StudyStay: Manual del Programador

José María Pozo Hidalgo

Índice

1. Introducción

La idea que fundamenta **StudyStay** surge al pensar en una temática sobre la que realizar el proyecto de fin de estudios que aborde un problema o necesidad real, en este caso, una necesidad propia, como la de encontrar alojamiento en otra ciudad en la que realizaré las prácticas de empresa, hecho que en ocasiones se complica por no tener un sitio especializado en **alojamientos para estudiantes.** En muchas ocasiones, buscando alojamiento en otras plataformas nos encontramos con alojamientos que no permiten estudiantes, alquileres de larga duración o incluso anuncios de venta de viviendas que no consiguen ser filtradas correctamente.

StudyStay es una plataforma interactiva diseñada para conectar a estudiantes con opciones de alojamiento cercanas a sus instituciones educativas. La aplicación no solo facilita la búsqueda y el alquiler de habitaciones y apartamentos, sino que también proporciona un foro comunitario y un chat para que los estudiantes interactúen, busquen compañeros de piso y compartan experiencias. Su enfoque se centra en la seguridad, accesibilidad y apoyo a la comunidad estudiantil.

StudyStay Desktop

La aplicación de escritorio cuenta con funcionalidades completas de CRUD para todas las tablas del sistema, permitiendo la gestión integral de usuarios, alojamientos, reservas, mensajes de chat, y foros. Además, incluye la gestión de copias de seguridad de la base de datos, asegurando que toda la información esté protegida y se pueda restaurar en caso de cualquier incidente.

StudyStay Android

La aplicación de Android está diseñada para proporcionar a los usuarios una experiencia móvil optimizada y completa. Las principales funcionalidades incluyen:

- **Personalización de Perfil:** Los usuarios pueden personalizar sus perfiles con información relevante y preferencias personales.
- **Chat:** Permite la comunicación entre usuarios, facilitando la interacción y el intercambio de información.
- **Foro:** Un espacio comunitario donde los estudiantes pueden crear y participar en discusiones, buscar compañeros de piso, y compartir experiencias.
- **Publicación y Alquiler de Habitaciones:** Los usuarios pueden publicar anuncios de habitaciones disponibles y realizar reservas directamente desde la aplicación.

Para cumplir el objetivo, las herramientas usadas serán las siguientes:

- **Entornos de desarrollo:** IntelliJ IDEA y Android Studio.
- **Lenguajes de programación:** Java para Android/Escritorio y SQL para la base de datos.
- **Base de datos:** MySQL.
- **Interfaces:** JavaFX y Android SDK.
- **Integración de la API de Google Maps** para visualizar ubicaciones.
- **Aplicaciones auxiliares** como JasperSoft Studio o SceneBuilder.

2. Análisis del sistema

Requisitos Funcionales Comunes

- **Autenticación de Usuarios:** Los usuarios deben poder registrarse e iniciar sesión utilizando credenciales seguras.

Requisitos Funcionales de la Aplicación de Escritorio

- **Gestión Completa de CRUD:** La aplicación de escritorio debe permitir la creación, lectura, actualización y eliminación de datos en todas las tablas del sistema, incluyendo usuarios, alojamientos, reservas, mensajes de chat y foros.
- **Gestión de Copias de Seguridad:** El administrador debe poder gestionar copias de seguridad de la base de datos para asegurar la protección de los datos.

Requisitos Funcionales de la Aplicación de Android

- **Personalización de Perfil:** Los usuarios deben poder personalizar sus perfiles con información relevante y preferencias personales.
- **Chat:** Los usuarios deben poder comunicarse entre ellos a través de un sistema de chat.
- **Foro:** Los usuarios deben tener acceso a un foro donde puedan crear y participar en discusiones, buscar compañeros de piso y compartir experiencias.
- **Publicación y Alquiler de Habitaciones:** Los usuarios deben poder publicar anuncios de habitaciones disponibles y realizar reservas directamente desde la aplicación.

Requisitos No Funcionales

- **Seguridad:** Protección de datos personales y transacciones seguras dentro de la plataforma.
- **Usabilidad:** Interfaz intuitiva y fácil de usar, tanto en dispositivos móviles como en escritorio.
- **Rendimiento:** Tiempos de respuesta rápidos para la búsqueda y carga de contenido.
- **Escalabilidad:** Capacidad de soportar un creciente número de usuarios y anuncios sin degradación del rendimiento.
- **Compatibilidad:** Funcionamiento adecuado en las versiones más recientes de los sistemas operativos para dispositivos móviles.
- **Mantenibilidad:** Facilidad de mantenimiento y actualización de la aplicación y la base de datos.

3. Casos de Uso

Este diagrama ilustra las interacciones entre los diferentes actores y los casos de uso dentro del sistema StudyStay, proporcionando una visión clara de las funcionalidades disponibles tanto para administradores como para usuarios.

