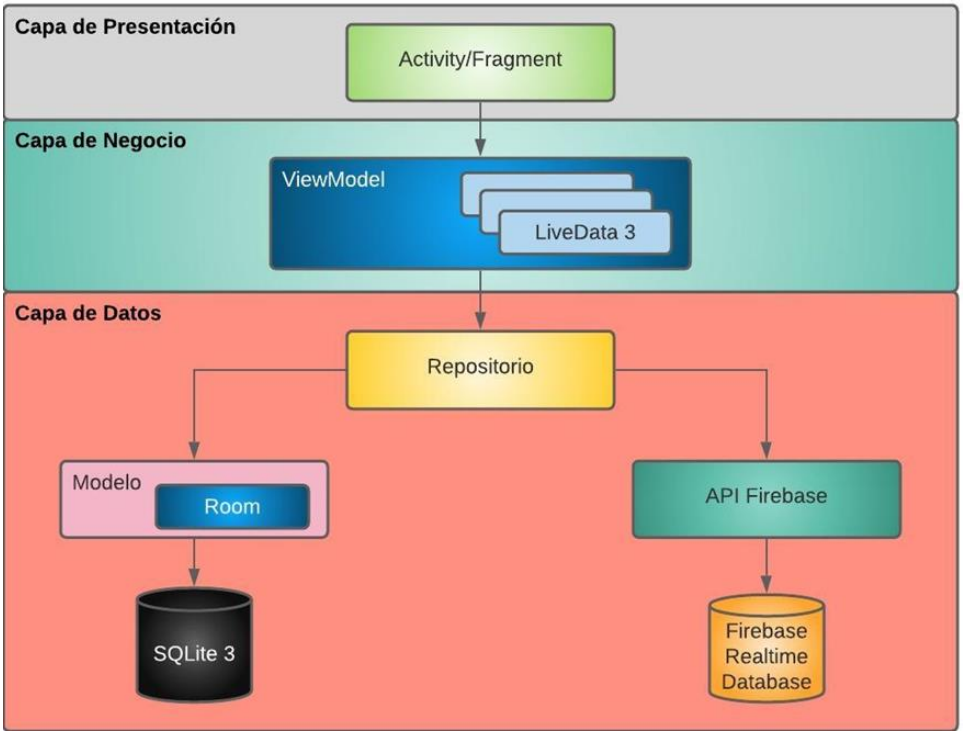



**Figura 1.1.** Arquitectura de 3 capas prototipo de aplicación móvil [6].

Se utiliza una arquitectura de 3 capas (Figura 1.1): presentación, negocio y datos.

En la capa de presentación se encuentra la interfaz gráfica, programada dentro de un *Activity* o *Fragment*, estos objetos permiten mantener la lógica de interacciones entre el sistema operativo y la interfaz gráfica.

La capa de negocio utiliza las clases *ViewModel* y *LiveData*. La clase *ViewModel* proporciona métodos para acceder a los datos dentro de un componente de UI (*User Inteface*) como un *Activity* o *Fragment*, y la clase *LiveData* es la encargada de retener los datos procesados para su posterior despliegue en la interfaz gráfica. Además, incluye la lógica para comunicarse con el modelo de datos persistente de una fuente local o una fuente de datos de back-end remota como *Firebase Realtime Database* [5].

La capa de datos se implementa mediante clases que representarán a los repositorios. Los repositorios pueden ser considerados como mediadores entre diferentes fuentes de datos, como modelos persistentes, servicios web y memorias caché [6].

El prototipo cuenta con los siguientes módulos:

- **Registro:** Permite registrar nuevos trabajadores o empleadores dentro de la aplicación. Los nuevos usuarios deben seguir el proceso de registro, que consiste en llenar un formulario donde se ingresan sus nombres completos, correo electrónico y una contraseña. En el caso del rol de trabajador existe un paso adicional, pues debe proporcionar su récord policial; además se incluye una casilla, donde puede escoger qué oficios tienen la posibilidad de realizar. Este módulo utiliza *Firebase ML kit*, que permite realizar la validación del récord policial del nuevo trabajador y *Firebase Realtime Database* que contiene las funciones necesarias para poder comunicarse con la base de datos.
- **Inicio de sesión:** En este módulo, los usuarios registrados dentro de la aplicación inician sesión utilizando el correo electrónico y contraseña ingresados al momento de su registro. El módulo de inicio de sesión utiliza *Firebase Authentication* para métodos de autenticación mediante cuentas de correo electrónico; y *Firebase Realtime Database* para tener acceso a los datos del usuario que ha iniciado sesión.
- **Buscador:** Permite a los empleadores realizar una búsqueda de trabajadores en base a un oficio. Los trabajadores que coincidan con la búsqueda realizada son desplegados en la pantalla de la aplicación en una lista ordenada de mayor a menor calificación de acuerdo con su desempeño laboral. El desempeño laboral lo califica un empleador luego que el trabajador ha cumplido con la actividad determinada para la cual fue contratado. Este módulo utiliza la herramienta de *Firebase Realtime Database* para poder acceder a la información de los trabajadores registrados dentro de la aplicación.
- **Notificaciones:** Permite que un empleador pueda enviar un mensaje corto con los requerimientos de una actividad determinada a un trabajador informal específico. Las notificaciones pueden ser aceptadas o rechazadas por el trabajador. En el caso de que una notificación sea aceptada se establece un canal de comunicación con el empleador dentro de la aplicación, caso contrario si la solicitud es rechazada simplemente se notifica al empleador que el trabajador no aceptó la oferta laboral.

- **Mensajería instantánea:** Este módulo permite realizar envío de mensajes y contenido multimedia entre trabajadores y empleadores. El servicio de mensajería instantánea es activado luego de que un trabajador acepta una notificación de trabajo enviada por un empleador.
- **Perfil:** Este módulo permite que un trabajador o empleador pueda modificar su respectiva información como correo electrónico y contraseña. En el perfil del trabajador se incluye una casilla extra que permite modificar los oficios registrados.

Los módulos de notificaciones, mensajería instantánea y perfil utilizan *Firebase Realtime Database* para obtener acceso a los datos de trabajadores y empleadores registrados dentro de la aplicación. Además, se utiliza la clase *Activity* para programar la interfaz gráfica de todos los módulos; esta clase permite obtener acceso a los objetos dentro de una interfaz gráfica, donde se envían y muestran los datos desde y hacia el servidor en cada módulo respectivamente. La API *(Application Programming Interface*) de *Firebase* se utiliza para programar la lógica de negocio que conecta cada uno de los módulos mencionados anteriormente.

Las pruebas de funcionamiento de la aplicación móvil se realizan con al menos 5 empleadores, 5 trabajadores informales y un usuario con el rol de Administrador.

El presente trabajo de integración curricular tiene un producto final demostrable.

## 1.4 Marco teórico

En esta sección se presentan los fundamentos teóricos y las herramientas necesarias para el desarrollo del proyecto. A continuación, se realiza una descripción del sistema operativo Android y su arquitectura, el desarrollo de aplicaciones para dispositivos móviles que utilizan Android, la metodología ágil XP, el lenguaje de programación Java, el lenguaje de etiquetado XML y la plataforma para el desarrollo de software *Firebase*.

### 1.4.1 Desarrollo de aplicaciones móviles en Android

Android es un sistema operativo y una plataforma móvil de programación, puede funcionar en cientos de millones de dispositivos móviles con pantalla táctil, como tablets y smartphones [7]. También incluye un SDK que permite escribir código para crear aplicaciones dirigidas a usuarios de Android y un mercado para distribuirlas. En conjunto, Android representa un ecosistema para aplicaciones móviles [7].

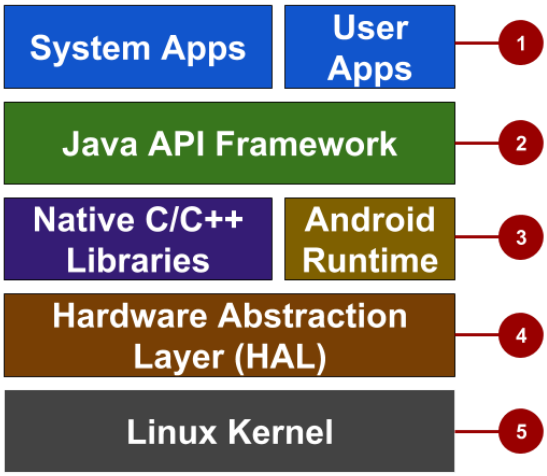



**Figura 1.2.** Capas del sistema operativo Android y su arquitectura de desarrollo [7]

En la Figura 1.2. se observan los siguientes componentes:

- **Aplicaciones:** En este nivel se encuentran las aplicaciones centrales del sistema operativo que permiten el envío de SMS (*Short Message Service*), correo electrónico, acceso a internet y contactos, entre otros. De la misma manera aquí se encuentran las aplicaciones desarrolladas para Android.
- **Java API framework:** Las funciones para el desarrollo en Android como los componentes de interfaz gráfica, gestión de recursos y gestión del ciclo de vida de la aplicación, se encuentran disponibles a través de APIs (*Application Programming Interfaces*).
- **Bibliotecas y Android Runtime:** En esta capa se encuentran las bibliotecas que permiten desarrollar aplicaciones a través del *framework* de la API de Java, pero se encuentran escritas nativamente en C y C++.
- **HAL (*Hardware Abstraction Layer*):** Esta capa consta de varios módulos que permiten exponer las capacidades del hardware del dispositivo, a través de interfaces estándar al *framework* de la API de JAVA. Cada interfaz sirve para implementar un tipo específico de componente de Hardware, como la cámara o el módulo Bluetooth.
- **Kernel de Linux:** Es el núcleo del sistema operativo Android. Todas las capas sobre el kernel de Linux lo utilizan para la generación de nuevos subprocesos o gestión de memoria de bajo nivel.

### 1.4.2 IDE Android Studio

Android Studio es el IDE oficial para el desarrollo de aplicaciones para dispositivos con tecnología Android. El IDE posee un editor de códigos, con código de muestra para funciones comunes de aplicaciones; también incluye un emulador acompañado de diferentes funciones que permiten probar prototipos de aplicaciones en diferentes dispositivos y simular diferentes funciones de hardware. Además, permite la integración con GitHub, y compatibilidad con *Firebase*; facilitando la integración con servicios como *Firebase Authentication, Firebase Realtime Database*, *Firebase Machine Learning Kit*, entre otros.

Al iniciar un nuevo proyecto en Android Studio, el IDE permite escoger la versión de SDK, escribir el nombre y elegir una plantilla para el proyecto, también crea una nueva carpeta que contiene todos los archivos necesarios para que la aplicación pueda funcionar.

### 1.4.3 Estructura de un proyecto en Android

Un proyecto de Android Studio contiene todo el código fuente y todos los recursos de una aplicación en Android. Los recursos incluyen diseños, cadenas, colores, dimensiones e imágenes [7].

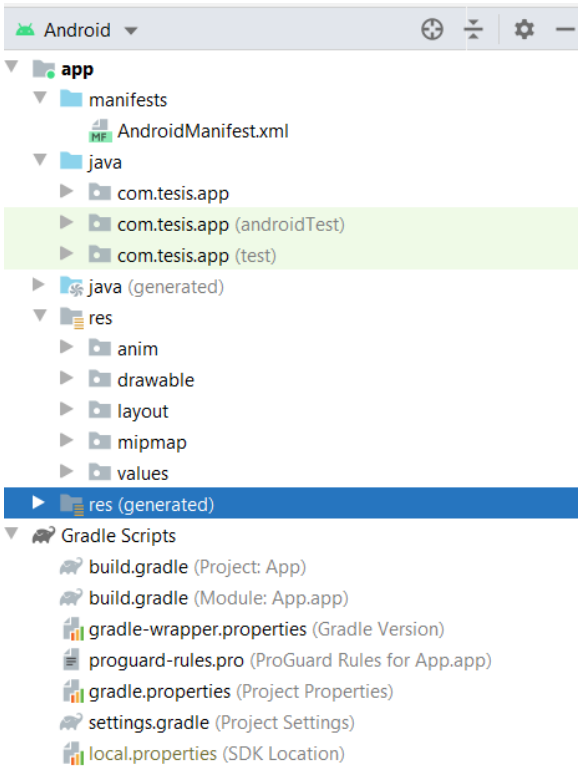



**Figura 1.3.** Vista Android dentro de un proyecto en Android Studio

En la Figura 1.3. se observan las diferentes carpetas que contiene un proyecto en Android Studio:

- **manifests:** se encuentra el archivo Manifest.xml, donde se describen todos los componentes de una aplicación en Android.
- **java:** se encuentra todo el código fuente de la aplicación y las pruebas unitarias escritas en Java.
- **res:** contiene los archivos de recursos como íconos, imágenes, diseños XML, audios, entre otros, necesarios para el funcionamiento de la aplicación.
- **Gradle Scripts:** se encuentran todos los archivos que la herramienta Gradle necesita para compilar la aplicación.

### 1.4.4 Firebase Software Development Kit

*Firebase* es una plataforma de desarrollo para aplicaciones web y móviles. Su objetivo principal es proporcionar herramientas y la infraestructura necesaria para el desarrollo de aplicaciones [8]. *Firebase* ofrece a los desarrolladores la posibilidad de evitar toda la infraestructura y mantenimiento requeridos por un servidor. Además, proporciona herramientas para la autenticación de usuarios y ofrece incluso funcionalidades para el desarrollo de aplicaciones con inteligencia artificial. Estas ventajas permiten que los desarrolladores se centren en el desarrollo del producto, evitando obstáculos innecesarios y posibles limitaciones al crear aplicaciones web o móviles. Al evitar el mantenimiento de servidores intermedios, *Firebase* simplifica el proceso de desarrollo y reduce posibles inconvenientes [8].

- ***Firebase Authentication*:** Facilita la creación de sistemas de autenticación seguros, al mismo tiempo que mejora la experiencia de inicio de sesión del usuario. Proporciona una solución de identidad extremo a extremo, compatible con cuentas de correo electrónico y contraseña, autenticación de teléfono y acceso a *Google*, *Twitter*, *Facebook*, *GitHub*, y más.
- ***Firebase Realtime Database*:** Almacena y sincroniza los datos de los usuarios registrados en la aplicación en una base de datos NoSQL alojada en la nube. Los datos son almacenados en formato JSON (*JavaSript Object Notation),* se sincronizan con todos los clientes en tiempo real y se mantienen disponibles cuando la aplicación no tiene conexión. Las reglas de *Firebase Realtime Database* representan una configuración declarativa para la base de datos, la configuración

declarativa permite imponer reglas para mejorar el control de los privilegios y el nivel de acceso de cada usuario que hace uso de *Firebase Realtime Database*; esto significa que las reglas se definen de manera independiente de la lógica de la aplicación. *Firebase* proporciona dos modos de inicio (modo de prueba y modo bloqueado) para crear las reglas de seguridad en la base de datos.

El modo de prueba permite que todos los usuarios lean y reemplacen los datos almacenados en la nube. Este modo es utilizado solo para pruebas y no es recomendado para ambientes de producción.

El modo bloqueado se encarga de rechazar todas las lecturas y escrituras de clientes móviles y web. En la versión gratuita de *Firebase*, se pueden almacenar hasta 1GB de datos y tener hasta 100 conexiones simultáneas [9].

La Figura 1.4. muestra la estructura de un árbol JSON para una aplicación de chat que almacena un perfil básico. En este caso no se tiene tablas, filas y tampoco columnas.


**Figura 1.4.** Estructura de un árbol JSON para una aplicación de chat [9]

- **ML (*Machine Learning*) *kit* de *Firebase***: Emplea algoritmos de aprendizaje automático (ML) para llevar a cabo la detección y extracción de texto dentro de imágenes. Este servicio se pone a disposición de los desarrolladores mediante su API de visión artificial. La API esta optimizada para el uso en dispositivos móviles y, además, permite la identificación de texto en tiempo real, ya que el procesamiento de imágenes se realiza directamente en el dispositivo móvil [5].

### 1.4.5 JAVA

Es un lenguaje de programación y una plataforma informática lanzada por primera vez por Sun Microsystems en 1995 [10]. Incluye un JDK *(Java Development Kit*) que contiene el software mínimo que se necesita para el desarrollo de Java. El JDK también contiene otras

herramientas, como el comando archivador (jar), que puede empaquetar archivos juntos, y el comando de documentación de la API (javadoc) para generar documentación [11]. Las piezas clave incluyen: un compilador y un lazador.

- **Compilador (javac):** Convierte los archivos .java en archivos .class e identifica errores dentro de un archivo de código fuente Java.
- **Lanzador java:** Crea la JVM (*Java Virtual Machine*) y ejecuta el programa.
- **Estructura de código en Java:** Un archivo de código fuente contiene una definición de clase. La clase representa una parte del programa. Un archivo de código fuente debe contener lo siguiente:
    - **Clases:** bloque de construcción básico que contiene uno o más métodos. Para utilizar la mayoría de las clases, hay que crear objetos. Un objeto es una instancia de una clase en la memoria [11].
    - **Métodos:** operación que puede ser llamada desde la misma u otra clase. El código del método es básicamente un conjunto de declaraciones [11].

### 1.4.6 XML (*Extensible Markup Language*)

Es un lenguaje de marcado y un formato simple basado en texto para representar información estructurada como: datos, configuraciones, libros, transacciones, facturas y mucho más [12]. XML también describe una clase de objetos llamados documentos XML, contiene formas, etiquetas, estructuras y protege la información.

Cada documento XML posee una estructura lógica y física. Físicamente, el documento está compuesto por unidades llamadas entidades. Una entidad puede referirse a otras entidades para provocar su inclusión en el documento. Un documento comienza en una "raíz" o entidad de documento. Lógicamente, el documento está compuesto por declaraciones, elementos, comentarios, referencias de caracteres e instrucciones de procesamiento, los cuales se indican en el documento mediante un marcado explícito [13].

XML permite incrustar símbolos en el texto, esto se conoce como *markup* (marcado) [14]. Un documento XML contiene elementos, los elementos son contenedores con información obtenidos mediante el *markup*.

### *1.4.7 HEROKU*

*Heroku* es una plataforma políglota de desarrollo y entrega de aplicaciones basada en la nube. *Heroku* utiliza una arquitectura multi-tenant construida sobre máquinas virtuales en *Amazon Elastic Compute Cloud* (Amazon EC2) [15].

*Heroku* elimina la necesidad de servidores físicos y la administración de todo el stack de protocolos necesarios para que este se mantenga funcionando sin inconvenientes, lo que permite a los desarrolladores enfocarse en la construcción de sus aplicaciones. La gestión y el escalado de aplicaciones se realizan a través de una CLI (interfaz de línea de comandos) o en línea utilizando el portal web, a través de la API.

La plataforma de *Heroku* ayuda al desarrollo de software que utilizan las metodologías ágiles, ya que cuenta con la integración del control de versiones de Git, y utilizando la plataforma de *Github*, el código de la aplicación puede ser compartido de manera inmediata con todas las personas que participan en el desarrollo del software. Esto permite una integración continua, una administración automatizada y la reversión de todo el sistema de software con un solo comando.

La CLI de *Heroku* permite controlar la base de código local y las aplicaciones remotas de *Heroku* en una terminal que se ejecuta en Windows, Mac o Linux.

Para utilizar *Heroku* se debe crear una cuenta en su portal web, instalar la CLI de la plataforma en una computadora y finalmente iniciar sesión en la CLI de *Heroku*. La guía inicial de *Heroku* publicada en su portal web permite desplegar un modelo de aplicación web escrita en diferentes lenguajes de programación como: Node.js, Ruby, Java, PHP, Phyton, Go, Scala, Clojure [16].

### *1.4.8 Agora*

*Agora* es un SDK que facilita la integración de llamadas de voz y videollamadas en aplicaciones móviles que utilicen S.O. Android [17].

*Agora* posee una red en tiempo real definida por software denominada *Agora* SD-RTN (*Software Defined Real-time Network*). La red de *Agora* permite que su servicio se encuentre disponible en cualquier parte del mundo y en cualquier momento. Además, cuenta con centros de datos ubicados en más de 200 países, lo que le permite establecer ventajas como: una red de baja latencia para transmitir datos de audio y video, calidad de servicio inigualable, alta disponibilidad y escalabilidad, y sobre todo un bajo costo [17]. Para

comenzar a utilizar los servicios de *Agora*, se deberá crear una cuenta en su plataforma utilizando un correo electrónico y una contraseña.

*Agora* utiliza el patrón publicación/suscripción. En la publicación un usuario envía datos de video o audio a un canal específico. La transmisión publicada se crea con los datos de audio muestreados desde un micrófono o los datos de video capturados por una cámara [17].

La suscripción es el acto de recibir flujos de medios publicados por usuarios remotos en el canal; en *Agora* los usuarios que tienen especificado el mismo nombre del canal podrán recibir datos de audio o video que se encuentran siendo transmitidos por otro usuario. Al publicar la transmisión local y suscribirse a transmisiones remotas, los usuarios se comunican entre sí en tiempo real [17].