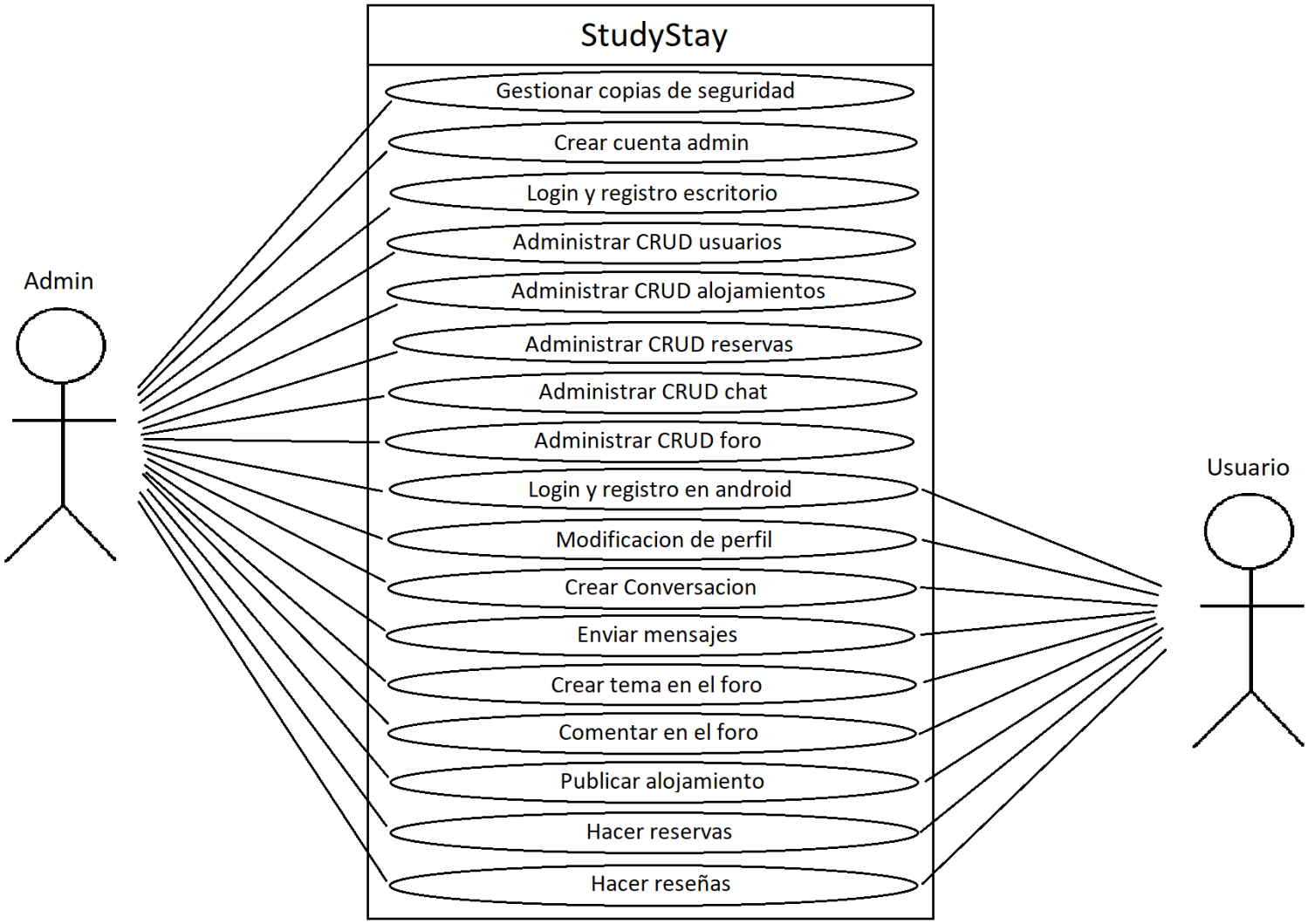
Actor Administrador

- Gestión Completa de CRUD: Permite la creación, lectura, actualización y eliminación de datos en todas las tablas del sistema, incluyendo usuarios, alojamientos, reservas, mensajes de chat y foros.
- Gestión de Copias de Seguridad: Permite al administrador realizar y gestionar copias de seguridad de la base de datos.
- Moderación de Foro: Permite supervisar y moderar el foro de la comunidad, incluyendo la eliminación de publicaciones inapropiadas.
- Resolución de Conflictos: Permite resolver conflictos entre usuarios y atender consultas.
- Gestión de Ofertas por Propietarios: Permite gestionar las ofertas de alojamiento publicadas por los propietarios.
- Generación de Informes: Permite generar informes sobre las actividades de la plataforma, como número de usuarios registrados, número de reservas realizadas, etc.
- Gestión de Usuarios: Permite gestionar los roles y permisos de los usuarios, así como desactivar o eliminar cuentas cuando sea necesario.
- Monitoreo de Actividades: Permite monitorear las actividades de los usuarios para detectar y prevenir comportamientos inapropiados o fraudulentos.

Actor Usuario (Cliente)

- Personalización de Perfil: Permite personalizar su perfil con información relevante y preferencias personales.
- Chat: Permite comunicarse con otros usuarios a través de un sistema de chat.
- Foro: Permite acceder a un foro donde se pueden crear y participar en discusiones, buscar compañeros de piso y compartir experiencias.
- Búsqueda de Alojamiento: Permite buscar alojamientos filtrando por ubicación, precio y otras preferencias.
- Publicación y Alquiler de Habitaciones: Permite publicar anuncios de habitaciones disponibles y realizar reservas directamente desde la aplicación.
- Notificaciones: Recibe notificaciones sobre nuevos mensajes, respuestas en el foro, y actualizaciones sobre sus reservas.