# 2 METODOLOGÍA

En este capítulo se describe la metodología utilizada y la implementación del prototipo de aplicación móvil.

Para el desarrollo del proyecto se utiliza como base la metodología XP (*Extreme Programming*), esto debido a qué: XP es una metodología ligera para equipos de tamaño pequeño o mediano que desarrollan software ante requisitos imprecisos o rápidamente cambiantes [18]. Además, se centra en la aplicación de técnicas de programación, una comunicación clara y el trabajo en equipo.

A continuación, se describen las fases que permiten seguir el proceso de la metodología XP, desde la planificación hasta la prueba del software.

- **Planificación:** En esta etapa se escriben las historias de usuario donde se define la funcionalidad que le gustaría ver al cliente, junto con el valor comercial y la prioridad de cada una de esas características [19].
- **Diseño:** XP señala que un diseño sencillo permitirá añadir funciones más complejas de una manera más fácil en el futuro.
- **Codificación:** Todos revisan el código y cualquier desarrollador puede agregar una nueva funcionalidad, corregir errores o refactorizar el mismo.
- **Pruebas:** Todo el equipo debe realizar las pruebas unitarias y corregir el código antes de que se publique la aplicación. Las pruebas unitarias ayudan a encontrar pequeños fallos en la aplicación y evitan que estos se propaguen a medida que avanza el desarrollo. Cada que se produzca un fallo en la aplicación luego de una prueba de aceptación, se debe crear una prueba unitaria que demuestre el fallo.

En esta metodología también se cuenta con una serie de roles necesarios para completar el proyecto. Los roles en XP sugieren que una variedad de personas trabaje juntas y se encuentren interconectadas para hacer un proyecto más efectivo. A continuación, se describen los principales roles de la metodología XP:

- **Tester:** Ayuda a los clientes a elegir y escribir pruebas automatizadas a nivel de sistema antes su implementación. Además, se encargan de probar el sistema y mostrar los resultados a todo el equipo XP.
- **Project Manager:** Facilita la comunicación dentro del equipo, con los clientes, los proveedores y el resto de la organización.

- **Product Managers:** Ayudan al equipo a decidir las prioridades de las historias de usuario, analizando las diferencias entre los requisitos reales y los supuestos.
- **Redactores técnicos:** Son los primeros en ver las características del proyecto, a menudo cuando son sólo bosquejos.
- **Usuario:** Ayuda a escribir y elegir las historias de usuario; además, se involucran en la toma de decisiones durante el desarrollo.
- **Programadores:** Estiman las historias, escriben pruebas, escriben código para implementar características, automatizan el proceso de desarrollo y mejoran paulatinamente el diseño del sistema.

Además, para la correcta aplicación de la metodología XP se incluyen herramientas cómo:

- **Historias de Usuario:** son explicaciones cortas de alguna característica o funcionalidad del sistema, y son escritas por el usuario final de la aplicación. Estas herramientas ayudan a crear nuevos planes de lanzamiento y estimaciones de tiempo en el desarrollo del software.
- **Pruebas de aceptación:** permiten determinar el fin de una iteración y el comienzo de una nueva. Además, son necesarias para comprobar las funcionalidades del sistema desarrolladas en cada historia de usuario, y permiten obtener una retroalimentación continua para el desarrollo de las próximas historias de usuarios.

En la Figura 2.1. se describe el proceso para completar de manera exitosa con el desarrollo del prototipo de aplicación móvil utilizando XP. En este caso se utilizarán entrevistas, diagramas de casos de uso, diagramas de clases y diagramas de actividades, además de *sketches* para la creación de las diferentes interfaces gráficas que conforman la aplicación móvil.

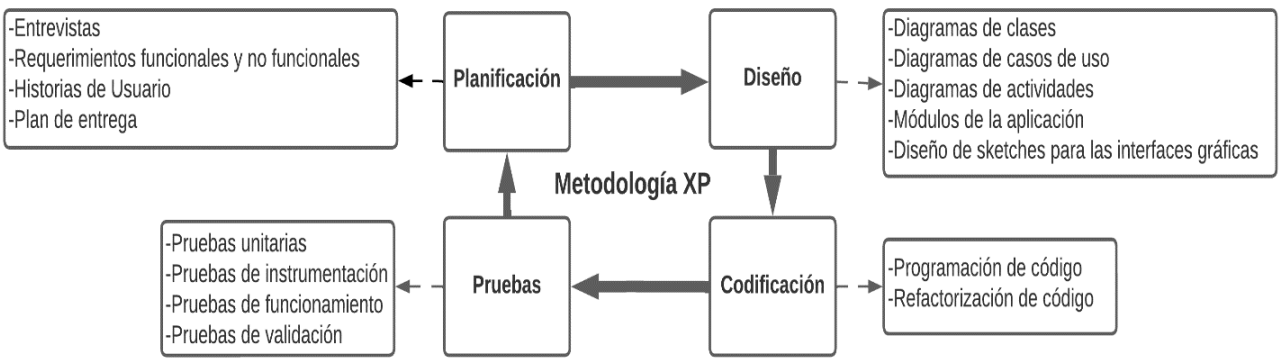



**Figura 2.1.** Ciclo de desarrollo XP [18]

## 2.1 Fase de Planificación

En esta fase se realizan las entrevistas a los posibles usuarios, se detallan los requerimientos funcionales y no funcionales de la aplicación, y finalmente se escriben las historias de usuario que se van a desarrollar en el proyecto.

### 2.1.1 Entrevistas

Para la obtención de los requerimientos funcionales y no funcionales se realizaron entrevistas a los potenciales usuarios de la aplicación. Los usuarios fueron identificados mediante tres roles: Administrador, Trabajador y Empleador. Cada rol tiene funciones diferentes y por lo tanto fue necesario realizar 3 modelos de entrevistas. Los resultados y modelos de las entrevistas realizadas a 5 Trabajadores, 5 Empleadores y el Administrador se encuentran en el ANEXO I.

### 2.1.2 Requerimientos Funcionales

Los requerimientos funcionales permiten describir el comportamiento del sistema. En base a la información obtenida por medio de las entrevistas, los requerimientos funcionales para el Administrador, Trabajadores, Empleadores y los requerimientos comunes entre los 3 usuarios se encuentran listados en la Tabla 2.1, Tabla 2.2, Tabla 2.3 y Tabla 2.4 respectivamente.

**Tabla 2.1.** Requerimientos funcionales del Administrador

| Número | Requerimiento funcional |
|---|---|
| 1 | Crear, visualizar, actualizar y eliminar empleadores |
| 2 | Crear, actualizar y eliminar trabajadores |
| 3 | Crear, actualizar y eliminar oficios |
| 4 | Visualizar y actualizar la información de su cuenta personal |

**Tabla 2.2.** Requerimientos funcionales del Trabajador

| Número | Requerimiento funcional |
|---|---|
| 5 | Crear, visualizar, actualizar y eliminar la información de su cuenta personal. |
| 6 | Validar récord policial utilizando técnicas de *Machine Learning* |
| 7 | Crear oficios |
| 8 | Crear, visualizar, actualizar y enviar de citas de trabajo |

**Tabla 2.3.** Requerimientos funcionales del Empleador

| Número | Requerimiento funcional |
|---|---|
| 9 | Crear, visualizar, actualizar y eliminar la información de su cuenta personal. |
| 10 | Visualizar la información de los oficios registrados |
| 11 | Permitir la búsqueda de trabajadores |
| 12 | Visualizar, actualizar y recibir de citas de trabajo |
| 13 | Calificar a un trabajador por su desempeño |

**Tabla 2.4.** Requerimientos funcionales comunes

| Número | Requerimiento funcional |
|---|---|
| 14 | Permitir el inicio de sesión utilizando un correo electrónico y una contraseña. |
| 15 | Visualizar la información de los trabajadores registrados |
| 16 | Visualizar la información de los oficios registrados |
| 17 | Permitir el envío, recepción y lectura de Notificaciones |
| 18 | Permitir el envío, recepción y lectura de mensajes de texto, audio e imágenes |
| 19 | Permitir el cierre de sesión dentro de la aplicación |
| 20 | Permitir notificaciones en segundo plano |
| 21 | Realizar llamadas de voz y videollamadas uno a uno |

### 2.1.3 Requerimientos no funcionales

Los requerimientos no funcionales representan atributos y características que establecen las pautas para la calidad, el rendimiento y las limitaciones del software. Los requisitos no funcionales considerados para este proyecto se encuentran detallados en la en la Tabla 2.5.

**Tabla 2.5.** Requerimientos no funcionales

| Número | Requerimiento no funcional |
|---|---|
| 1 | El prototipo de aplicación móvil está orientado a dispositivos con sistema operativo Android. |
| 2 | El lenguaje de programación utilizado para el diseño del prototipo es JAVA. |
| 3 | El lenguaje de etiquetado para la capa View es XML. |
| 4 | La base de datos utilizada es no relacional (NoSQL). |
| 5 | Se necesita de conexión a internet para utilizar la aplicación. |

### 2.1.4 Historias de Usuario

Las historias de usuario son tarjetas con información escritas por el cliente representando los requerimientos que la aplicación debe cumplir. En las Tabla 2.6, se observa el formato de Historia de Usuario utilizado en el proyecto. El detalle de las historias de usuario se encuentra en el ANEXO II.

**Tabla 2.6.** Historia de usuario: Interfaces de Usuario

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Usuario: Administrador/Trabajador/Empleador</td></tr>
<tr><td colspan="3">Nombre historia: Interfaces de Usuario</td></tr>
<tr><td colspan="2">Prioridad en negocio:<br>Alta</td><td>Riesgo en desarrollo:<br>Medio</td></tr>
<tr><td colspan="2">Puntos estimados: 2</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="3">Programador responsable: Marlon Apolo</td></tr>
<tr><td colspan="3">Descripción:<br>El sistema debe contener las siguientes interfaces:<br>• Interfaz gráfica de bienvenida a la aplicación<br>• Interfaz gráfica de información inicial<br>• Interfaz gráfica de inicio y cierre de sesión<br>• Interfaz gráfica principal<br>• Interfaz gráfica para registrar trabajadores, empleadores y oficios<br>• Interfaz gráfica que permita validar el récord policial.<br>• Interfaz gráfica que permita visualizar la información de los trabajadores, empleadores y oficios registrados<br>• Interfaz gráfica que permita visualizar los chats entre usuarios<br>• Interfaz gráfica que permita enviar mensajes de texto, audio e imágenes<br>• Interfaz gráfica que permita realizar llamadas de voz y videollamadas<br>• Interfaz gráfica que permita visualizar y actualizar mis datos personales<br>• Interfaz gráfica que permita crear, visualizar, modificar y actualizar citas de trabajo<br>• Interfaz gráfica que permita compartir mi ubicación<br>• Interfaz gráfica que permita eliminar mis datos personales<br>• Interfaz gráfica para realizar la búsqueda de Trabajadores<br>• Interfaz gráfica que permita visualizar la política de privacidad de la aplicación<br>• Interfaz gráfica que contenga información sobre la empresa desarrolladora</td></tr>
<tr><td colspan="3">Observaciones:<br>CONFIRMADO con el cliente.</td></tr>
</table>

### 2.1.5 Plan de entrega

El plan de entrega es elaborado luego de identificar las Historias de Usuario, para su elaboración se debe establecer el tiempo calendario y estimar el esfuerzo de desarrollo del proyecto. El tiempo calendario para el desarrollo del proyecto se encuentra detallado en la Tabla 2.7, y servirá para poder calcular el esfuerzo utilizado en el desarrollo del prototipo.

**Tabla 2.7.** Tiempo calendario

| Horas calendario | Días calendario |
|---|---|
| 8 horas (número de horas diarias dedicadas al proyecto) | 5 días (número de días laborables que se va a dedicar al proyecto) |

Para realizar la estimación del esfuerzo de desarrollo de las historias de usuario detalladas en la Tabla 2.9, el desarollador asigna la siguiente relación: el esfuerzo tiene como métrica el valor de un punto de historia de usuario, donde 1 punto de historia de usuario es igual a 1 semana de trabajo sin interrupciones por parte del desarrollador, y 1 semana de trabajo equivale a 10 días laborables. Además, se tiene que 1 día laborable es igual a 8 horas de trabajo, y de esta manera se concluye que en 1 semana el desarrollador trabaja un total de 40 horas sin interrupciones.

Con estas acotaciones, se procede a realizar un ejemplo para el cálculo de la estimación del esfuerzo de desarrollo para completar la historia de usuario número 1.

El desarrollador estima que la implementación de la historia de usuario 1 le puede costar un esfuerzo de 2 puntos de historia. Para llegar a esta conclusión y, al no tener referentes como certificados que avalen su conocimiento el lenguaje de programación Java. El desarrollador ha decido tomar como referente el haber aprobado la asignatura de programación orientada a objetos, con esa aclaración procede a llegar a la conclusión que le costaría un esfuerzo de 2 puntos de historia de usuario en completar la historia de usuario número 1.

El detalle de la estimación del esfuerzo para la historia de usuario 1 se encuentra en la Tabla 2.8.

**Tabla 2.8.** Estimación de esfuerzo para la historia de usuario 1 en base a una persona

| Equipo | Horas de esfuerzo de desarrollo | Días de esfuerzo de desarrollo | Semanas de esfuerzo de desarrollo | Esfuerzo |
|---|---|---|---|---|
| 1 persona | 80 horas | 10 días | 2 semanas | 2 puntos de historia |

Todos los cálculos de la Tabla 2.9 se encuentran elaborados en base a las métricas y relaciones mencionadas anteriormente.

**Tabla 2.9.** Estimación de esfuerzo individual de las Historias de Usuario.

| Nº | Nombre | Esfuerzo | Esfuerzo de desarrollo | | |
|---|---|---|---|---|---|
| | | | Semanas | Días | Horas |
| 01 | Interfaces de Usuario | 2 | 2 | 10 | 80 |
| 02 | Registro de Oficios | 1 | 1 | 5 | 40 |
| 03 | Registro de Trabajadores | 1,6 | 1,6 | 8 | 64 |
| 04 | Registro de Empleadores | 1,4 | 1,4 | 7 | 56 |
| 05 | Autenticación de Usuarios | 1,8 | 1,8 | 9 | 72 |
| 06 | Administración de Datos Personales | 1,2 | 1,2 | 6 | 48 |
| 07 | Buscador | 2 | 2 | 10 | 80 |
| 08 | Comunicación entre Usuarios | 2 | 2 | 10 | 80 |
| 09 | Videollamadas entre Usuarios | 2 | 2 | 10 | 80 |
| 10 | Llamadas de Voz entre Usuarios | 2 | 2 | 10 | 80 |
| 11 | Notificaciones | 2 | 2 | 10 | 80 |

Para estimar el esfuerzo de desarrollo total del proyecto detallado en la Tabla 2.10, se realiza la suma de todos los tiempos de esfuerzo individuales de cada historia de usuario, y se considera que el equipo de trabajo está conformado por una sola persona.

**Tabla 2.10.** Estimación de esfuerzo total del proyecto en base a una persona

| Equipo | Horas de esfuerzo de desarrollo | Días de esfuerzo de desarrollo | Semanas de esfuerzo de desarrollo |
|---|---|---|---|
| 1 persona | 760 horas | 95 días | 19 semanas |

Una vez calculada la estimación del esfuerzo total del proyecto, se procede a la elaboración del plan de entrega mostrado en la Tabla 2.11.

**Tabla 2.11.** Plan de Entrega del proyecto

| Módulo | Nro. | Nombre de Historia | Esfuerzo de desarrollo | | | Calendario Estimado | | | Iteración Asignada | | | | | | | | Entrega Asignada | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Semanas ideales | Días ideales | Horas ideales | Semanas estimadas | Días estimados | Horas estimadas | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 |
| Todos | 1 | Interfaces de Usuario | 2 | 10 | 80 | 2 | 10 | 80 | x | | | | | | | | x | | | |
| Módulo de Registro | 2 | Registro de Oficios | 1 | 5 | 40 | 1 | 5 | 32 | x | | | | | | | | x | | | |
| | 3 | Registro de Trabajadores | 1,6 | 8 | 64 | 1,6 | 8 | 48 | | x | | | | | | | x | | | |
| | 4 | Registro de Empleadores | 1,4 | 7 | 56 | 1,4 | 7 | 48 | | x | | | | | | | x | | | |
| Módulo de Inicio de Sesión | 5 | Autenticación de Usuarios | 1,8 | 9 | 72 | 1,8 | 9 | 72 | | | x | | | | | | | x | | |
| Módulo de Perfil | 6 | Administración de Datos Personales | 1,2 | 6 | 48 | 1,2 | 6 | 40 | | | x | | | | | | | x | | |
| Módulo de Búsqueda | 7 | Buscador | 2 | 10 | 80 | 2 | 10 | 80 | | | | x | | | | | | x | | |
| Módulo de Mensajería Instantánea | 8 | Comunicación entre Usuarios | 2 | 10 | 80 | 2 | 10 | 72 | | | | | x | | | | | | x | |
| | 9 | Videollamadas entre Usuarios | 2 | 10 | 80 | 2 | 10 | 80 | | | | | | x | | | | | x | |
| | 10 | Llamadas de Voz entre Usuarios | 2 | 10 | 80 | 2 | 10 | 80 | | | | | | | x | | | | x | |
| Módulo de Notificaciones | 11 | Notificaciones | 2 | 10 | 80 | 2 | 10 | 48 | | | | | | | | x | | | | x |
| | | | | | | | | Total de semanas | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 19 | | | |

Las semanas estimadas del Plan de Entrega utilizado en el proyecto se encuentran detalladas en la Tabla 2.12.

**Tabla 2.12.** Semanas estimadas del Plan de Entrega del proyecto.

| Nº | Historia de Usuario | Semanas | Semana inicio | Semana fin |
|---|---|---|---|---|
| 01 | Interfaces de Usuario | 3 semanas | 1ra semana de marzo | 3ra semana de marzo |
| 03 | Registro de Trabajadores | 3 semanas | 4ta semana de marzo | 2da semana de abril |
| 05 | Autenticación de Usuarios | 3 semanas | 3ra semana de abril | 1ra semana de mayo |
| 07 | Buscador | 2 semanas | 2da semana de mayo | 3ra semana de mayo |
| 08 | Comunicación entre Usuarios | 2 semanas | 4ta semana de mayo | 1ra semana de junio |
| 09 | Videollamadas entre Usuarios | 2 semanas | 2da semana de junio | 3ra semana de junio |
| 10 | Llamadas de Voz entre Usuarios | 2 semanas | 4ta semana de junio | 1ra semana de julio |
| 11 | Notificaciones | 2 semanas | 2da semana de julio | 3ra semana de julio |

## 2.2 Fase de Diseño

### 2.2.1 Diagrama de Casos de Uso

El diagrama de casos de usos permite modelar y entender de mejor manera como se relacionan los diferentes actores dentro de un sistema. El diagrama de casos de uso del prototipo de aplicación móvil se muestra en el ANEXO III.

### 2.2.2 Diagrama de clases

El diagrama de clases muestra la relación que existe entre los objetos que conforman la aplicación. Los diagramas de clases fueron clasificados en base a la arquitectura MVVM (*Model-View-ViewModel*), esta arquitectura divide la aplicación en tres partes independientes: modelo (*Model*), negocio (*ViewModel*) y presentación (*View*).

#### 2.2.2.1 Capa modelo (*Model*)

Esta capa contiene los tipos de datos con los que trabajará la aplicación. En la Figura 2.2 se puede observar el diagrama de clases de la capa modelo. Además, el prototipo cuenta

con un sistema de notificaciones perteneciente a la capa modelo y se visualiza en el ANEXO IV.

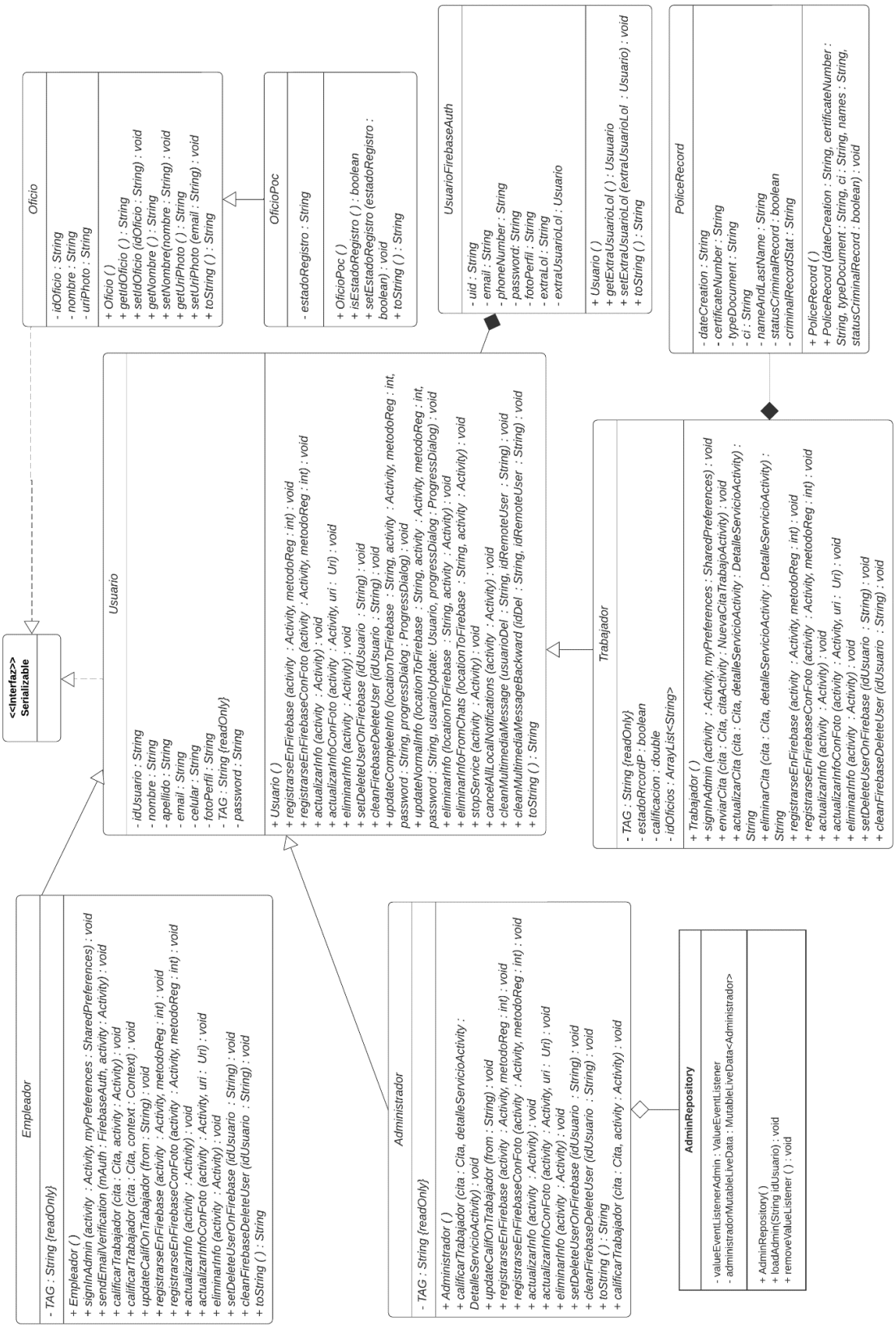



**Figura 2.2.** Diagrama de Clases general de la capa *Model*

### 2.2.2.2 Capa de presentación (*View*)

Esta capa contiene las clases *Activity* o *Fragment* necesarias para construir toda la interfaz gráfica de la aplicación. En la Figura 2.3. se muestra el ejemplo del diagrama de clases para el módulo de inicio de sesión. Además, el prototipo cuenta con otras interfaces pertenecientes a la capa presentación y se localizan en el ANEXO IV.

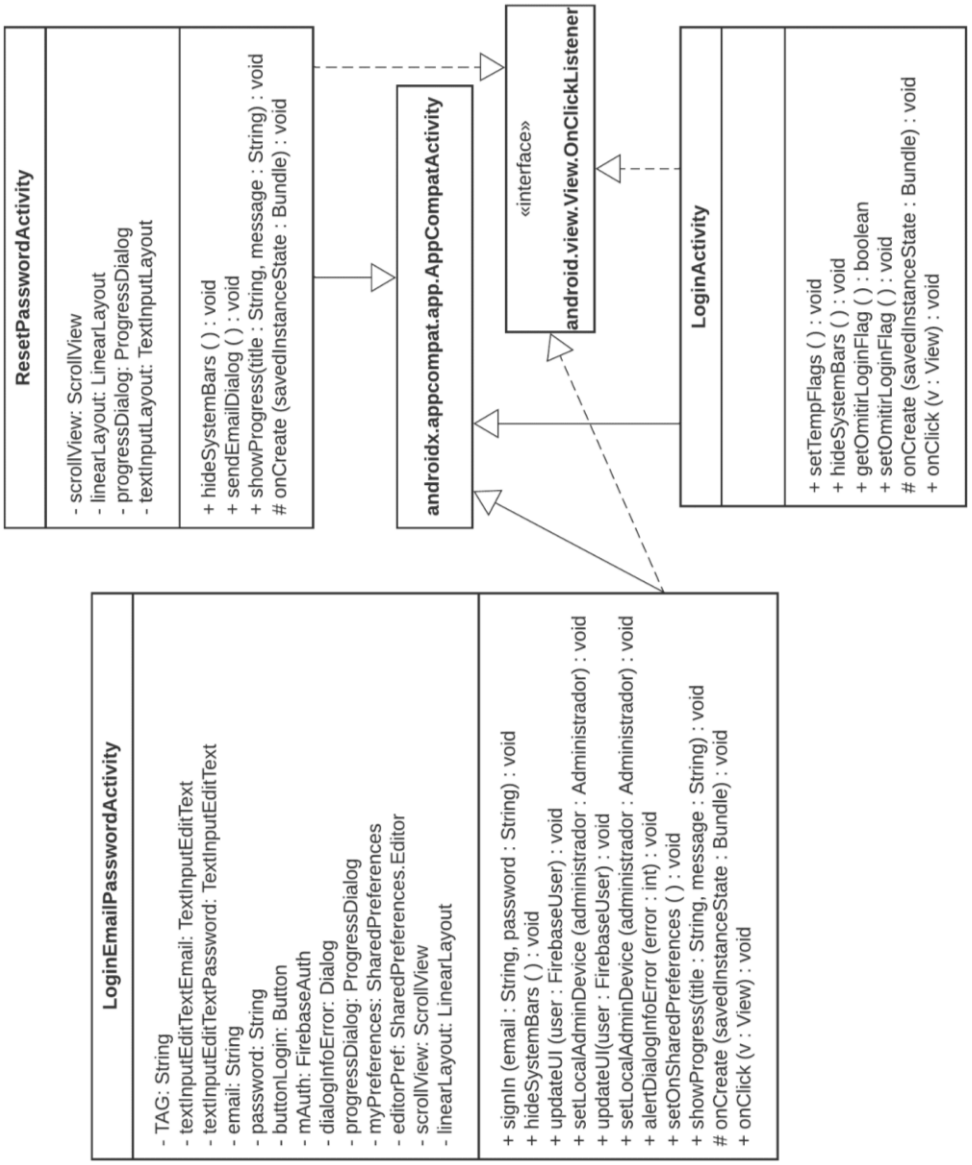



**Figura 2.3.** Diagrama de Clases: Módulo inicio de sesión (capa *View*)

### 2.2.2.3 Capa de negocio (*ViewModel*)

Esta capa contiene la lógica de negocio que se implementa en el prototipo de aplicación móvil (Figura 2.4).

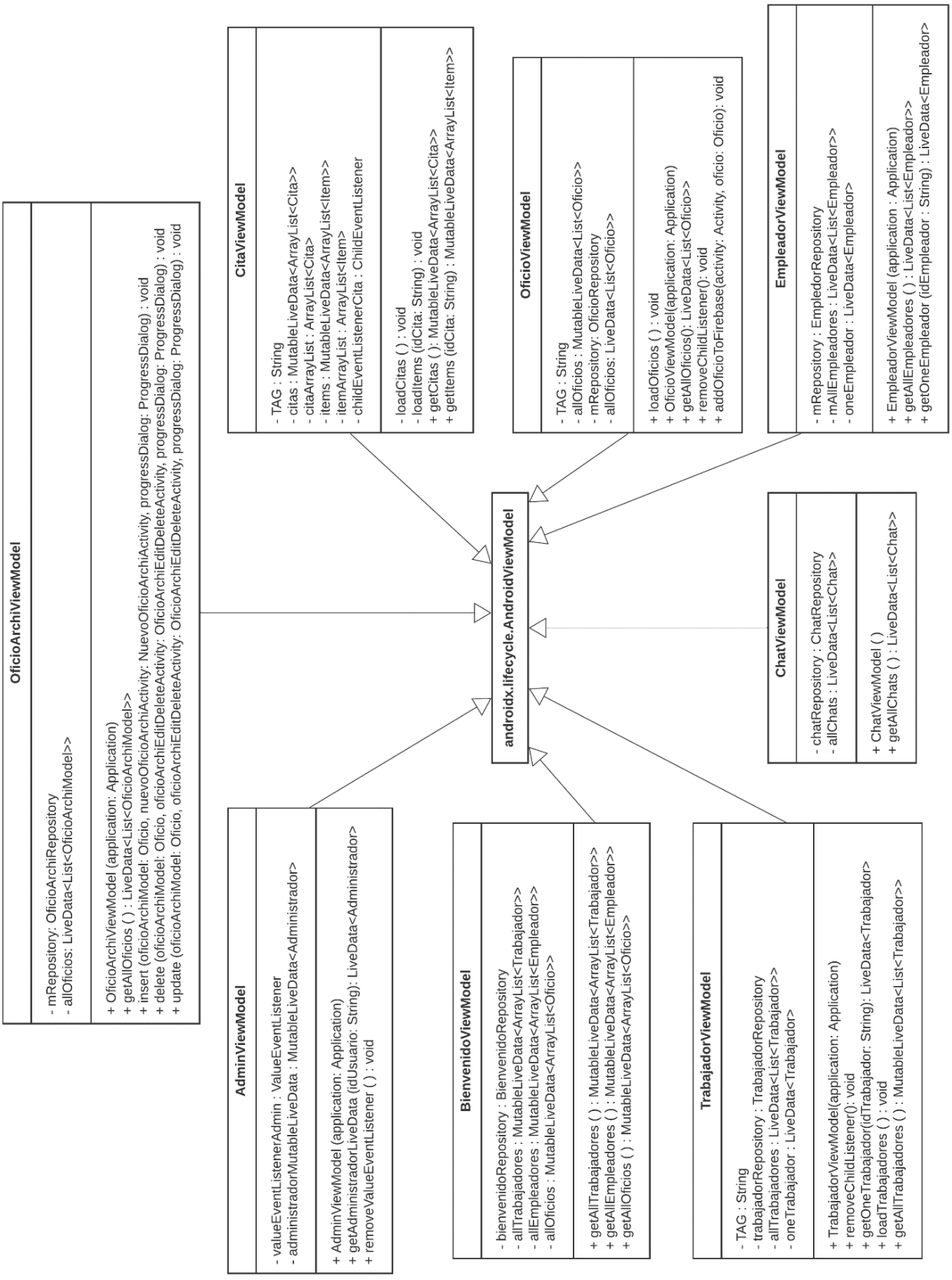



**Figura 2.4.** Diagrama de Clases: Capa ViewModel

### 2.2.3 Diagrama de actividades

En esta sección se incluye el funcionamiento del sistema mediante el uso de diagramas de actividades. En la Figura 2.5 se observa el proceso para iniciar sesión en la aplicación, en este caso se necesita tener una cuenta con un correo electrónico y contraseña para poder utilizar todas las funciones del prototipo de aplicación móvil. Sin embargo, el prototipo cuenta con más de un diagrama de actividades. Estos diagramas se ponen a disposición en el ANEXO V.

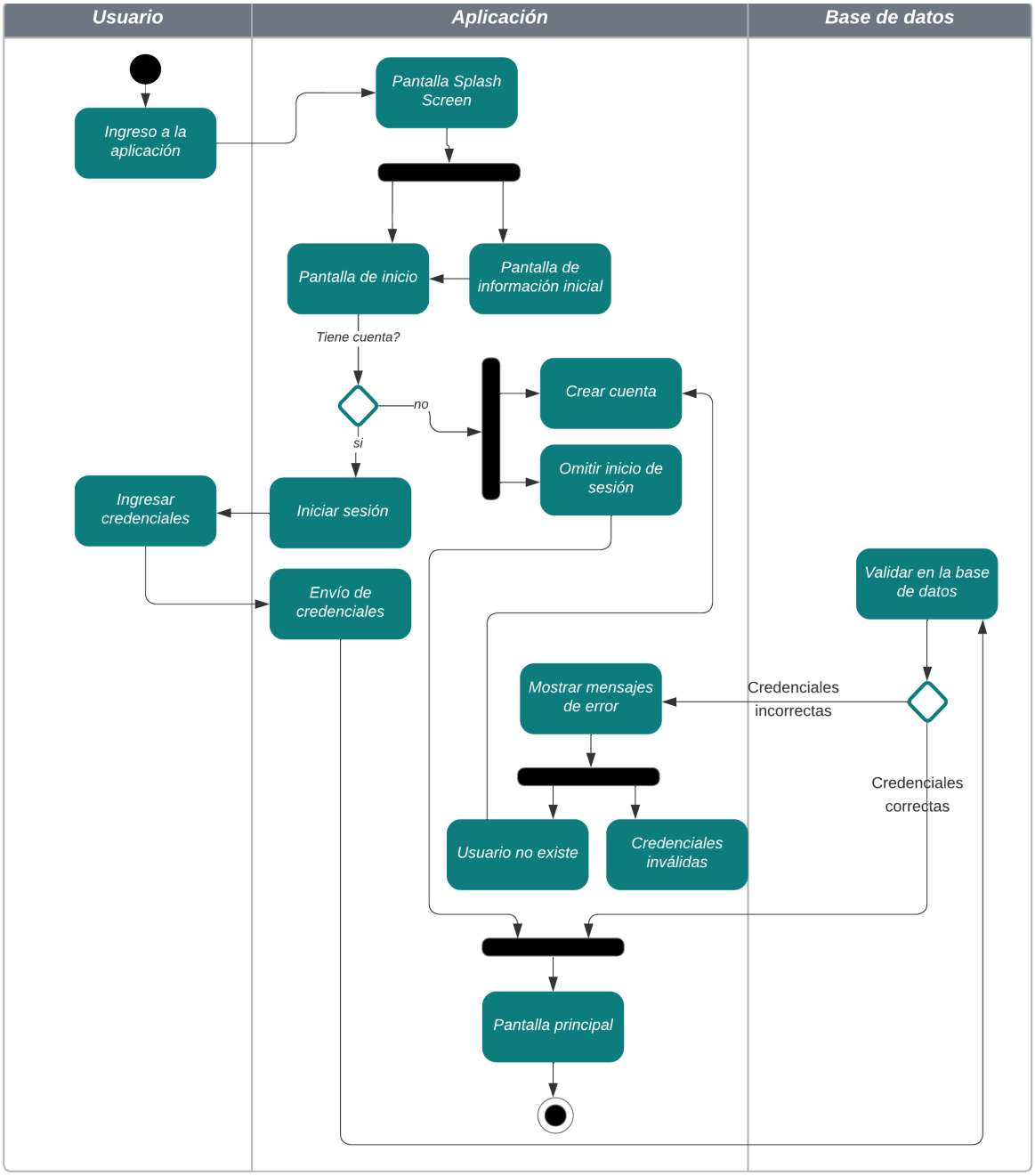



**Figura 2.5.** Diagrama de actividades para el inicio de sesión

### 2.2.4 Módulos de la aplicación

Se ha determinado que el proyecto contará con al menos 6 módulos para su funcionamiento.

#### 2.2.4.1 Módulo De Registro

El módulo de registro permite crear nuevos Trabajadores, Empleadores y Oficios.

- **Registro de Oficios**
  - Registro de nuevo Oficios.
  - Filtrado de duplicados utilizando el nombre del oficio.
- **Registro de Trabajadores**
  - Registro de nuevos Trabajadores
  - Filtrado de duplicados utilizando una cuenta de correo electrónico.
  - Procesamiento del récord policial utilizando imágenes o documentos PDF.
- **Registro de Empleadores**
  - Registro de nuevos Empleadores.
  - Filtrado de duplicados utilizando una cuenta de correo electrónico.

#### 2.2.4.2 Módulo de Inicio de Sesión

Este módulo permite iniciar sesión a los Usuarios en la aplicación e incluye la función de recuperación de contraseña.

- **Autenticación de Usuarios**
  - Se realiza la autenticación utilizando una cuenta de correo electrónico y contraseña.
  - Se utiliza la autenticación para iniciar sesión.
  - Se utiliza la autenticación para la eliminación y actualización de los datos personales de Trabajadores y Empleadores
- **Recuperación de contraseña**

- Se realiza la recuperación utilizando una cuenta de correo electrónico y contraseña, previamente el usuario debe encontrarse registrado en la aplicación.
- La recuperación de contraseña incluye un link enviado al correo electrónico del usuario.

#### 2.2.4.3 Módulo de Búsqueda

El módulo de búsqueda es el encargado de encontrar a los diferentes Trabajadores mediante un nombre de oficio o su nombre y apellido.

- **Buscador**
  - La búsqueda de los trabajadores se la puede realizar utilizando el criterio de un nombre de Oficio o un nombre y apellido.
  - El buscador maneja las solicitudes que no contengan ningún resultado exitoso en la búsqueda.
  - Los resultados de búsqueda contienen la información del perfil de cada Trabajador.

#### 2.2.4.4 Módulo de Notificaciones

El módulo de notificaciones mejora la comunicación entre los usuarios registrados en la aplicación. Las notificaciones emitidas pueden ser: mensajes de otros usuarios, llamadas de voz, videollamadas y citas de trabajo.

- **Notificaciones**
  - Envío y recepción de notificaciones de mensajes de texto y contenido multimedia.
  - Envío y recepción de notificaciones de videollamadas.
  - Envío y recepción de notificaciones de llamadas de voz.
  - Envío y recepción de notificaciones de citas de trabajo.

#### 2.2.4.5 Módulo de Mensajería Instantánea

El módulo de mensajería instantánea permite el intercambio de mensajes entre los diferentes usuarios registrados en la aplicación.

- **Comunicación entre Usuarios**
    - Intercambio de mensajes de texto y contenido multimedia como: audio e imágenes.
    - Control de estados de mensajes leído/no leído.
- **Videollamadas entre Usuarios**
    - Comunicación entre usuarios a través de videollamadas.
    - Control de estados en videollamadas entrantes y salientes.
    - Control de micrófono, altavoz, cámara de video y calidad de video.
- **Llamadas de Voz entre Usuarios**
    - Comunicación entre usuarios a través de llamadas de voz.
    - Control de estados en llamadas entrantes y salientes.
    - Control de micrófono y altavoz.

#### 2.2.4.6 Módulo de Perfil

El módulo de perfil es el encargado de mostrar los datos personales como: correo electrónico, foto de perfil, nombre y apellido, de cada usuario registrado en la aplicación.

- **Administración de datos personales**
    - La actualización de datos personales la realiza únicamente la persona autenticada o el Administrador.
    - El correo electrónico no puede ser modificado.
    - La actualización de datos personales permite modificar información como: nombre, apellido, foto de perfil y oficios registrados en el caso de los Trabajadores.

### 2.2.5 Interfaces de Usuario

Los *sketches* que fueron utilizados para el desarrollo del prototipo se encuentran ubicados en el ANEXO VI. En la Figura 2.6 se observa el ejemplo del *sketch* para la interfaz de usuario que se presenta al ingresar a la aplicación.

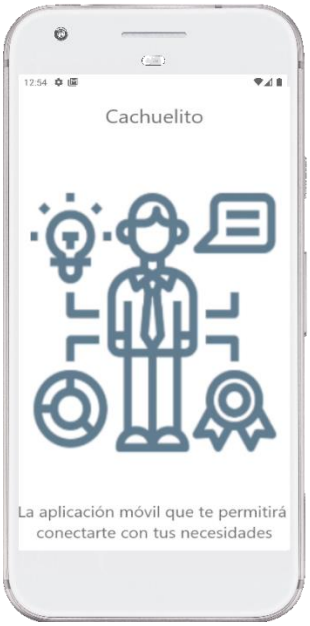



**Figura 2.6.** Sketch de pantalla de inicio de la aplicación

## 2.3 Fase de Implementación

Esta fase abarca la codificación de los distintos diagramas de clases, casos de uso y de actividades mediante el lenguaje Java. De la misma manera, incluye la codificación de los diferentes *sketches* utilizando código XML. En esta fase, se emplean herramientas como Android Studio, Git y *Firebase* para el desarrollo del prototipo de la aplicación móvil.

### 2.3.1 Configuración e instalación de Android Studio, Git y *Firebase* SDK

Para continuar con la implementación del prototipo es necesario instalar y configurar algunas herramientas que permiten facilitar su desarrollo.

#### 2.3.1.1 Instalación de Android Studio

Para instalar Android Studio, el usuario debe dirigirse al enlace proporcionado: https://developer.android.com/studio. Una vez completada la descarga, el usuario debe ejecutar el proceso de instalación del archivo descargado. A continuación, deberá seguir las instrucciones presentadas por el asistente de instalación con el fin de concluir la instalación de manera exitosa. El proceso de instalación se visualiza en el ANEXO IX.

#### 2.3.1.2 Instalación de Git

Para instalar la herramienta del control de versiones Git, se debe descargar Git en el enlace: https://git-scm.com/downloads. Una vez que la descarga haya concluido, será necesario ubicar el archivo descargado y proceder a efectuar doble clic sobre el mismo, con el propósito de dar inicio al asistente de instalación. Con el fin de verificar la correcta instalación de la herramienta Git, se recomienda abrir una terminal y teclear el comando: "git --version", el resultado debe ser algo parecido a lo mostrado en la Figura 2.7.