- Mapas y Localización: Visualiza la ubicación de los alojamientos y calcula distancias a las instituciones educativas.
- Historial de Actividades: Permite ver un historial de sus actividades en la aplicación, incluyendo mensajes enviados, publicaciones en el foro, y reservas realizadas.
- Sistema de Favoritos: Permite marcar alojamientos como favoritos para acceder a ellos rápidamente en el futuro.



4. Arquitectura del Sistema

La arquitectura del sistema StudyStay se ha diseñado para ser escalable, segura y fácil de mantener, permitiendo una gestión eficiente de los datos y una experiencia de usuario fluida tanto en la aplicación de escritorio como en la aplicación móvil. Esta arquitectura se divide en tres capas principales: la capa de presentación, la capa de lógica de negocio y la capa de datos.

Visión General de la Arquitectura

La capa de presentación incluye las interfaces de usuario de las aplicaciones de escritorio y móvil.

La aplicación de escritorio, desarrollada utilizando JavaFX, proporciona una interfaz de usuario rica y dinámica, permitiendo la creación, lectura, actualización y eliminación de datos en todas las tablas del sistema. Además, la aplicación de escritorio permite la gestión de copias de seguridad. Esta aplicación se conecta directamente a la base de datos mediante el driver JDBC, lo que facilita un acceso rápido y directo a los datos.

Por otro lado, la aplicación de Android está desarrollada utilizando el Android SDK y utiliza la biblioteca Volley para realizar solicitudes de red. A diferencia de la aplicación de escritorio, la aplicación móvil no accede directamente a la base de datos. En su lugar, realiza solicitudes a través de archivos PHP alojados en un servidor. Este enfoque ofrece ventajas significativas en términos de seguridad, ya que el acceso a los datos y la lógica de negocio están centralizados en el servidor, reduciendo el riesgo de exposición de la base de datos directamente al cliente.

Componentes del Sistema

La capa de lógica de negocio está representada por el servidor web, que utiliza Apache para alojar los archivos PHP. Estos archivos PHP implementan la lógica de negocio necesaria para procesar las solicitudes de la aplicación de Android, manejando la autenticación de usuarios, el procesamiento de datos y la gestión de sesiones.

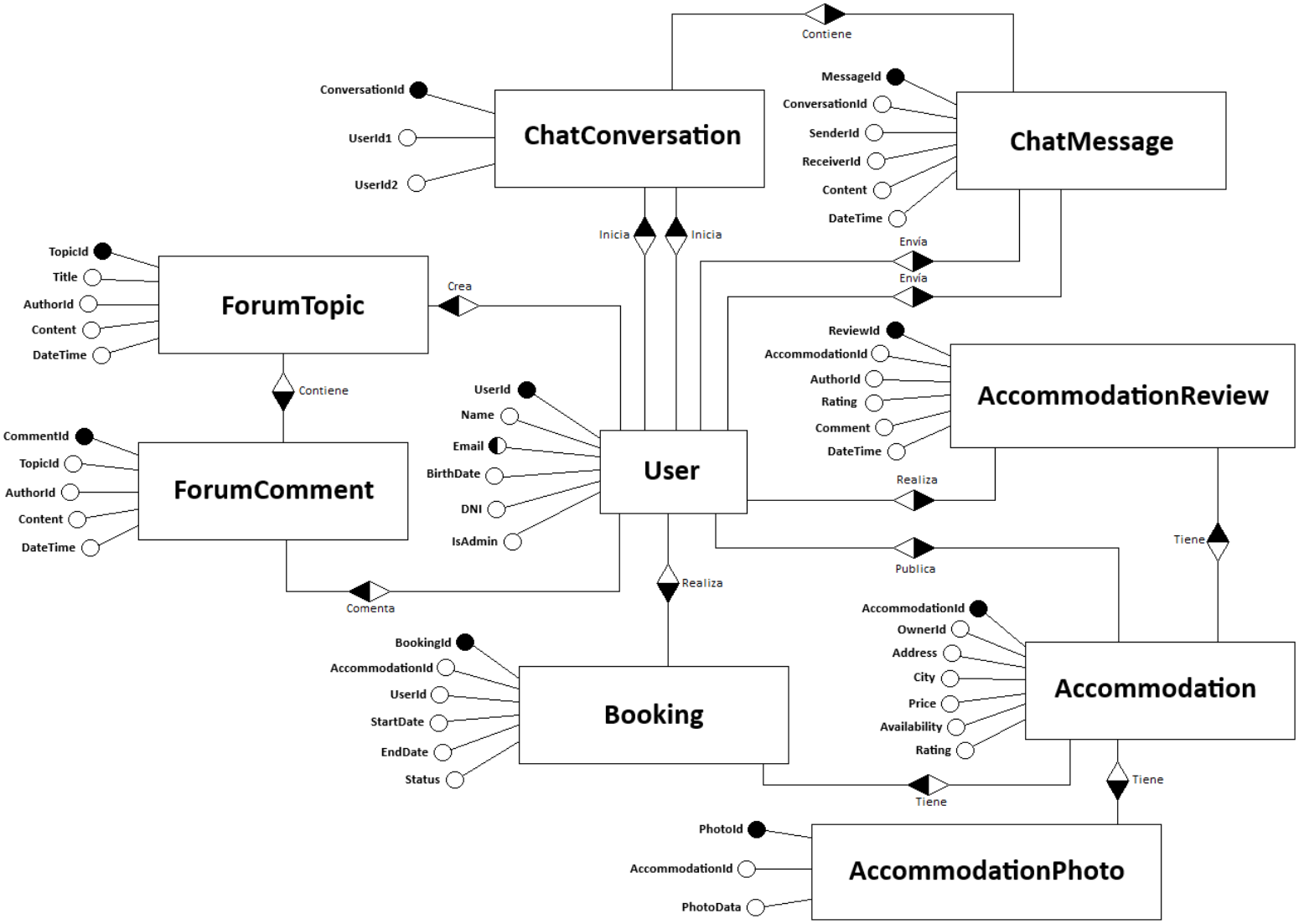
Las aplicaciones cliente, tanto la de escritorio como la de Android, implementan lógica de negocio local. Sin embargo, en la aplicación de Android, las operaciones más complejas y la lógica de negocio pesada se delegan al servidor, lo que no solo mejora la seguridad sino también la eficiencia de la aplicación móvil.