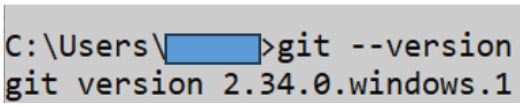



**Figura 2.7.** Verificación de instalación de la herramienta Git

#### 2.3.1.3 Configuración de *Firebase* SDK y Android Studio

Para finalizar la configuración de Android Studio y utilizar las bibliotecas de *Firebase*, es necesario crear un nuevo proyecto en el IDE. Una vez que el proyecto esté configurado, se seleccione la pestaña Tools>Firebase. Esto abrirá el asistente de *Firebase* en el panel derecho del IDE.

Dependiendo de las funcionalidades requeridas por el proyecto, se puede seleccionar varias herramientas. En este caso, se selecciona las siguientes herramientas: *Authentication*, *Realtime Database* y *Firebase ML* (*Machine Learning*). El proceso de configuración de Firebase y Android Studio se encuentra en el ANEXO X.

Es importante destacar que el asistente de *Firebase* viene preinstalado en la versión de Android Studio Chipmunk 2021.2.1, que se utiliza para el desarrollo de este proyecto. Además, cada vez que se seleccione una nueva herramienta de *Firebase* después de haber registrado la aplicación en dicha plataforma, el IDE le preguntará al desarollador mediante un cuadro de diálogo si desea incluir las dependencias en el proyecto. Si acepta, el archivo *Gradle* que contiene las dependencias se actualizará y sincronizará automáticamente, permitiendo utilizar las diferentes funcionalidades del SDK de Firebase.

#### 2.3.1.4 Registro de la aplicación en Firebase

Para crear un proyecto en *Firebase* es necesario tener a disposición una cuenta en la plataforma de *Gmail*. Luego de creada la cuenta en *Gmail* se puede empezar a vincular un proyecto de Android con *Firebase*. En este caso como ya se ha desplegado anteriormente en el explorador web la consola de *Firebase*, lo que procede es configurar el proyecto en dicha plataforma con la ayuda del asistente y como resultado final se obtiene la vinculación

de Android Studio y *Firebase*. Luego de la vinculación con *Firebase*, finalmente se pueden utilizar las bibliotecas de la plataforma seleccionando la opción *Add the Realtime Database SDK to your app* o *Add the Firebase Authentication to your app.*

Para la creación de la base de datos, se debe escoger el proyecto dentro de la plataforma *Firebase*. Una vez seleccionado el proyecto, se debe seleccionar la pestaña Compilación y escoger el módulo de *Realtime Database* para la creación de la base de datos, y *Authentication* para el inicio de sesión, como se muestra en la Figura 2.8. Los dos módulos son necesarios para poder iniciar con el desarrollo del prototipo.

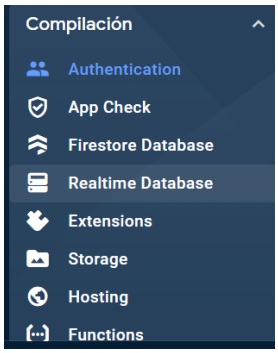



**Figura 2.8.** Pestaña de compilación dentro del proyecto en Firebase.

### 2.3.2 Control de iteraciones

El control de iteraciones permite documentar los errores que pueden surgir durante el desarrollo del prototipo y realizar un control de las historias de usuario que se van desarrollando. Como nomenclatura se utiliza H1 para indicar la o las historias de usuario que se han desarrollado en cada iteración. En el caso anterior H1 indica que la historia de usuario es la número uno.

#### 2.3.2.1 Primera iteración

En la primera iteración se completaron las historias de usuario H1 y H2. Como observaciones se tiene que no existieron novedades durante el desarrollo de las historias de Usuario.

En la Figura 2.9 (a) se presenta la implementación de la interfaz de inicio de la aplicación, esta interfaz es mostrada cada que el usuario inicia la aplicación. La Figura 2.9 (b) muestra la interfaz de los diferentes métodos de acceso a la aplicación, en este caso se puede crear una cuenta, omitir el inicio de sesión o iniciar sesión con una cuenta previamente creada.


**Figura 2.9.** Interfaz inicial (a) e interfaz de métodos de acceso a la aplicación (b)

En la Figura 2.10 se encuentra la implementación de las diferentes interfaces gráficas que permiten el registro de nuevos oficios y la visualización de estos dentro de la aplicación. La Figura 2.10.(a) permite visualizar los oficios que el usuario trabajador puede escoger al momento de realizar su registro. Mientras que la Figura 2.10 (b) muestra la interfaz para el registro de un nuevo oficio por parte del usuario Trabajador y la Figura 2.10 (c) presenta la interfaz para el registro de un nuevo oficio por parte del usuario Administrador.

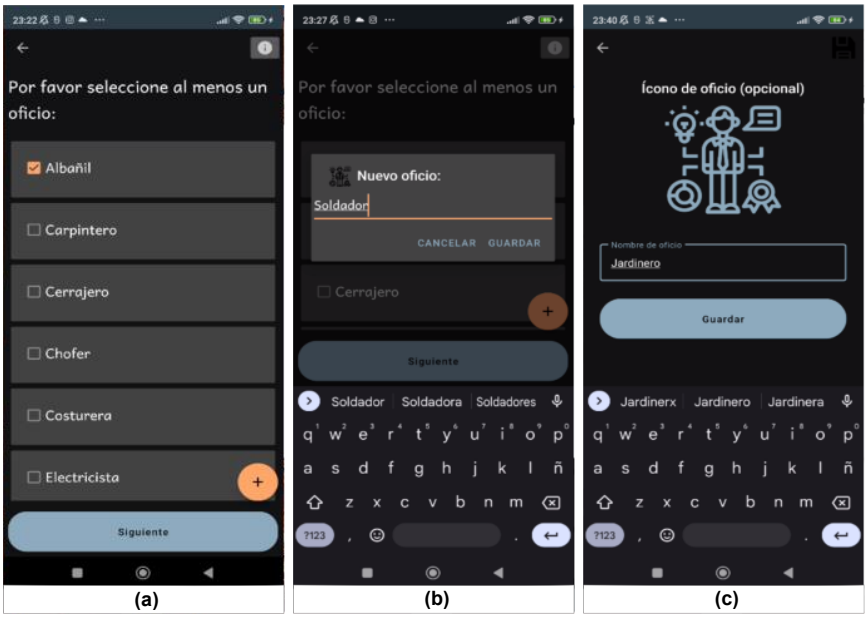



**Figura 2.10.** Interfaces gráficas para la visualización y registro de oficios.

El código que permite guardar nuevos oficios en *Firebase Realtime Database* se encuentran en la Figura 2.11. El método *public void addOficioTofirebase(Activity activity, Oficio oficio) { }* recibe dos parámetros. El primer parámetro es un objeto *Activity* que permite interactuar con la interfaz gráfica que el usuario se encuentra observando al momento de realizar el registro de un nuevo oficio. El segundo parámetro es un objeto "Oficio" detallado en el diagrama de clases de la capa modelo de la aplicación ubicado en la Figura 2.2. Además, este método contiene un mensaje de confirmación en caso de un registro exitoso o fallido.

```
public void addOficioTofirebase(Activity activity, Oficio oficio) {
    String idOficio = FirebaseDatabase.getInstance().getReference().child("oficios").push().getKey();
    oficio.setIdOficio(idOficio);
    FirebaseDatabase.getInstance().getReference()
            .child("oficios")
            .child(idOficio)
            .setValue(oficio)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText(activity,  text: "Registro exitoso", Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(activity,  text: "Registro fallido", Toast.LENGTH_SHORT).show();
                    }
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(activity,  text: "Registro fallido: " + e.toString(), Toast.LENGTH_SHORT).show();
                }
            });
}
```

**Figura 2.11.** Código para registrar nuevos oficios en Firebase

### 2.3.2.2 Segunda iteración

En esta iteración se completaron las historias de usuario H3 y H4. Como observaciones se tiene que el mensaje de verificación del correo electrónico no funciona. Esto quiere decir que el usuario no recibe el mensaje de confirmación en su correo electrónico. A continuación, se muestran las implementaciones de las diferentes gráficas para el registro de trabajadores y empleadores. En la Figura 2.12 (a) se observa la interfaz que permite seleccionar al nuevo usuario un perfil que se adapte a sus necesidades; mientras que en la Figura 2.12 (b) se observa la interfaz del registro de datos personales como nombre, apellido y opcionalmente una foto de perfil.

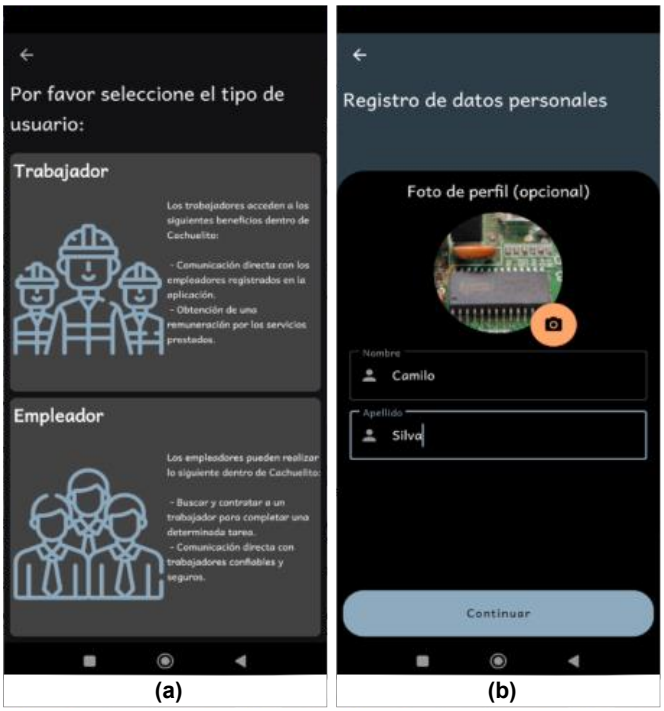



**Figura 2.12.** Interfaces de selección de perfil (a) y registro de datos personales (b).

En la Figura 2.13 (a) se encuentra la interfaz que permite el registro del récord policial del usuario Trabajador. La Figura 2.13 (b) presenta un mensaje donde se visualizan los datos que fueron recopilados del récord policial para su posterior validación. En la Figura 2.13 (c) se visualiza la interfaz gráfica que le permite al usuario Trabajador seleccionar los oficios para los cuales está cualificado.

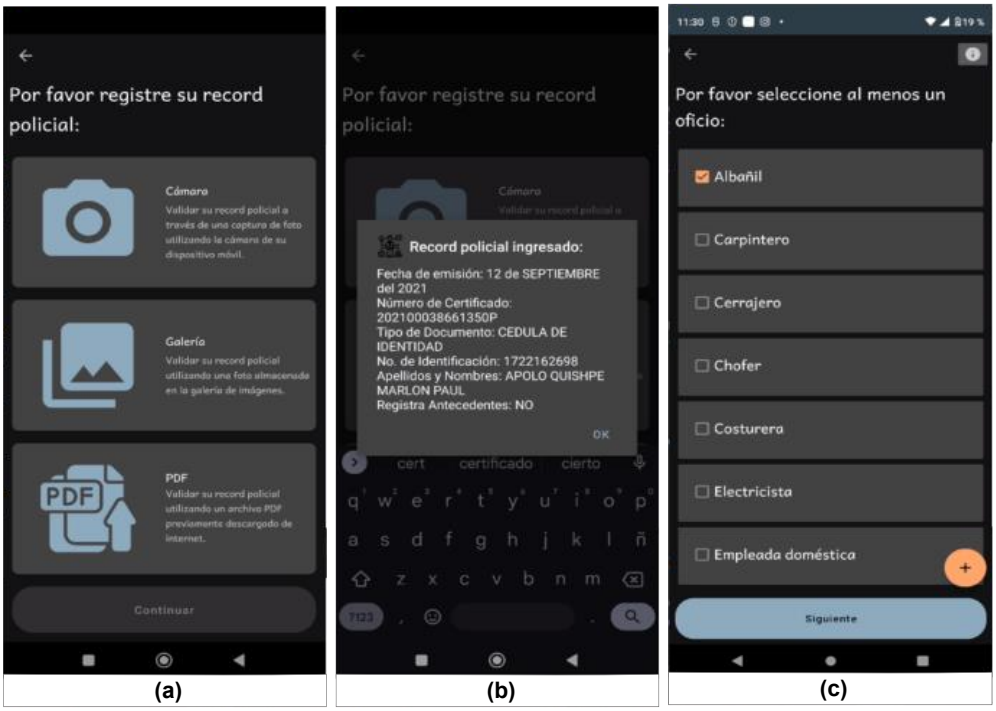


**Figura 2.13.** Interfaces de registro (a) y validación del record policial(b), selección de oficios (c).

En la Figura 2.14 se observa el método *public void registrarseEnFirebase(Activity activity) { }* que permite registrar nuevos empleadores en *Firebase*. El parámetro de entrada *Activity* permite controlar las acciones del usuario a través de la interfaz gráfica. Además, el método contiene un mensaje de confirmación en caso de un registro exitoso o fallido.

```
public void registrarseEnFirebase(Activity activity) {
    SharedPreferences myPreferences = activity.getSharedPreferences( name: "MyPreferences", MODE_PRIVATE);
    boolean adminFlag = myPreferences.getBoolean( key: "adminFlag",  defValue: false);
    FirebaseDatabase.getInstance().getReference().child("empleadores").child(this.getIdUsuario()).setValue(this)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText(activity,  text: "Registro exitoso", Toast.LENGTH_LONG).show();
                        sendEmailVerification(FirebaseAuth.getInstance(), activity);
                                try {...} catch (Exception e) {
                                    Log.d(TAG, e.toString());
                                }
                        activity.finishAffinity();
                        Intent intent = new Intent(activity, MainNavigationActivity.class);
                        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        activity.startActivity(intent);
                        try {...} catch (Exception e) {
                            Log.d(TAG, e.toString());
                        }
                    } else {
                        Toast.makeText(activity, "Se ha producido un error inesperado.  Por favor intént...", Toast.LENGTH_SHORT).sho
                    }
                }
```

**Figura 2.14.** Código de registro para un empleador en Firebase

### 2.3.2.3 Tercera iteración

Se han finalizado las historias de usuario H5 y H6. Como observaciones se tiene la corrección del envío del mensaje de verificación por correo electrónico. Los nuevos usuarios reciben el correo de confirmación sin ningún inconveniente.

Para la autenticación de usuarios se utiliza la herramienta *Firebase Authentication*. En la Figura 2.15 se observan las diferentes interfaces gráficas para la autenticación e inicio de sesión de los usuarios. La Figura 2.15 (a) presenta la interfaz gráfica del menú principal de la aplicación, si el usuario ha decidido omitir el inicio de sesión. La Figura 2.15 (b) muestra la interfaz gráfica de inicio de sesión utilizando un correo electrónico y una contraseña. En la Figura 2.15 (c) se tiene la interfaz gráfica que permite realizar una recuperación de contraseña utilizando un correo electrónico. El mensaje de recuperación de contraseña es enviado al correo electrónico ingresado.

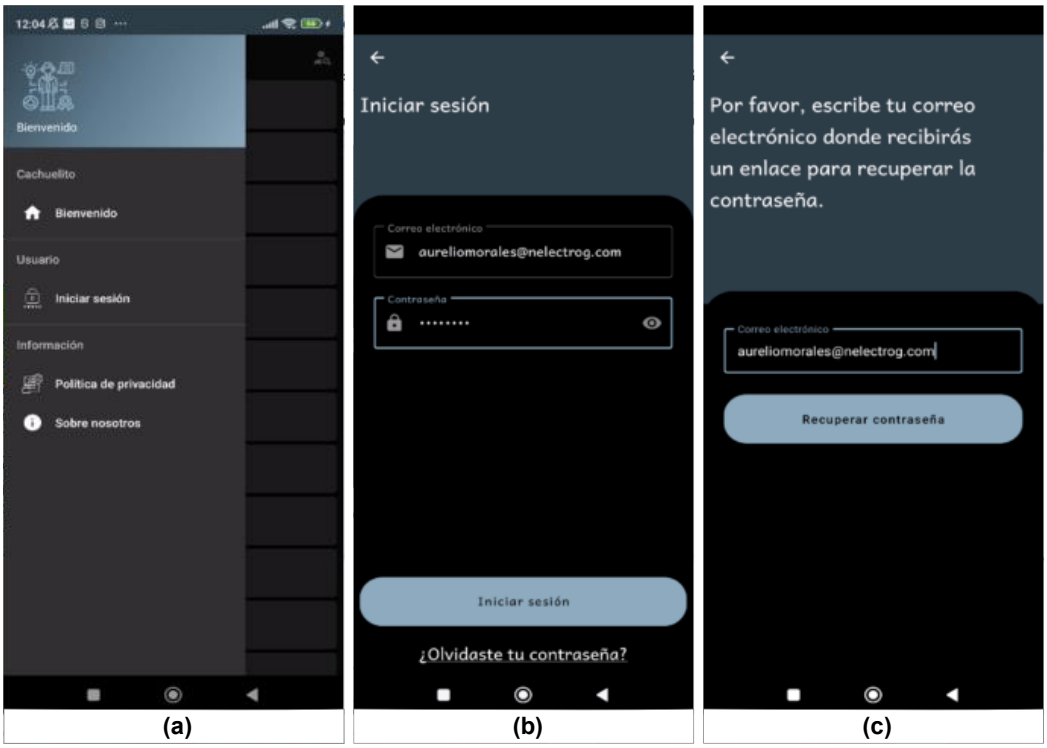



**Figura 2.15.** Interfaces de menú principal (a), inicio de sesión (b) y recuperación de contraseña (c).

En la Figura 2.16 se encuentra la implementación del código que permite autenticar al usuario cuando este desea iniciar sesión. Este método recibe dos objetos del tipo *String* como parámetros de entrada. El primer parámetro es el e-mail del usuario y el segundo parámetro su contraseña. Además, el método tiene un control de errores interno que se encarga de informar al usuario si se ha producido algún error cuando intenta iniciar sesión.

```
public void signIn(String email, String password) {
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        FirebaseUser user = mAuth.getCurrentUser();
                        updateUI(user);
                    } else {
                        String errorCode = ((FirebaseAuthException) Objects
                                .requireNonNull(task.getException())).getErrorCode();
                        try {
                            progressDialog.dismiss();
                        } catch (Exception e) {
                            Log.e(TAG, e.toString());
                        }
                        switch (errorCode) {...}
                    }
                }
            });
}
```

**Figura 2.16.** Código de autenticación para inicio de sesión

La implementación del código que permite actualizar la información personal de un Trabajador se encuentra en la Figura 2.17. El método *public void actualizarInfo(Activity activity) { }* recibe como parámetro de entrada un objeto *Activity*, que representa la interfaz gráfica que el usuario se encuentran observando al momento de actualizar su información personal. El método actualiza la información únicamente cuando los datos del usuario han sufrido alguna modificación. Además, la actualización de datos personales se la realiza únicamente cuando el usuario (Trabajador, Empleador o Administrador) ha modificado algún campo de su información.

```
public void actualizarInfo(Activity activity) {
    FirebaseDatabase.getInstance().getReference()
            .child("trabajadores").child(this.getIdUsuario()).setValue(this)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    Toast.makeText(activity,  text: "Infomación actualizada", Toast.LENGTH_LONG).show();
                    try {
                        EditarDataActivity editarDataActivity = (EditarDataActivity) activity;
                        editarDataActivity.closeProgressDialog();
                    } catch (Exception e) {
                        Log.d(TAG, e.toString());
                    }
                }
            });
}
```

**Figura 2.17.** Código para actualizar los datos personales del usuario Trabajador

La Figura 2.18 (a) presenta la implementación de la interfaz gráfica que contiene los datos personales del usuario Trabajador. Como se mencionó anteriormente, los datos que se pueden editar son: nombres, apellido, foto de perfil y oficios registrados. Además, la interfaz incluye un submenú ubicado en la parte derecha que permite actualizar los oficios en caso de que el usuario Trabajador necesite modificar esa información y, proporciona información sobre los trabajos completos, incompletos, asistidos y no asistidos. En la Figura 2.18 (b) se muestra la interfaz gráfica que permite actualizar los oficios del usuario Trabajador.

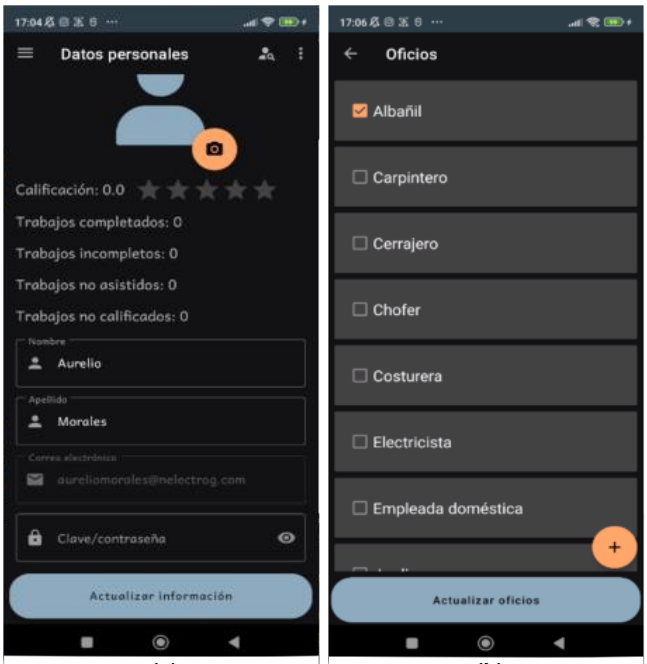



**Figura 2.18.** Interfaces de actualización de datos personales (a) y oficios registrados (b)

#### 2.3.2.4 Cuarta iteración

Se desarrolla la historia de usuario H7. Como observaciones se tiene que se produjo un error en la interfaz gráfica que muestra la información personal del usuario. El error consiste en que no se muestra la imagen del usuario dentro del objeto *ImageView*, es decir no se visualiza la foto de perfil.

En esta iteración se implementaron las interfaces gráficas mostradas en la Figura 2.19. La Figura 2.19 (a) muestra el buscador presentado al usuario que no se encuentra logeado en la aplicación. Esta interfaz no contiene acciones que permitan al usuario interactuar con los Trabajadores registrados. En la Figura 2.19 (b) se visualiza la interfaz del buscador para un usuario logeado en la aplicación. Además, en la Figura 2.19 (c) se observa el buscador con el resultado de la búsqueda de un trabajador y las diferentes acciones que un usuario logeado como Empleador puede utilizar en el prototipo de aplicación.

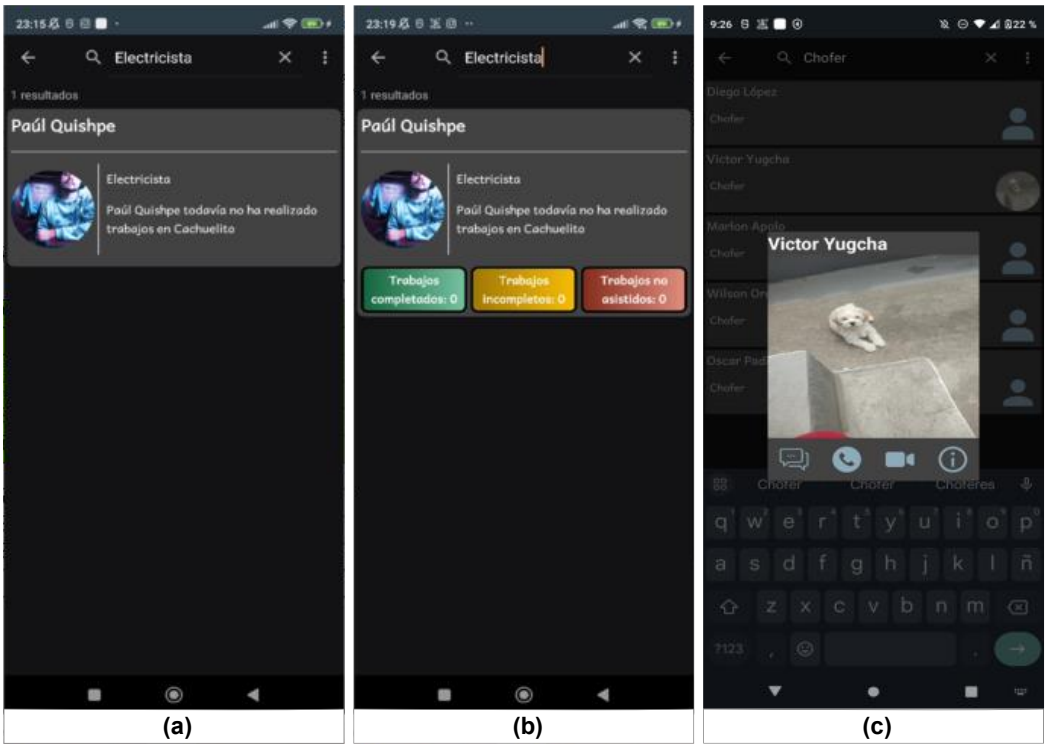



**Figura 2.19.** Interfaces gráficas para la búsqueda de trabajadores

En la Figura 2.20 se presenta el código para buscar trabajadores dentro de *Firebase*. El método recibe el id del nombre del oficio que el usuario se encuentra buscando y filtra los resultados utilizando el nodo trabajadores dentro de *Firebase Realtime Database*.

```
public void searchTrabajadores(String idOficio) {
    valueEventListenerTrabajadores = new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            ArrayList<Trabajador> trabajadorArrayList = new ArrayList<>();
            for (DataSnapshot data : snapshot.getChildren()) {
                Trabajador trabajador = data.getValue(Trabajador.class);
                for (String idOf : trabajador.getIdOficios()) {
                    if (idOf.equals(idOficio)) {
                        trabajadorArrayList.add(trabajador);
                        break;
                    }
                }
            }
            showTrabajadores(trabajadorArrayList);
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    };
    FirebaseDatabase.getInstance().getReference()
            .child("trabajadores")
            .addValueEventListener(valueEventListenerTrabajadores);
}
```

**Figura 2.20.** Código de búsqueda de trabajadores

### 2.3.2.5 Quinta iteración

En la quinta iteración se completó la historia de usuario H8. Como observaciones se corrigió el error de la foto de perfil en la interfaz gráfica de los datos personales. Además, se presentaron errores al intentar compartir la ubicación de los usuarios utilizando el chat.

En la Figura 2.21 (a) se visualiza la implementación de la interfaz gráfica del menú principal que además permite seleccionar la lista de chats. La Figura 2.21 (b) muestra la interfaz gráfica que alberga los diferentes chats que son creados entre los usuarios de la aplicación; mientras que en la Figura 2.21 (c) se observa la interfaz gráfica para el envío y recepción de mensajes entre Trabajadores y Empleadores.

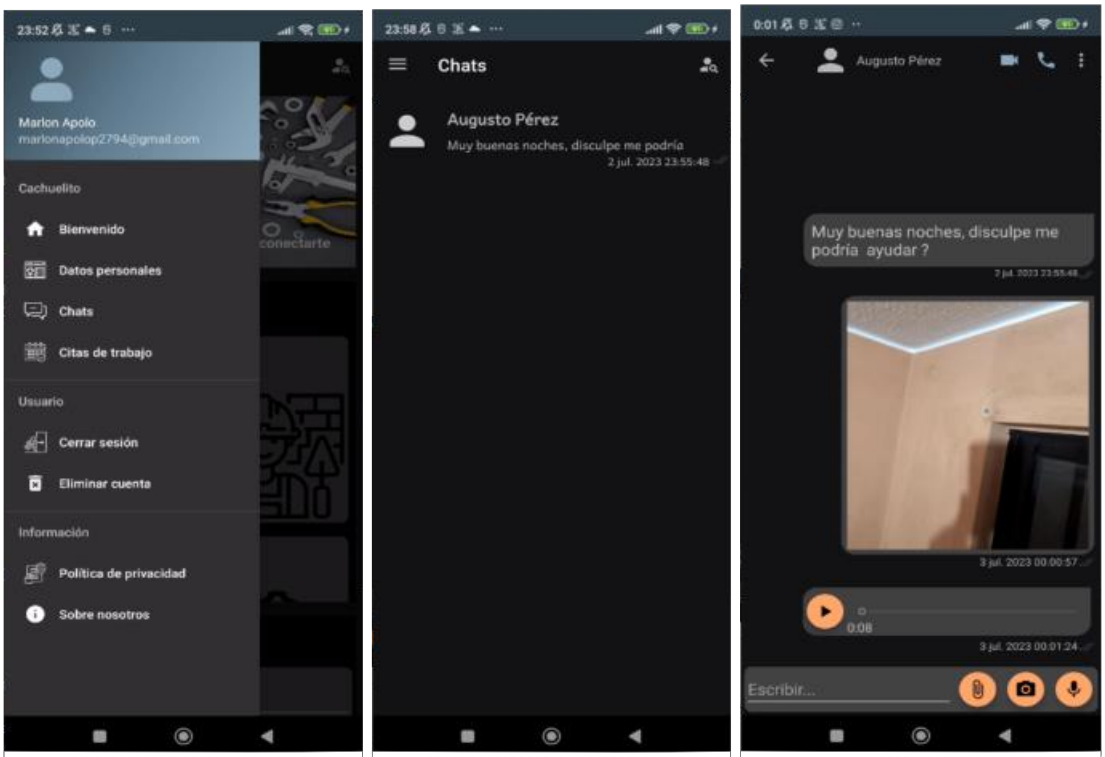



**Figura 2.21.** Interfaz de menú principal (a), lista de chats (b) y chat individual (c).

El método para enviar un mensaje de texto o multimedia se encuentra en la Figura 2.22. Este método recibe un objeto *MessageCloudPoc* como parámetro de entrada, el cual contiene toda la información del mensaje que el usuario ha enviado.

```
public void sendMessage(MessageCloudPoc messageCloudPoc) {
    Log.d(TAG,  msg: "############################");
    Log.d(TAG,  msg: "sendMessage");
    Log.d(TAG, messageCloudPoc.toString());
    Log.d(TAG,  msg: "############################");
    Timestamp timestamp = new Timestamp(new Date());
    messageCloudPoc.setTimeStamp(timestamp.toString());

    String key = FirebaseDatabase.getInstance().getReference()
            .child("crazyMessages").push().getKey();
    messageCloudPoc.setIdMensaje(key);
    Map<String, Object> postValues = messageCloudPoc.toMap();

    Map<String, Object> childUpdates = new HashMap<>();
    childUpdates.put("/crazyMessages/" + messageCloudPoc.getFrom()
            + "/" + messageCloudPoc.getTo() + "/" + key, postValues);
    childUpdates.put("/crazyMessages/" + messageCloudPoc.getTo()
            + "/" + messageCloudPoc.getFrom() + "/" + key, postValues);
    childUpdates.put("/notificaciones/" + messageCloudPoc.getTo()
            + "/" + messageCloudPoc.getFrom() + "/" + key, postValues);

    FirebaseDatabase.getInstance().getReference().updateChildren(childUpdates).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            Log.d(TAG,  msg: "Mensaje enviado");
        } else {
            Log.d(TAG,  msg: "Error al enviar mensaje");
        }
    });
```

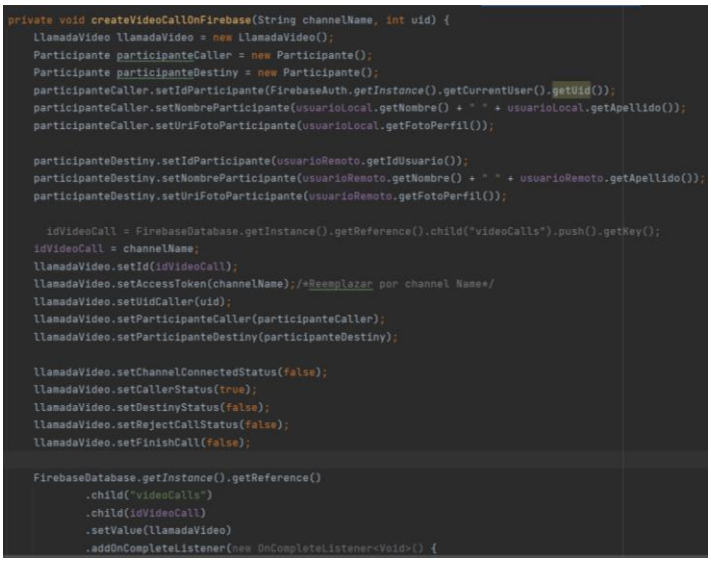
**Figura 2.22.** Código para el envío y almacenamiento de mensajes en Firebase

#### 2.3.2.6 Sexta iteración

En la sexta iteración se completó la historia de usuario H9. En la Figura 2.23 se muestra la interfaz gráfica para la realización de videollamadas. La interfaz contiene botones que permiten activar/desactivar micrófono, cámara de video, altavoz, controlar la calidad del video y colgar la videollamada.


**Figura 2.23.** Interfaz gráfica para realizar y contestar videollamadas

En la Figura 2.24 se observa el código que posibilita la realización de videollamadas. El método "*createVideoCallOnFirebase*" acepta dos parámetros de entrada. El primero corresponde al nombre del canal donde se establecerá la videollamada, mientras que el segundo parámetro se refiere al identificador del usuario con el cual se desea establecer la comunicación.



```
private void createVideoCallOnFirebase(String channelName, int uid) {
    LlamadaVideo llamadaVideo = new LlamadaVideo();
    Participante participanteCaller = new Participante();
    Participante participanteDestiny = new Participante();
    participanteCaller.setIdParticipante(FirebaseAuth.getInstance().getCurrentUser().getUid());
    participanteCaller.setNombreParticipante(usuarioLocal.getNombre() + " " + usuarioLocal.getApellido());
    participanteCaller.setUriFotoParticipante(usuarioLocal.getFotoPerfil());

    participanteDestiny.setIdParticipante(usuarioRemoto.getIdUsuario());
    participanteDestiny.setNombreParticipante(usuarioRemoto.getNombre() + " " + usuarioRemoto.getApellido());
    participanteDestiny.setUriFotoParticipante(usuarioRemoto.getFotoPerfil());

      idVideoCall = FirebaseDatabase.getInstance().getReference().child("videoCalls").push().getKey();
    idVideoCall = channelName;
    llamadaVideo.setId(idVideoCall);
    llamadaVideo.setAccessToken(channelName);/*Reemplazar por channel Name*/
    llamadaVideo.setUidCaller(uid);
    llamadaVideo.setParticipanteCaller(participanteCaller);
    llamadaVideo.setParticipanteDestiny(participanteDestiny);

    llamadaVideo.setChannelConnectedStatus(false);
    llamadaVideo.setCallerStatus(true);
    llamadaVideo.setDestinyStatus(false);
    llamadaVideo.setRejectCallStatus(false);
    llamadaVideo.setFinishCall(false);

    FirebaseDatabase.getInstance().getReference()
            .child("videoCalls")
            .child(idVideoCall)
            .setValue(llamadaVideo)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
```

**Figura 2.24.** Código para realizar una videollamada

### 2.3.2.7 Séptima iteración

En la séptima iteración se completó la historia de usuario H10. La Figura 2.25 contiene la interfaz gráfica que permite realizar y contestar llamadas de voz. Esta interfaz contiene botones para activar/desactivar micrófono, altavoz y para colgar la llamada de voz.

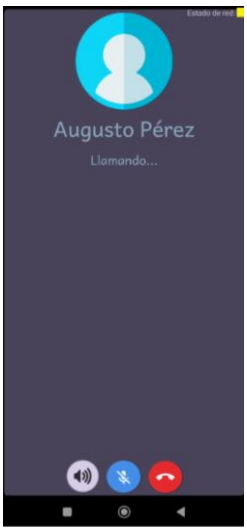


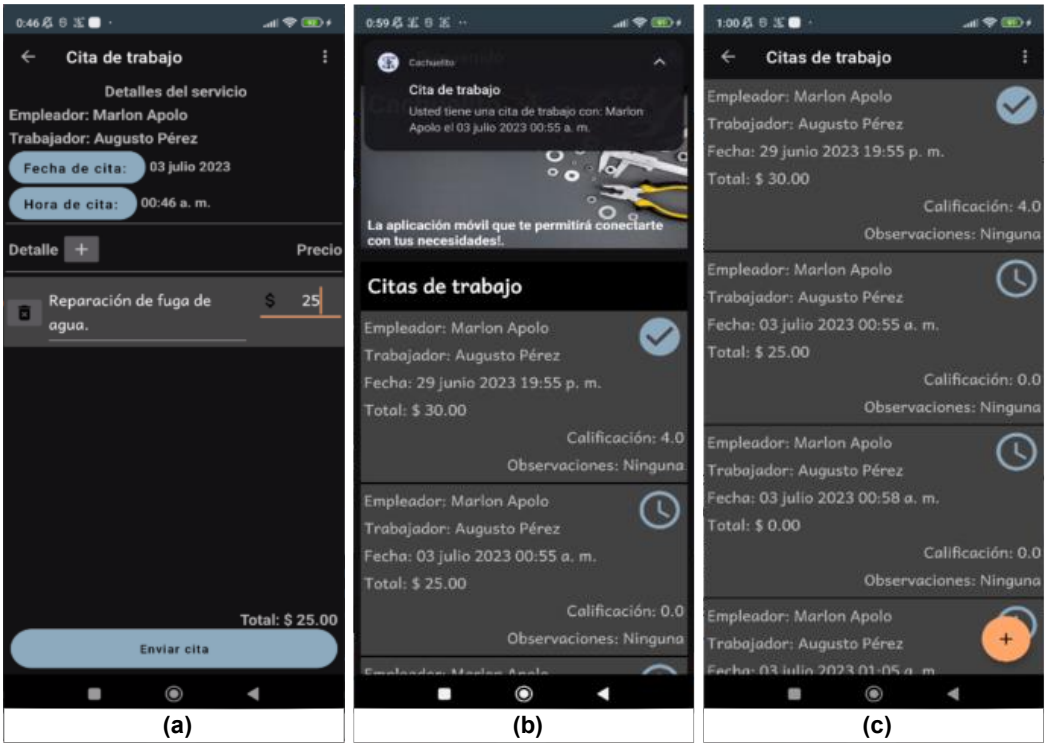

**Figura 2.25.** Interfaz gráfica para realizar y contestar llamadas de voz.

Como observaciones se tiene que las llamadas de voz presentan errores al intentar establecer la comunicación.

#### 2.3.2.8 Octava iteración

En la octava iteración se completó la historia de usuario H11. Como observaciones se ha corregido el error en las llamadas de voz, pero se presentaron errores al momento de responder las notificaciones mensajes.

En la Figura 2.26 (a) se visualiza la interfaz gráfica que permite la creación de nuevas notificaciones de citas de trabajo. En la Figura 2.26 (b) se observa la implementación de una notificación de cita de trabajo, estas notificaciones sirven como recordatorios a los usuarios en caso de que se encuentren trabajos pendientes. Además, la Figura 2.26 (c) muestra la lista de notificaciones de citas de trabajo que se han generado durante el uso de la aplicación.




(a) (b) (c)

**Figura 2.26.** Interfaz para creación de notificaciones de citas (a), notificación de cita de trabajo (b) y lista de notificaciones de citas (c)

# 3 PRUEBAS, RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

## 3.1 Actualización del plan de entrega

En la Tabla 3.1 se presenta la actualización del plan de entrega del proyecto. La actualización incluye dos historias de usuarios que permiten administrar la información de los usuarios y los oficios registrados dentro de la aplicación.

**Tabla 3.1.** Actualización del plan de entrega del proyecto

| Módulo | Nro. | Nombre de Historia | Esfuerzo de desarrollo | | | Calendario Estimado | | | Iteración Asignada | | | | | | | | | | Entrega Asignada | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Semanas ideales | Días ideales | Horas ideales | Semanas estimadas | Días estimados | Horas estimadas | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 |
| Todos | 1 | Interfaces de Usuario | 2 | 10 | 80 | 2 | 10 | 80 | x | | | | | | | | x | | | |
| Módulo de Registro | 2 | Registro de Oficios | 1 | 5 | 40 | 1 | 5 | 32 | x | | | | | | | | x | | | |
| | 3 | Registro de Trabajadores | 1,6 | 8 | 64 | 1,6 | 8 | 48 | | x | | | | | | | x | | | |
| | 4 | Registro de Empleadores | 1,4 | 7 | 56 | 1,4 | 7 | 48 | | x | | | | | | | x | | | |
| Módulo de Inicio de Sesión | 5 | Autenticación de Usuarios | 1,8 | 9 | 72 | 1,8 | 9 | 72 | | | x | | | | | | | x | | |
| Módulo de Perfil | 6 | Administración de Datos Personales | 1,2 | 6 | 48 | 1,2 | 6 | 40 | | | x | | | | | | | x | | |
| Módulo de Búsqueda | 7 | Buscador | 2 | 10 | 80 | 2 | 10 | 80 | | | | x | | | | | | x | | |
| Módulo de Mensajería Instantánea | 8 | Comunicación entre Usuarios | 2 | 10 | 80 | 2 | 10 | 72 | | | | | x | | | | | | x | |
| | 9 | Videollamadas entre Usuarios | 2 | 10 | 80 | 2 | 10 | 80 | | | | | | x | | | | | x | |
| | 10 | Llamadas de Voz entre Usuarios | 2 | 10 | 80 | 2 | 10 | 80 | | | | | | | x | | | | x | |
| Módulo de Notificaciones | 11 | Notificaciones | 2 | 10 | 80 | 2 | 10 | 48 | | | | | | | | x | | | | x |
| Módulo de Administración | 12 | Administración de Usuarios | 1 | 5 | 40 | 1 | 5 | 40 | | | | | | | | x | | | | x |
| | 13 | Administración de Oficios | 1 | 5 | 40 | 1 | 5 | 40 | | | | | | | | x | | | | x |
| | | | | | | | | Total de semanas | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 4 | 1 | 2 | 3 | 4 |
| | | | | | | | | | | | | | | | | | | 21 | | |

## 3.2 Pruebas y resultados

En esta sección se presentan las pruebas unitarias, de instrumentación, de funcionamiento y de validación, que se llevaron a cabo para el desarrollo del prototipo de la aplicación móvil. En las pruebas unitarias, se verifica la coherencia de la lógica interna de la aplicación y la correcta encapsulación de los datos utilizados. En el segmento dedicado a las pruebas instrumentadas, se emplean emuladores o dispositivos físicos para simular las acciones generadas por los usuarios, como pulsar el botón de inicio y luego iniciar la aplicación Cachuelito. Además, se llevan a cabo pruebas de funcionamiento y validación utilizando dispositivos móviles reales como el Motorola Moto G60 y el Xiaomi Redmi Note 9s. Finalmente, se detallan los resultados obtenidos de las pruebas de validación realizadas en un grupo de 11 posibles usuarios de la aplicación móvil.