En la capa de datos, MySQL es el sistema de gestión de bases de datos utilizado. La base de datos está diseñada para almacenar información sobre usuarios, alojamientos, reservas, mensajes y actividades del foro.

Las tablas de la base de datos incluyen User, Accommodation, Booking, ChatConversation, ChatMessage, ForumTopic, ForumComment, AccommodationReview y AccommodationPhoto. Cada una de estas tablas está diseñada para manejar diferentes aspectos de la plataforma, desde la gestión de usuarios hasta la reserva de alojamientos y la interacción en los foros.

Diagrama E/R

A continuación se presenta el modelo Entidad/Relación (E/R) actualizado del sistema. Este diagrama muestra las entidades principales y las relaciones entre ellas, proporcionando una visión clara de cómo se estructuran y gestionan los datos en StudyStay. La entidad User es central y se relaciona con varias otras entidades, facilitando funcionalidades como la reserva de alojamientos, la interacción en el foro, la gestión de mensajes en el chat y la publicación de reseñas y fotos de los alojamientos. Las relaciones entre las entidades aseguran la integridad referencial y permiten un acceso eficiente a la información relevante.



5. Diseño y Desarrollo

El apartado de Diseño y Desarrollo describe cómo se ha estructurado y llevado a cabo el proceso de implementación de las diferentes partes del sistema StudyStay. Este apartado está dividido en varias secciones que cubren el ciclo de vida del desarrollo, la arquitectura y diseño UML, el diseño de interfaces, la implementación de funcionalidades específicas, la planificación del proyecto, y otros aspectos importantes como la responsividad y accesibilidad.

Ciclo de Vida del Desarrollo

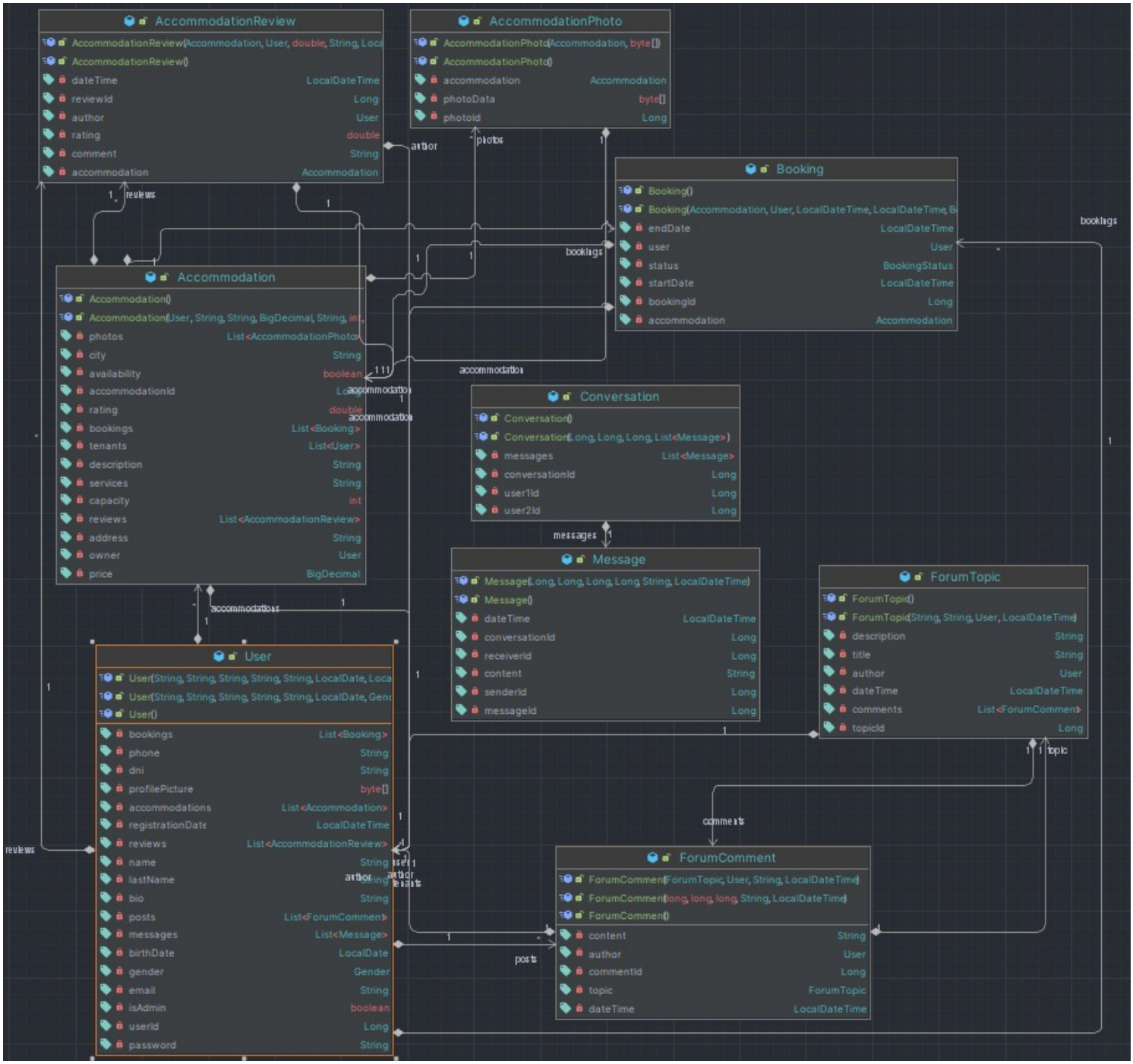
Para el proyecto se ha adoptado el modelo de ciclo de vida Kanban personal debido a sus ventajas en términos de flexibilidad y gestión visual del flujo de trabajo. El Kanban personal permite manejar tareas de manera eficiente, priorizando actividades y asegurando un progreso continuo sin sobrecargar al desarrollador. Este enfoque facilita la adaptación a cambios y la mejora continua del sistema, lo que es crucial para un proyecto de esta naturaleza.

Arquitectura y Diseño UML

Una vez establecida la base de datos, el siguiente paso fue trasladar esto a código Java. La arquitectura del sistema sigue el patrón de diseño MVC (Modelo-Vista-Controlador), con una clara separación de las responsabilidades. En la aplicación de escritorio, se utiliza MVC junto con un paquete DAO (Data Access Object) para gestionar la conexión con la base de datos. La estructura del proyecto incluye los paquetes DAO, Model, View y Controller.

En la aplicación de Android, también se sigue el patrón MVC. La lógica de acceso a datos se maneja mediante PHP en el servidor, facilitando la interacción a través de la biblioteca Volley. La estructura del proyecto está organizada en los paquetes Model, View y Controller. A continuación se muestra el diagrama UML de las principales clases y sus interacciones dentro del sistema.

UMLs App Escritorio




UserDAO
UserDAO()
PASSWORD String
JDBC_URL String
USERNAME String
getAll() List<User>
getById(Long) User?
delete(Long) boolean
mapUser(ResultSet) User
findByEmail(String) User
create(User) boolean
update(User) boolean
authenticate(String, String) boolean
BookingDAO
BookingDAO()
JDBC_URL String
PASSWORD String
USERNAME String
getAll() List<Booking>
mapBooking(ResultSet) Booking
delete(Long) boolean
update(Booking) boolean
create(Booking) boolean
«create»
«create»
ForumCommentDAO
ForumCommentDAO()
PASSWORD String
USERNAME String
JDBC_URL String
mapComment(ResultSet) ForumComment
create(ForumCommen) boolean
update(ForumCommen) boolean
getAll() List<ForumCommen
getCommentsByTopic(Long) List<ForumCommen
delete(Long) boolean
AccommodationPhotoDAO
AccommodationPhotoDAO()
USERNAME String
PASSWORD String
JDBC_URL String
update(AccommodationPhot) boolean
getAll() List<AccommodationPhoto
delete(Long) boolean
create(AccommodationPhot) boolean
mapAccommodationPhot(ResultSet) AccommodationPhotc
ConversationDAO
ConversationDAO()
JDBC_URL String
USERNAME String
PASSWORD String
createConversation(Conversation) boolean
updateConversation(Conversation) boolean
getMessagesForConversation(Long) List<Message>
deleteConversation(Long) boolean
getAllConversations() List<Conversation>
mapConversation(ResultSet) Conversation
AccommodationDAO
AccommodationDAO()
PASSWORD String
connection Connection
JDBC_URL String
USERNAME String
getPhotosForAccommodation(Long) List<AccommodationPhoto>
getReviewsByAccommodation(Long) List<AccommodationReview>
getTenantsForAccommodation(Long) List<User>
mapAccommodation(ResultSet) Accommodation
update(Accommodation) boolean
getById(Long) Accommodation?
create(Accommodation) boolean
getAll() List<Accommodation>
delete(Long) boolean
getByOwner(Long) List<Accommodation>
ForumTopicDAO
ForumTopicDAO()
PASSWORD String
JDBC_URL String
USERNAME String
create(ForumTopic) boolean
getComments(Long) List<ForumCommen
mapTopic(ResultSet) ForumTopic
delete(Long) boolean
getAll() List<ForumTopic>
update(ForumTopic) boolean
getById(Long) ForumTopic
AccommodationReviewDAO
AccommodationReviewDAO()
USERNAME String
PASSWORD String
JDBC_URL String
getReviewsForAccommodation(Long) List<AccommodationReview>
getAll() List<AccommodationReview>
mapReview(ResultSet) AccommodationReview
create(AccommodationReview) boolean
delete(Long) boolean
update(AccommodationReview) boolean
MessageDAO
MessageDAO()
JDBC_URL String
USERNAME String
PASSWORD String
createMessage(Message) boolean
modifyMessage(Message) boolean
getAllMessages() List<Message>
deleteMessage(Long) boolean
mapMessage(ResultSet) Message
getMessagesByConversation(Long) List<Message>