### 3.2.1 Pruebas unitarias

Las pruebas unitarias fueron desarrolladas utilizando el framework JUnit. El cual permite crear estas pruebas que se compilan y ejecutan en la JVM (*Java Virtual Machine*), es decir estas prueba no necesitan de acceso a un emulador o dispositivo móvil, por lo que generalmente son utilizadas para probar la lógica interna de la aplicación.

En la Figura 3.1 se presenta el código de la prueba unitaria relacionada con el encapsulamiento de los datos que utiliza el usuario Trabajador. El método *testData()* permite comprobar el funcionamiento interno de los métodos *getter* y *setter* del usuario Trabajador.

```
@Test
public void testData() {
    trabajador.setIdUsuario("asd35uhjma6u322");
    trabajador.setNombre("Marlon");
    trabajador.setApellido("Apolo");
    trabajador.setEmail("marlonapolo@gmail.com");
    trabajador.setCalificacion(2.0);
    trabajador.setEstadoRrcordP(false);

    assertThat(trabajador.getNombre(), is(equalTo( operand: "Marlon")));
    assertThat(trabajador.getApellido(), is(equalTo( operand: "Apolo")));
    assertThat(trabajador.getEmail(), is(equalTo( operand: "marlonapolo@gmail.com")));
    assertThat(trabajador.getCalificacion(), is(equalTo( operand: 2.0)));
    assertThat(trabajador.isEstadoRrcordP(), is(equalTo( operand: false)));
}
```

**Figura 3.1.** Prueba unitaria para métodos getter y setter del usuario Trabajador

En la Figura 3.2 se presenta el código para realizar la comprobación del método de registro del usuario Administrador. El método *assetThat()* permite evaluar si se cumple una afirmación expuesta entre dos valores. Las afirmaciones son expresiones que deben evaluarse y dar como resultado un valor de verdadero para que pase la prueba.

```
public void testRegFirebase() {
    administrador.setIdUsuario("-asd35udoboa08760pl");
    administrador.setNombre("Marcia");
    administrador.setApellido("Quishpe");
    administrador.setEmail("marci4qshp12@gmail.com");
    administrador.setPassword("f5488346tng80qhm");

    assertThat(administrador.getIdUsuario(), is(equalTo( operand: "-asd35udoboa08760pl")));
    assertThat(administrador.getNombre(), is(equalTo( operand: "M")));
    assertThat(administrador.getApellido(), is(equalTo( operand: "Q")));
    assertThat(administrador.getEmail(), is(equalTo( operand: "marci4qshp12@gmail.com")));
    assertThat(administrador.getPassword(), is(equalTo( operand: "f5488346tng80qhm")));

    administrador.registrarseEnFirebase(new Activity());
}
```

**Figura 3.2.** Prueba unitaria para el registro del usuario Administrador

### 3.2.2 Pruebas instrumentadas

Las pruebas de instrumentación se encargan de probar o simular comportamientos que los usuarios pueden tener al interactuar con la aplicación. Permiten monitorear toda la interacción que el sistema Android tiene con la aplicación, hace posible que las pruebas invoquen métodos en la aplicación, así como modifiquen y examinen campos en la aplicación, independientemente del ciclo de vida normal de la aplicación.

En la Figura 3.3 se presenta el código para la prueba de instrumentación correspondiente al inicio de la aplicación. La prueba se encarga de simular la siguiente acción: el usuario pulsa el botón *home* e inicia la aplicación Cachuelito. Dentro de la prueba, el método *onView()* permite comprobar si se cumple una afirmación. En este caso, se evalúa que los componentes de la interfaz gráfica sean desplegados de manera correcta.

```
@Test
public void activityLaunch() {
    Log.d( tag: "TAG",  msg: "activityLaunch" + " : " + appContext.getPackageName());

    onView(withId(R.id.textView1)).check(matches(isDisplayed()));
    onView(withId(R.id.imageViewLogo)).check(matches(isDisplayed()));
    onView(withId(R.id.textView2)).check(matches(isDisplayed()));
    onView(withId(R.id.textViewVersionCode)).check(matches(isDisplayed()));
    Log.d(ActivityMainTest.class.getSimpleName(),  msg: "Test sobre interfaz gráfica exitoso");

}
```

**Figura 3.3.** Prueba instrumentada de inicio de la aplicación

En la Figura 3.4 se observa la prueba de instrumentación que simula el siguiente escenario: El usuario inicia la aplicación, luego de unos segundos observa el primer mensaje de información de la aplicación. A continuación, el usuario navega entre cuatro diferentes vistas, y finalmente presiona el botón continuar para iniciar la pantalla de Login.

```java
@Test
public void activityLaunch() {
    Intent intent = new Intent(Intent.ACTION_MAIN);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setClassName(getInstrumentation().getTargetContext(), InformacionInicialActivity.class.getName());
    getInstrumentation().startActivitySync(intent);

    onView(withId(R.id.btnChangeSlide)).perform(click());
    onView(withId(R.id.btnChangeSlide)).perform(click());
    onView(withId(R.id.btnChangeSlide)).perform(click());

    onView(withId(R.id.btnChangeSlide1)).perform(click());
    onView(withId(R.id.btnChangeSlide1)).perform(click());
    onView(withId(R.id.btnChangeSlide1)).perform(click());

    onView(withId(R.id.btnChangeSlide)).perform(click());
    onView(withId(R.id.btnChangeSlide)).perform(click());
    onView(withId(R.id.btnChangeSlide)).perform(click());

    onView(withId(R.id.btnLogin)).perform(click());
}
```

**Figura 3.4.** Prueba instrumentada para navegación de información inicial.

### 3.2.3 Pruebas de funcionamiento

Estás pruebas se encargan de probar características específicas del prototipo de aplicación móvil, permiten encontrar errores que tal vez no fueron detectados durante el desarrollo, y, además, ayudan a garantizar que el prototipo de aplicación móvil funcione correctamente antes de su lanzamiento para el público en general. Para las pruebas de funcionamiento se utilizan los siguientes dispositivos móviles: Motorola Motog60 perteneciente al desarollador, el Xiaomi Redmi Note 10 Lite utilizado por el Administrador, además del Huawei P20Lite utilizado por un Trabajador. En la Figura 3.5 se observan los dispositivos donde la aplicación fue instalada y probada utilizando la plataforma de Play Store.

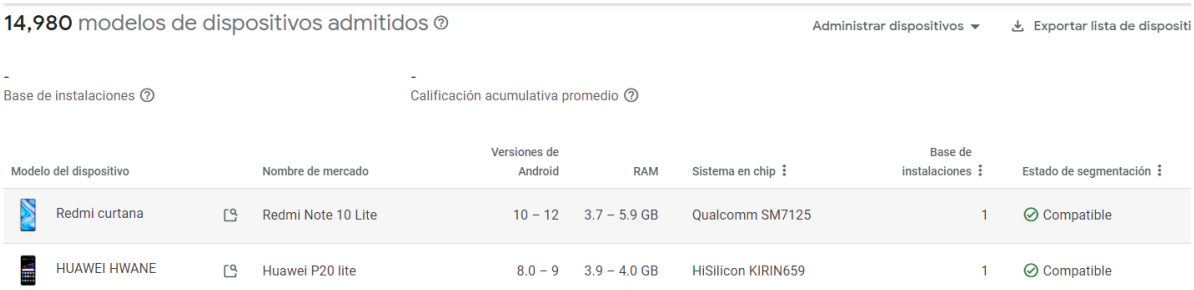



**Figura 3.5.** Catalogo de dispositivos soportados donde la aplicación se ha instalado al menos una vez

#### 3.2.3.1 Prueba de funcionamiento: Registro de Trabajador

Las pruebas de funcionamiento del módulo de registro se encargan de comprobar que se puedan registrar Trabajadores, Empleadores y Oficios. El proceso de registro para un Trabajador se encuentra detallado desde la Figura 3.6 a la Figura 3.10.

Para empezar el proceso de registro de un trabajador, el usuario logeado como Administrador debe dar click en el ícono que se encuentra ubicado en la parte superior izquierda de la aplicación, como se visualiza en la Figura 3.6 (a). Luego, debe seleccionar el submenú Trabajadores como se muestra en la Figura 3.6 (b) y, al presionar click en este submenú, se despliega la pantalla de registro y visualización de Trabajadores de Figura 3.6 (c).

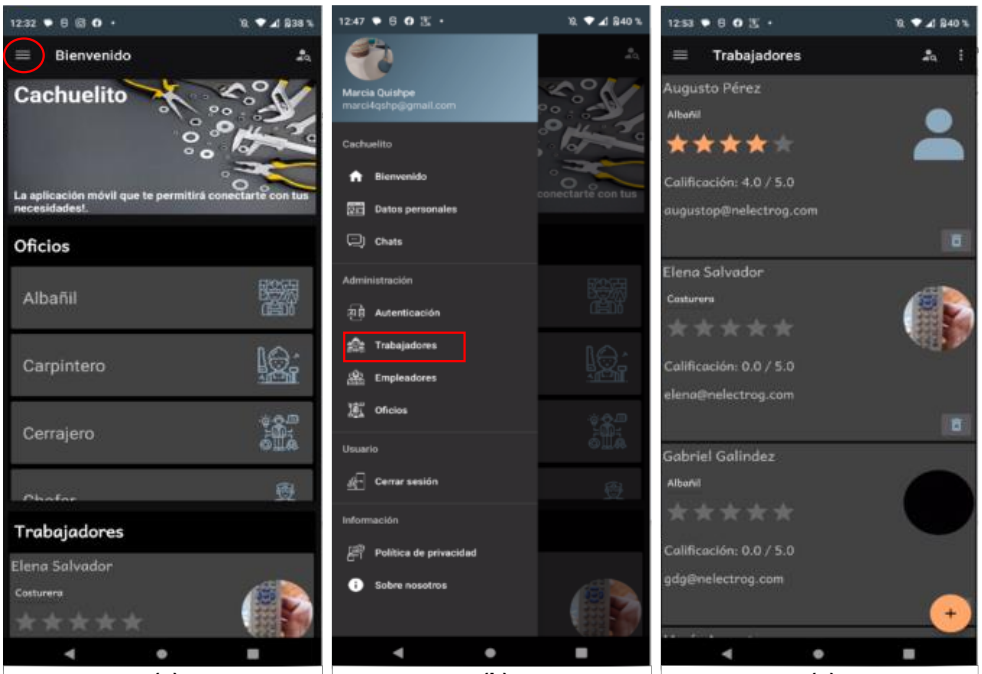



**Figura 3.6.** Interfaz gráfica principal (a), menú principal (b), registro y visualización de Trabajadores(c).

Para el registro de datos del trabajador, el Administrador presiona el botón “+”, como se muestra en la Figura 3.6 (c). Al presionar este botón, se despliega la pantalla donde se procede a ingresar los datos personales de un Trabajador (Figura 3.7 (a)). Además, el Administrador deber llenar los campos de nombre y apellido del nuevo trabajador de manera obligatoria. Para cargar una foto de perfil, el Administrador debe presionar el ícono de una cámara, y la aplicación dará la opción de escoger entre cargar una foto realizada con la cámara del dispositivo móvil o una foto desde la galería de imágenes, como se presenta en el cuadro de diálogo de la Figura 3.7 (b). Además, en la Figura 3.7 (c), se pueden visualizar los mensajes informativos que el usuario recibe en caso de no proporcionar un nombre y apellido al crear una nueva cuenta.

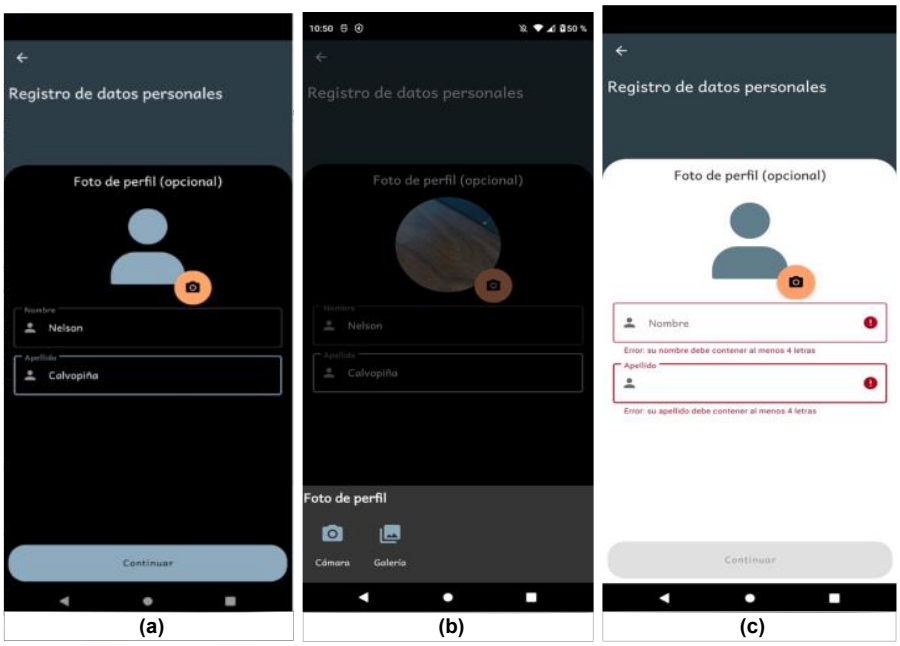



**Figura 3.7.** Registro de datos personales (a) y selección de foto de perfil (b).

Luego de presionar el botón “Continuar” se muestra la pantalla de registro de record policial como se visualiza en la Figura 3.8 (a). El Administrador ingresa el documento utilizando una foto capturada con la cámara (Figura 3.8 (b)). Finalmente, el mensaje de procesamiento de datos se observa en la Figura 3.8 (c).

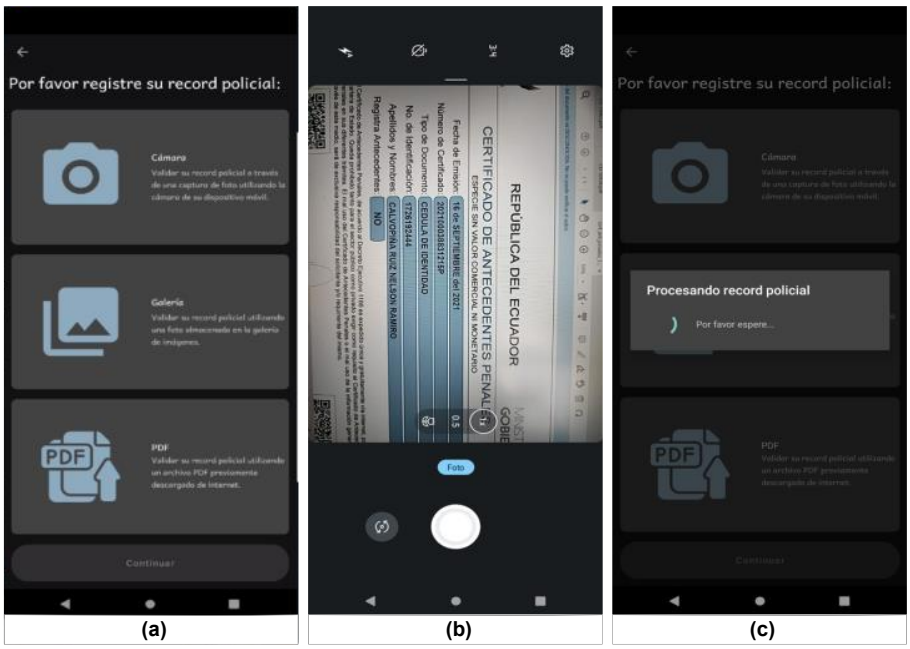



**Figura 3.8.** Registro de record policial (a), captura de record policial por medio de la cámara (b) y procesamiento del record policial (c).

Luego de que la aplicación móvil se encarga de procesar los datos del registro policial, el Administrador presiona "Continuar" para acceder a la pantalla de selección de oficios, tal

como se visualiza en la Figura 3.9 (a). Es importante recalcar que en esta etapa, el Administrador debe seleccionar al menos un oficio, como se muestra en la Figura 3.9 (b). En caso de no seleccionar ningún oficio, el sistema mostrará al usuario el mensaje de la Figura 3.9 (c), que le indicará la necesidad de elegir al menos un oficio para poder seguir con el proceso de registro.

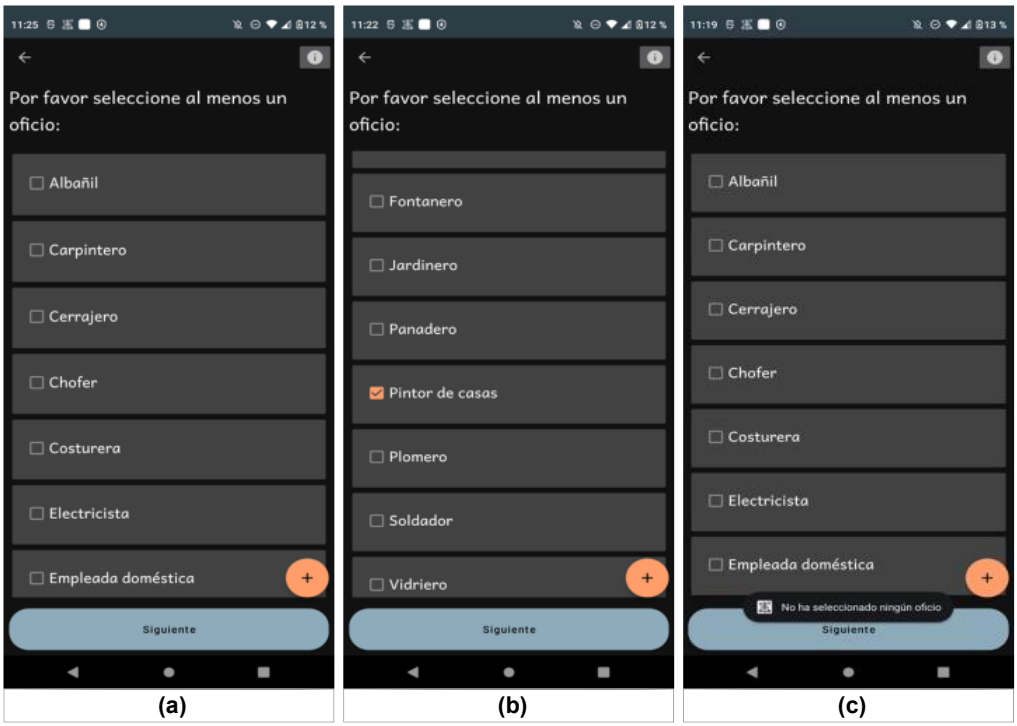
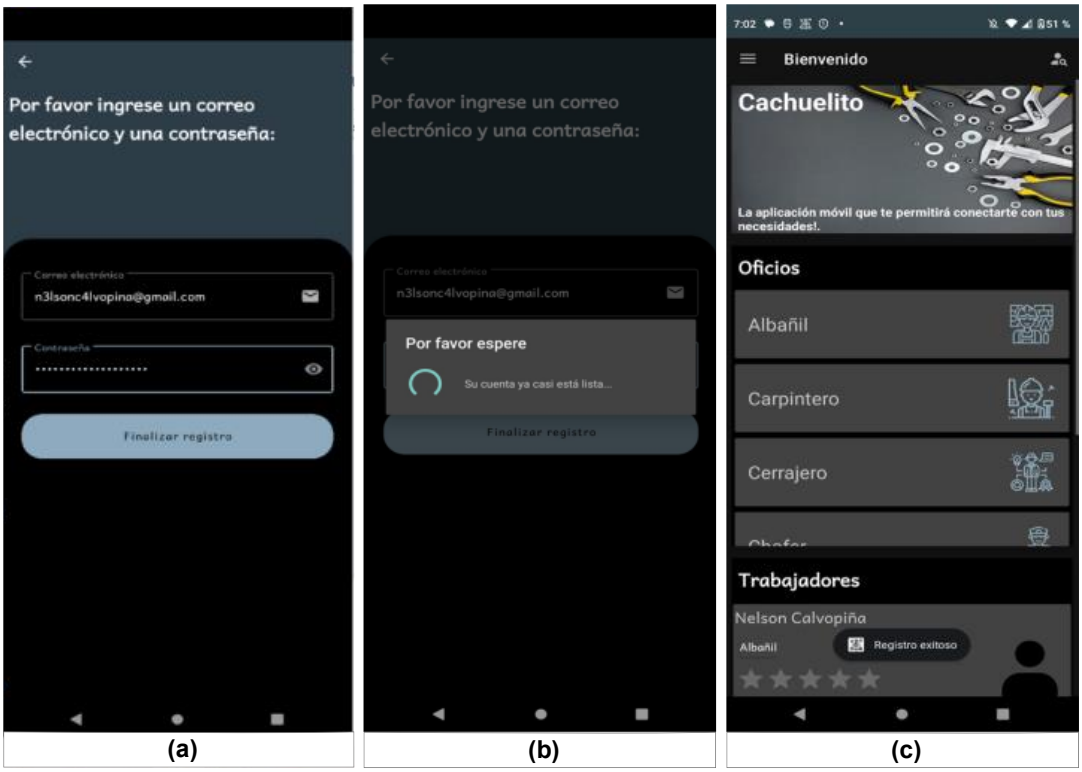



**Figura 3.9.** Visualización de oficios (a), selección de oficio (b) y mensaje de error (c)

A continuación, el Administrador debe presionar el botón “Siguiente” que se encuentra en la parte inferior de la pantalla de visualización de oficios (Figura 3.9 (b)). Posteriormente, se desplegará la pantalla de registro de correo electrónico y contraseña, como se presenta en la Figura 3.10 (a). El proceso de registro finaliza luego de que el Administrador llena los campos de correo electrónico y contraseña; a continuación, presiona el botón “Finalizar registro”.

Además, durante este proceso, la aplicación se muestra el mensaje de la Figura 3.10 (b), indicando que el proceso de registro casi se encuentra listo. Si el registro fue exitoso, la aplicación presenta un mensaje de “Registro exitoso” (Figura 3.10 (c)).




(a) (b) (c)

**Figura 3.10.** Registro de correo electrónico y contraseña (a), procesamiento de información de registro (b), mensaje de registro existoso (c)

### 3.2.3.2 Prueba de funcionamiento: Registro de Empleador

Para registrar un Empleador, el Administrador debe comenzar haciendo clic en el submenú Empleadores, como se ilustra en la Figura 3.6 (b). Luego, se abre la pantalla de registro y visualización de Empleadores, que se muestra en la Figura 3.11 (a). Para acceder a la pantalla de registro de datos del Empleador, el Administrador debe presionar el botón "+", como se representa en la Figura 3.11 (a). El Administrador debe llenar los campos con los datos del nuevo Empleador y, posteriormente, hacer clic en el botón "Continuar" para avanzar a la pantalla de registro de correo electrónico y contraseña, como se muestra en la Figura 3.11 (b).

Para finalizar el registro, el Administrador debe llenar los campos de correo electrónico y contraseña y, a continuación, debe presionar el botón “Finalizar registro” Figura 3.11 (c). Es importante destacar que los mensajes mostrados en el proceso de registro del Empleador son idénticos a los que se presentan en el proceso de registro del Trabajador y se pueden ver en la Figura 3.10 (b) y Figura 3.10 (c).

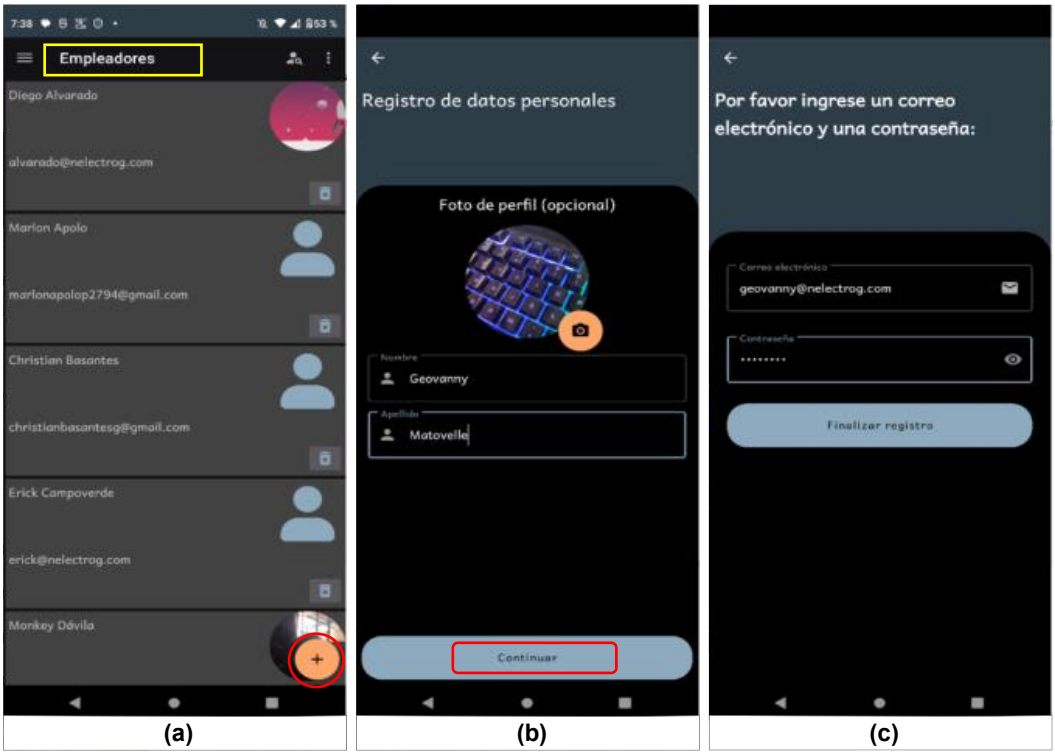


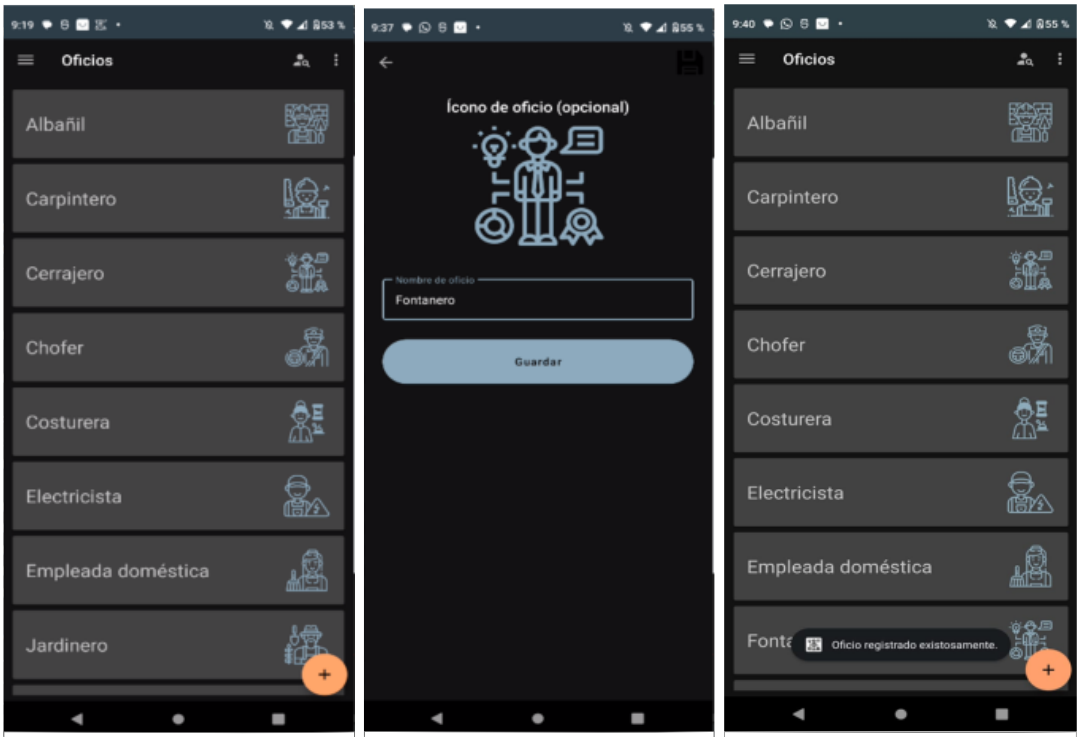

**Figura 3.11.** Visualización y registro de Empleadores (a), registro de datos personales (b), registro de correo electrónico y contraseña (c)

#### 3.2.3.3 Prueba de funcionamiento: Registro de Oficio

Para el registro de un nuevo oficio, el Administrador ingresa al submenú Oficios (Figura 3.6 (b)). A continuación, se despliega la pantalla de registro y visualización de Oficios como se observa en la Figura 3.12 (a).

Para registrar un nuevo oficio, el Administrador presiona el botón “+”, luego de lo cual aparece la pantalla de la Figura 3.12 (b) donde se ingresa el nombre del nuevo oficio. Además, se debe cargar una imagen para identificar a dicho oficio. Si no se carga una imagen para identificar al oficio, la aplicación asigna al nuevo oficio el ícono mostrado en la Figura 3.12 (b).

Por otra parte, si el oficio fue registrado de manera exitosa, la aplicación presenta el mensaje de la Figura 3.12 (c) de “Registro exitoso”.



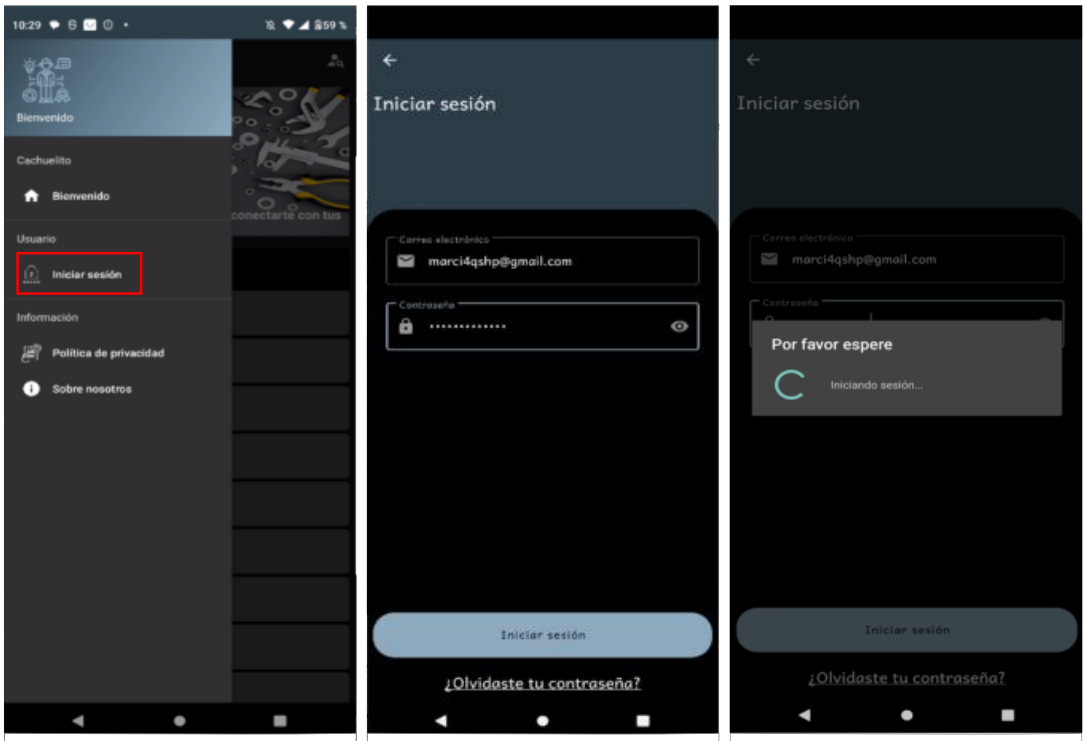

(a) (b) (c)

**Figura 3.12.** Visualización y registro de oficios (a), nuevo oficio (b), mensaje de registro exitoso (c)

### 3.2.3.4 Pruebas de funcionamiento: Módulo de inicio de sesión

Las pruebas de funcionamiento del módulo de inicio de sesión permiten comprobar que los usuarios Trabajadores, Empleadores y Administrador, puedan acceder de manera correcta a la aplicación utilizando las respectivas credenciales de correo electrónico y contraseña.

Para iniciar sesión, el usuario Administrador se ubica en el submenú inicio de sesión como se muestra en la Figura 3.13 (a). A continuación, aparece la pantalla de inicio de sesión, donde se procede a ingresar un correo electrónico y una contraseña como se presenta en la Figura 3.13 (b). Finalmente, si las credenciales son correctas se procede a mostrar un mensaje de información al usuario cómo el que se visualiza en la Figura 3.13 (c).




(a) (b) (c)

**Figura 3.13.** Menú principal usuario no logeado (a), pantalla de inicio de sesión (b) y mensaje de información al usuario (c)

Las interfaces gráficas que permiten el inicio de sesión de los usuarios Trabajadores y Empleadores son las mismas que se han utilizado para el usuario Administrador.

### 3.2.3.5 Pruebas de funcionamiento: Módulo de búsqueda

En esta sección se realizan las pruebas de funcionamiento del módulo de búsqueda utilizadas por el usuario Empleador.

En la Figura 3.14 (a) se observa la búsqueda del oficio Albañil y se muestra una vista con los Trabajadores que se encuentran registrados en dicho oficio. Además, los resultados de búsqueda son presentados en la Figura 3.14 (b), donde se observa un poco más de información acerca de los Trabajadores que coinciden con el oficio buscado. De igual manera, si no se encuentran Trabajadores que coincidan con el oficio buscado, el usuario Empleador recibe mensajes de información que le indican que no se encontraron resultados (Figura 3.14 (c)).

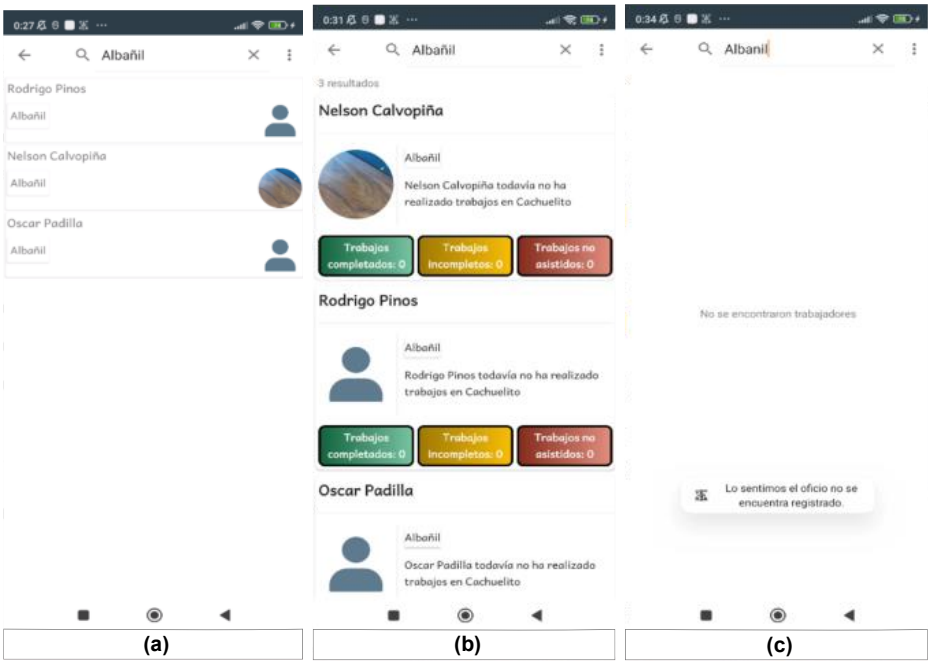



**Figura 3.14.** Búsqueda de Trabajadores (a), resultados de búsqueda (b) y mensajes de información al usuario (c).

### 3.2.3.6 Pruebas de funcionamiento: Módulo de notificaciones

A continuación, se describen las pruebas de funcionamiento para el módulo de notificaciones.

La aplicación cuenta con notificaciones de: mensajes, llamadas de voz, videollamadas y citas de trabajo.

En la Figura 3.15 (a) se muestra la notificación de un nuevo mensaje de texto, que permite alertar al usuario que nuevos mensajes se encuentran en la bandeja de entrada y contiene un botón de respuesta rápida para contestar el mensaje como se observa en la Figura 3.15 (b). Además, luego de realizar una respuesta rápida, el chat o la conversación de donde proviene el mensaje, se actualiza como se muestra en la Figura 3.15 (c).

El usuario puede responder notificaciones de llamadas de voz o videollamadas como la presentada en la Figura 3.16 (a). Si el usuario presiona el botón de contestar, se despliega la pantalla de la Figura 3.16 (b), donde se encuentran botones para controlar las acciones durante la videollamada.

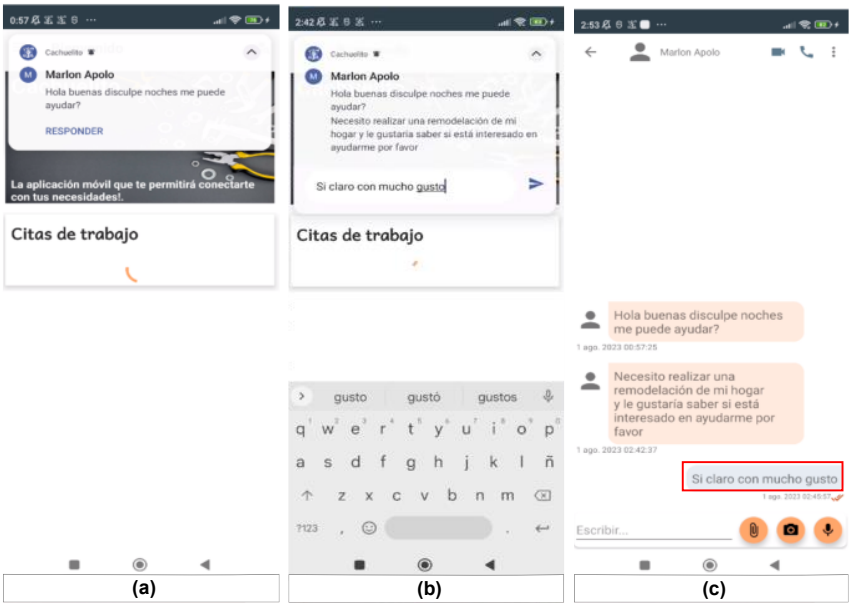



**Figura 3.15.** Notificación de mensaje de texto (a) y respues ràpida a un mensaje de texto (b), actualización de chat utilizando respuesta rápida (b).

Caso contrario, si el usuario presiona el botón rechazar, se muestra el mensaje de la Figura 3.16 (c), donde se indica que la llamada fue rechazada. Además, se debe recalcar que las notificaciones de videollamadas y llamadas de voz tienen el mismo tratamiento.

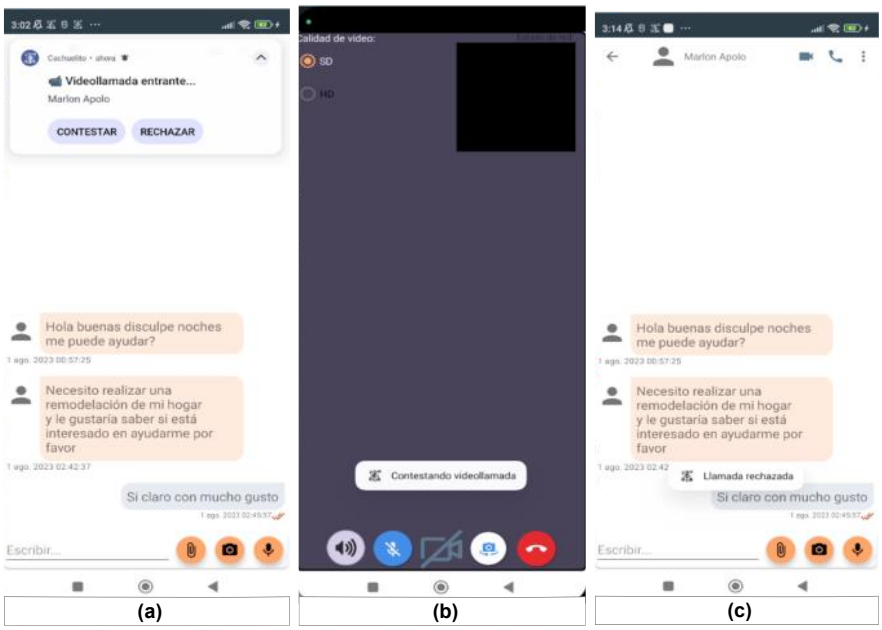



**Figura 3.16.** Notificación de videollamada (a), contestando videollamada (b) y rechazando videollamada (c).

En la Figura 3.17 (a) se presenta la notificación de una llamada de voz entrante. Si el usuario presiona el botón contestar, se despliega la pantalla mostrada en la Figura 3.17 (b). Si el usuario presiona el botón rechazar, se presenta el mensaje de la Figura 3.17 (c).