MessageController
MessageController()
messageDAO MessageDAO
createMessage(Message) boolean
deleteMessage(Long) boolean
getMessagesByConversation(Long) List<Message>
getAllMessages() List<Message>
modifyMessage(Message) boolean
UserController
UserController()
userDAO UserDAO
create(User) boolean
getAll() List<User>
findByEmail(String) User
update(User) boolean
getById(Long) User
delete(Long) boolean
authenticate(String, String) boolean
AccommodationPhotoController
AccommodationPhotoController()
photoDAO AccommodationPhotoDAO
updatePhoto(AccommodationPhoto) boolean
getAllPhotos() List<AccommodationPhoto>
deletePhoto(Long) boolean
createPhoto(AccommodationPhoto) boolean
AccommodationReviewController
AccommodationReviewController()
accommodationReviewDAO AccommodationReviewDAO
createReview(AccommodationReview) boolean
updateReview(AccommodationReview) boolean
getAllReviews() List<AccommodationReview>
deleteReview(Long) boolean
ForumTopicController
ForumTopicController()
forumTopicDAO ForumTopicDAO
deleteTopic(Long) boolean
getTopic(Long) ForumTopic
getAllTopics() List<ForumTopic>
updateTopic(ForumTopic) boolean
getComments(Long) List<ForumComment>
createTopic(ForumTopic) boolean
ForumCommentController
ForumCommentController()
forumCommentDAO ForumCommentDAO
getCommentsByTopic(Long) List<ForumComment>
createComment(ForumComment) boolean
getAllComments() List<ForumComment>
updateComment(ForumComment) boolean
deleteComment(Long) boolean
ConversationController
ConversationController()
conversationDAO ConversationDAO
createConversation(Conversation) boolean
updateConversation(Conversation) boolean
deleteConversation(Long) boolean
getAllConversations() List<Conversation>
AccommodationController
AccommodationController()
accommodationDAO AccommodationDAO
getAccommodationById(Long) Accommodation
getTenantsForAccommodation(Long) List<User>
updateAccommodation(Accommodation) boolean
getAllAccommodations() List<Accommodation>
createAccommodation(Accommodation) boolean
getReviewsByAccommodation(Long) List<AccommodationReview>
deleteAccommodation(Long) boolean
getAccommodationsByOwner(Long) List<Accommodation>
BookingController
BookingController()
bookingDAO BookingDAO
updateBooking(Booking) boolean
getAllBookings() List<Booking>
deleteBooking(Long) boolean
createBooking(Booking) boolean

UMLs App Android

El tamaño de los diagramas de clases de android supone un verdadero problema para insertarlos directamente en el documento, es por ello que tanto los de android, como los de la aplicación de escritorio se adjuntarán en formato de