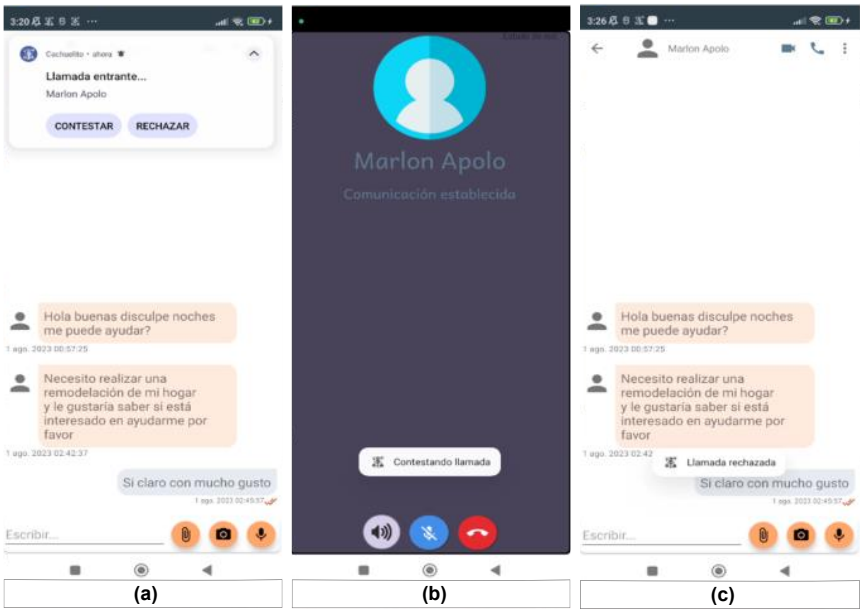



**Figura 3.17.** Notificación de videollamada (a), contestando videollamada (b) y rechazando videollamada (c).

En la Figura 3.18 (a) se presenta la notificación de una nueva cita de trabajo. El usuario puede presionar la notificación para visualizar el detalle de la misma (Figura 3.18 (b)). Además, la fecha seleccionada para cualquier cita de trabajo debe ser mayor a la fecha y hora actual, caso contrario se presenta un mensaje de error (Figura 3.18 (c)).

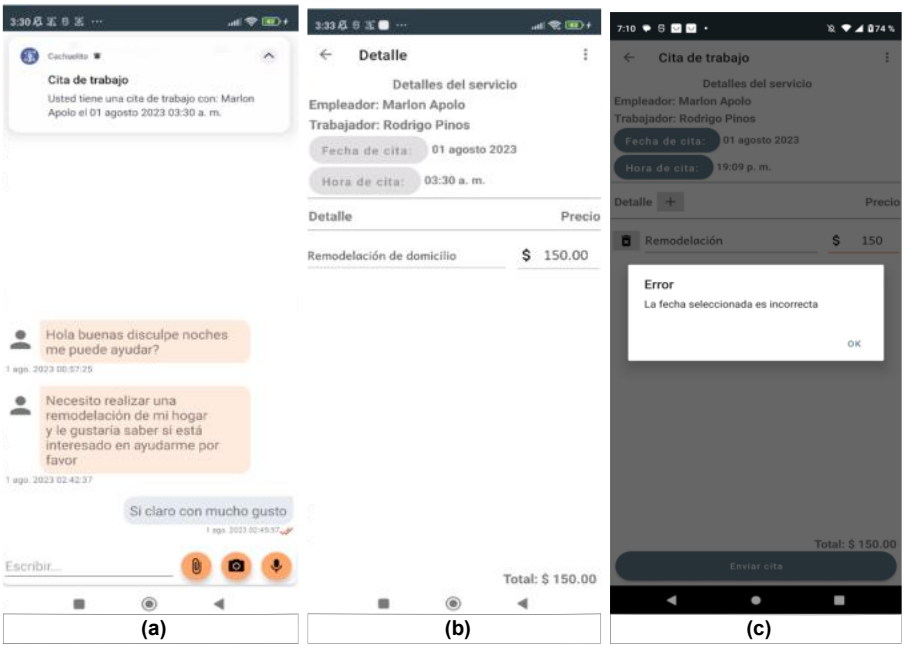



**Figura 3.18.** Notifiación de cita de trabajo (a), detalle de una cita de trabajo (b) y mensaje de información al seleccionar una fecha incorrecta (c)

### 3.2.3.7 Pruebas de funcionamiento: Módulo de mensajería instantánea

Las pruebas de funcionamiento para este módulo se las realiza en base a una conversación entre un Trabajador y un Empleador.

En la Figura 3.19 (a) se presenta el chat del usuario Trabajador con los diferentes mensajes intercambiados hasta el momento. Además, se realiza la prueba de grabación de audio y envío del mismo por parte del usuario Empleador, lo que se puede visualizar en la Figura 3.19 (b). Si el audio es cargado de manera correcta, esto se ve reflejado en la conversación de los participantes como se muestra en la Figura 3.19 (a) y (b).

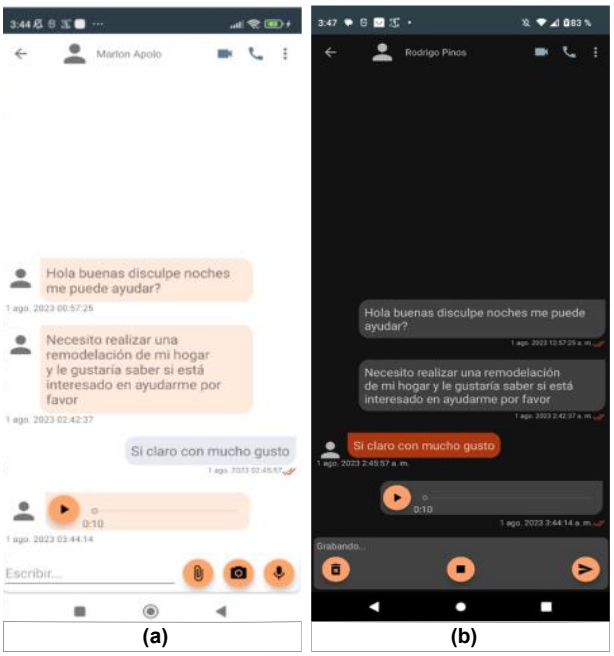



**Figura 3.19.** Chat del usuario trabajador (a) y chat del usuario empleador (b)

### 3.2.3.8 Pruebas de funcionamiento: Módulo de perfil

El funcionamiento del módulo de perfil es evaluado realizando actualizaciones a los datos personales del usuario Trabajador.

En la Figura 3.20 (a) se muestra la información personal del usuario Trabajador llamado Rodrigo Pinos. En caso de realizar una actualización, el usuario recibe un mensaje de información que indica que se está llevando a cabo dicho proceso como se visualiza en la Figura 3.20 (b).

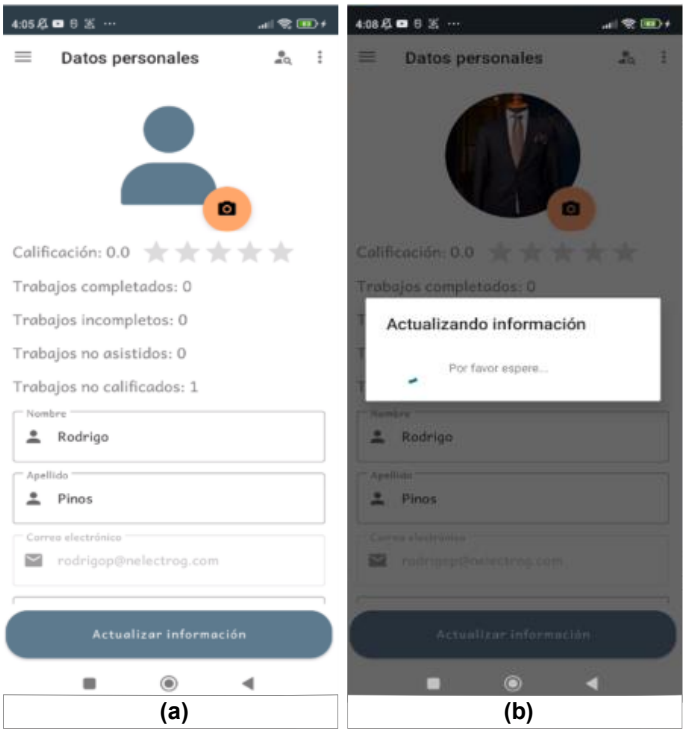



**Figura 3.20.** Datos personales (a), proceso de actualización de datos personales(b)

### 3.2.4 Pruebas de validación

Las pruebas de validación fueron realizadas a los 11 usuarios que participaron en el apartado de las entrevistas; en este caso se aplicaron 3 modelos de encuestas.

El modelo de encuesta aplicada al Administrador y sus resultados se encuentra en la Tabla 3.2, y los otros 2 modelos de encuestas se encuentran detallados en el ANEXO VII.

### 3.2.5 Resultados

En esta sección se presentan los resultados de las encuestas realizadas a 5 Empleadores, 5 Trabajadores y al usuario Administrador. Los resultados de las encuestas permiten evidenciar que se cumplan los requerimientos funcionales y no funcionales que han sido previamente establecidos.

El resultado de la encuesta aplicada al Administrador de la aplicación se muestra en la Tabla 3.2. Además, los resultados de las pruebas de validación realizadas a los Trabajadores y empleadores se localizan en el ANEXO VII.

**Tabla 3.2.** Resultados de encuesta realizada al Administrador

| **Pregunta** | **Si (%)** | **No (%)** |
|---|---|---|
| 1. ¿Cómo usuario Administrador por favor acceda a la aplicación con la contraseña=$6gsas34tdf#v y el correo=marci4qshp@gmail.com, es posible? | 100 | 0 |
| 2. ¿Ingrese a la opción Datos personales y revise su información, es ésta correcta? | 100 | 0 |
| 3. ¿Cuándo accedió a editar su perfil, su nueva información se guardó de manera correcta? | 100 | 0 |
| 4. ¿La aplicación permite hacer un cambio en la contraseña? | 0 | 100 |
| 5. ¿La aplicación permite crear un nuevo usuario con el rol de Trabajador? | 100 | 0 |
| 6. ¿La aplicación permite crear un nuevo usuario con el rol de Empleador? | 100 | 0 |
| 7. ¿La aplicación permite ver todos los usuarios registrados, Trabajadores y Empleadores? | 100 | 0 |
| 8. ¿La aplicación permite registrar nuevos oficios? | 100 | 0 |
| 9. ¿La aplicación permite realizar el envío de mensajes de texto? | 100 | 0 |
| 10. ¿La aplicación permite realizar llamadas de voz? | 100 | 0 |
| 11. ¿La aplicación permite realizar videollamadas? | 100 | 0 |
| 12. ¿La aplicación permite eliminar su cuenta personal? | 100 | 0 |

Los resultados de las pruebas de validación permitieron evidenciar 3 fallas que no fueron detectadas en la aplicación móvil durante su proceso de desarrollo. Las fallas están relacionadas con la calidad de las llamadas de voz y videollamadas, la actualización de datos personales y los recordatorios de citas de trabajo.

A continuación, se presenta un detalle de los inconvenientes que tuvo la aplicación móvil.

- Las llamadas de voz y videollamadas pueden presentar deterioro en la calidad cuando se utiliza la red celular y no la red WiFi, evitando en algunos casos incluso el establecimiento de la comunicación con otro usuario.

- Si un usuario intenta actualizar su información personal y, en el proceso, agrega una nueva foto de perfil, pero se queda sin saldo en su línea de telefonía móvil mientras ocurre toda esta actualización, el proceso se detendrá automáticamente debido a la insuficiencia de saldo. Para solucionar este problema, se presenta un mensaje al usuario informándole que sus datos móviles serán utilizados en el proceso de actualización y, de continuar, la actualización podría no reflejar los cambios que ha realizado en su información personal.

- Se ha notado que las notificaciones que sirven como recordatorios para las citas de trabajo, generadas por los Trabajadores, pueden experimentar retrasos. Este fenómeno se vuelve particularmente significativo en dispositivos que operan con Android 8.0 o versiones más recientes, ya que el tratamiento de las notificaciones cambia drásticamente para cada versión.

- En el caso del dispositivo móvil Motorola MotoG60 con Android 12, las notificaciones de recordatorio pueden llegar con un retraso de entre 1 y 5 minutos con respecto a la hora programada para la cita de trabajo. Este error se encuentra muy relacionado con la API de Notificaciones que se utiliza en el desarrollo de prototipo de aplicación móvil. Para evitar inconvenientes, se incorporó redundancia, esto quiere decir que la notificación de cita de trabajo alerta al trabajador y empleador la primera vez que se genera la cita de trabajo y, una segunda vez notifica a los mismos usuarios a la hora que se encuentra registrada en la cita de trabajo.

## 3.3 Conclusiones

- La aplicación necesita al menos que el dispositivo móvil del cliente cuente con una versión de Android 7.0 para su funcionamiento. Esto debido a que el SDK de *Agora* demanda una versión mínima de Android 7.0 para utilizar los servicios de comunicaciones como llamadas de voz y videollamadas [17]. Además, *Firebase*, necesita una versión de Android 4.4 o superior para operar. Por lo tanto, no se presentan conflictos entre el SDK de *Agora* y *Firebase* en términos de requerimientos de versión de Android.

- *Firebase Authentication* únicamente permite eliminar la información personal de un usuario cuando este mantiene una sesión activa en la aplicación. En este contexto, el Administrador no tendría la capacidad de eliminar las cuentas de los usuarios registrados. Por lo tanto, se implementó una REST API para otorgar al Administrador el control total de la gestión de usuarios.

- El uso del SDK Agora para el desarrollo del módulo de llamadas y videollamadas demostró ser una integración perfecta en el proyecto. Este SDK fue de gran utilidad al evitar la necesidad de implementar un servidor dedicado para reenviar audio y video, lo que resultó en una significativa reducción de líneas de código y evitó el mantenimiento adicional que habría conllevado dicho servidor.

- El Kit de Machine Learning proporcionado por Firebase puede experimentar dificultades en el reconocimiento de texto. Estos problemas pueden surgir debido a varios factores, incluyendo la calidad de la imagen capturada, el tamaño y formato de la imagen (ImageFormat.YUV_420_888 o ImageFormat.NV21) y el tipo de API utilizado para la captura de fotos (CameraX o Camera). En tales situaciones, la solución más apropiada es tomar una nueva fotografía con una mayor calidad, permitiendo así que el ML Kit pueda reconocer el texto de manera precisa. Es importante destacar que si la imagen contiene múltiples bloques de texto, se recomienda capturarla con una resolución superior a 640 x 480 píxeles.

## 3.4 Recomendaciones

- Se recomienda utilizar el SDK Agora para aplicaciones que necesiten incorporar servicios de comunicaciones en tiempo real. Esto debido a que Agora posee un Cloud Proxy que actúa como intermediario entre internet y el cliente, manteniendo la dirección IP o la ubicación real del dispositivo móvil fuera del alcance de los hackers, que intentan controlar su actividad en la red, robar datos confidenciales, o acceder de manera no autorizada.

- Para asegurar un óptimo procesamiento con *Machine Learning* al registrar el récord policial, se recomienda realizar la captura de imágenes a una distancia aproximada de 50 cm. Mantener esta distancia permitirá obtener imágenes más nítidas y detalladas, lo que mejorará la precisión y calidad del procesamiento de datos.

- Para aplicaciones de chat, noticias o blogs, se recomienda utilizar una base de datos en tiempo real como *Firebase Realtime Database*. Esta elección proporciona actualizaciones de datos de manera inmediata, con un tiempo de respuesta menor a 200 ms. De esta manera, se logra mantener una experiencia de usuario actualizada y altamente satisfactoria.

- A la hora de diseñar los *sketches* de las diferentes interfaces gráficas, se recomienda el uso del software Adobe XD. Este software ofrece la posibilidad de crear prototipos de interfaces gráficas adaptados a dispositivos móviles o páginas web, y al utilizarlo, se podrá agilizar y optimizar el proceso de diseño, lo que permitirá desarrollar interfaces que brinden una experiencia de usuario más satisfactoria.

- Para garantizar una gestión eficaz de usuarios y datos en la aplicación, se recomienda utilizar *Firebase Authentication*, ya que facilita la validación de correos

electrónicos y contraseñas; además de proporcionar un ID único para identificar a cada usuario de manera segura dentro de la aplicación.

# 4 REFERENCIAS BIBLIOGRÁFICAS


[1] INEC, “Boletín Técnico N° 11-2023-ENEMDU”, 24 Julio 2023. [En línea]. Disponible en: https://www.ecuadorencifras.gob.ec/documentos/web-inec/EMPLEO/2023/Junio/202306_Boletin_empleo_ENEMDU.pdf. [Último acceso: 19 de Agosto de 2023].

[2] INEC, “Boletín Técnico N° 05-2023-ENEMDU”, 28 Febrero 2023. https://www.ecuadorencifras.gob.ec/documentos/web-inec/EMPLEO/2022/Anual/Bolet%C3%ADn%20t%C3%A9cnico%20anual%20enero-diciembre%202022.pdf. [Último acceso: 22 de Mayo de 2023].

[3] ALQUIMIASOFT SA, “Jobra”, 2020. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.ec.alquimiasoft.jobra&hl=es_EC&gl=US. [Último acceso: 22 de mayo de 2023].

[4] “¿Qué es Aora?”. [En línea]. Disponible en: https://aoraservicios.com/se-un-pas/. [Último acceso: 22 de Mayo de 2023].

[5] Firebase Developers, “ML Kit for Firebase”, Diciembre 2020. [En línea]. Disponible en: https://firebase.google.com/docs/ml-kit. [Último acceso: 30 de Abril de 2023].

[6] Android Developers, “Guía de arquitectura de apps”, Febrero 2021. [En línea]. Disponible en: https://developer.android.com/jetpack/guide?hl=es-419. [Último acceso: 29 de Mayo de 2023].

[7] Google Developers, “1.0: Introduction to Android”, Septiembre 2018. [En línea]. Disponible en: https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-1-build-your-first-app/1-0-c-introduction-to-android/1-0-c-introduction-to-android.html. [Último acceso: 30 de Abril de 2023].

[8] A. Kumar, *Mastering Firebase for Android Development*, 1st. ed. Birmingham: Packt Publishing Ltd., 2018.

[9] Firebase Developers, “Estructura tu base de datos”, 2023. [En línea]. Disponible en: https://firebase.google.com/docs/database/android/structure-data?hl=es-419. [Último acceso: 22 de Mayo de 2023].

[10] K. Sierra y B. Bates, *Head First Java™*, 2nd. ed. O´Reilly Media, 2009.

[11] J. Boyarsky y S. Selikoff, *Oracle® Certified Professional Java® SE 11 Developer*. Indianapolis: John Wiley & Sons, 2021.

[12] L. Quin, “XML ESSENTIALS”, 2015. [En línea]. Disponible en: https://www.w3.org/standards/xml/core.html. [Último acceso: 29 de Mayo de 2023].

[13] T. Bray, J. Paoli, E. Maler, C. M. Sperberg-McQueen, y F. Yergeau, “Extensible Markup Language (XML) 1.0 ”, 2008. https://www.w3.org/TR/xml/. [Último acceso: 29 de Mayo de 2023].

[14] E. T. Ray, *Learning XML: Creating Self-Describing Data*, 2nd. ed. Sebastopol: O´Reilly Media, 2003.

[15] N. Middleton y R. Schneeman, *Heroku: Up and Running*, 1st. ed. Sebastopol: O’Reilly Media, 2013.

[16] Gradle Community, “Getting Started with Gradle on Heroku”, 15 Julio 2022. [En línea]. Disponible en: https://devcenter.heroku.com/articles/getting-started-with-gradle-on-heroku#set-up22. [Último acceso: 22 de Mayo de 2023].

[17] Agora Developers, “Agora:Core Concepts”, 2023. [En línea]. Disponible en: https://docs.agora.io/en/video-calling/overview/core-concepts?platform=android. [Último acceso: 20 de Mayo de 2023].

[18] K. Beck y A. Cynthia, “Extreme Programming Explained: Embrace Change”, 2nd. ed. Massachusetts: Pearson Education, 2004.

[19] Lucid Content Team, “What Is Extreme Programming? An Overview of XP Rules and Values”. [En línea]. Disponible en: https://www.lucidchart.com/blog/what-is-extreme-programming. [Último acceso: 2 de Mayo de 2023].

[20] Android Developers, “Descripción general de notificaciones”,17 Agosto 2023,[En línea]. Disponible en: https://developer.android.com/guide/topics/ui/notifiers/notifications?hl=es-419. [Último acceso: 20 de Agosto de 2023].

[21] L. Moroney, “The Definitive Guide to Firebase: Build Android Apps on Google’s Mobile Platform”, 2017.