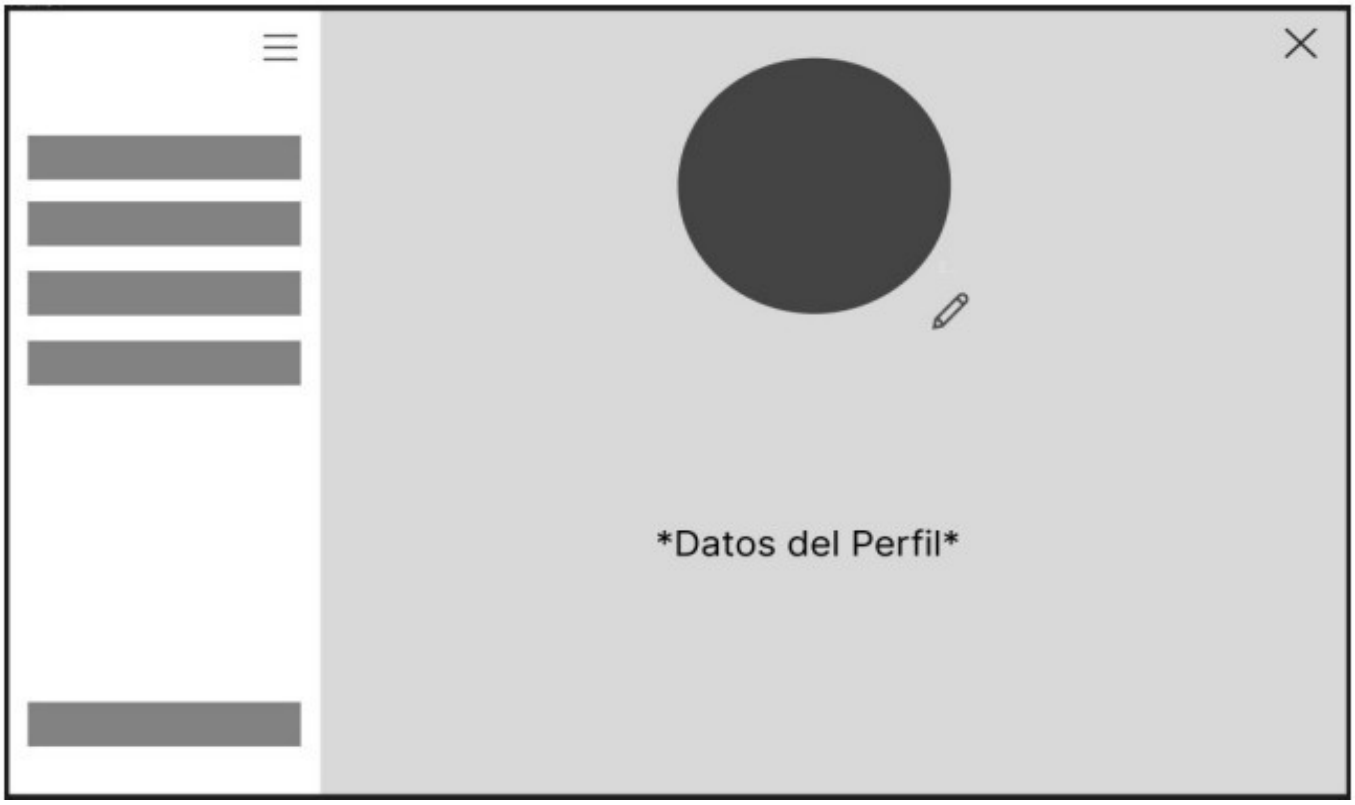
Diseño de interfaces

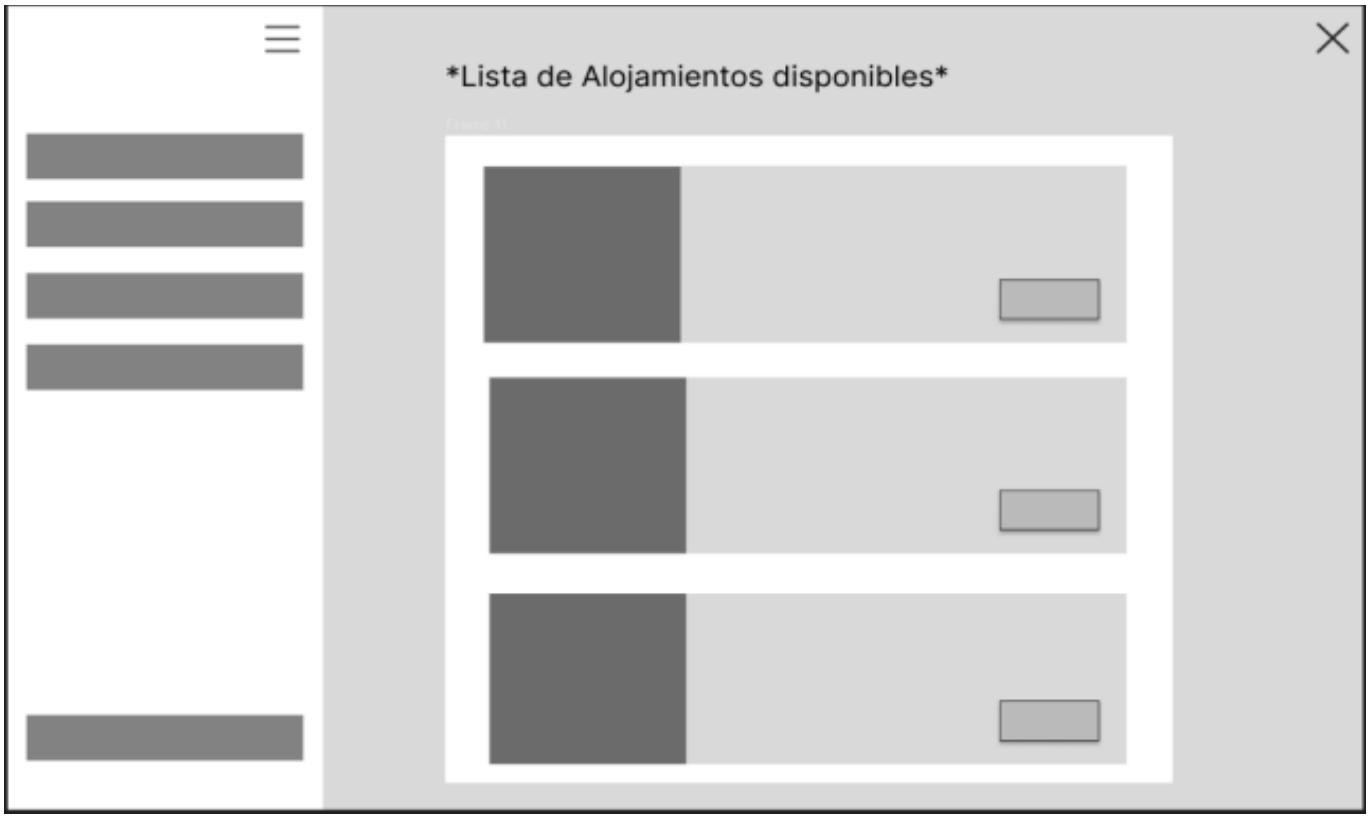
Aplicación de Escritorio

La pantalla de inicio de sesión (Login) incluye campos de texto para el correo electrónico y la contraseña del usuario, un botón para iniciar sesión, y un enlace para registrarse o recuperar la contraseña.

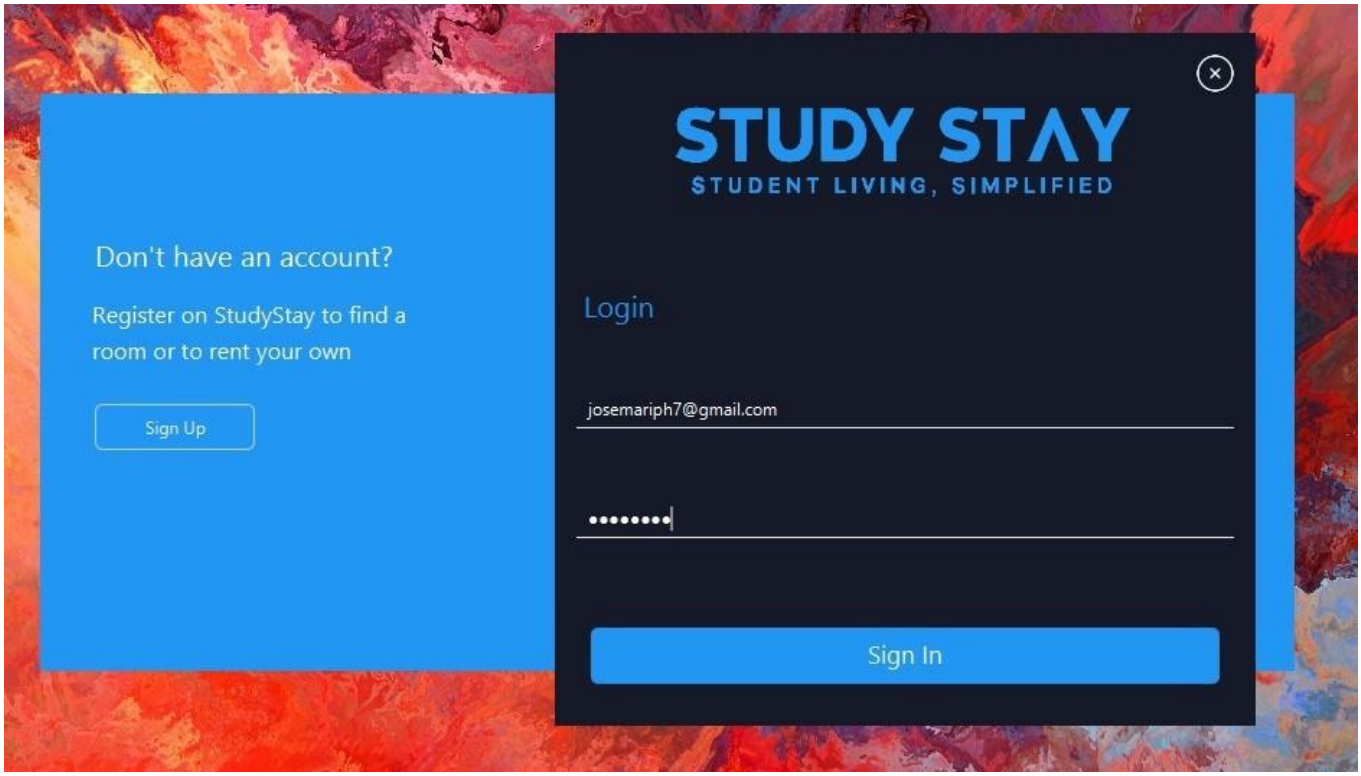
Al iniciar sesión, se accede al dashboard, que da la bienvenida al administrador con su nombre. Desde el dashboard, se pueden realizar operaciones CRUD sobre los datos de la base de datos en distintas pantallas del mismo. Además, incluye un botón de refresh, un botón de backup, una ventana de acerca de, y un perfil donde se puede subir una foto de perfil o modificar los datos personales.

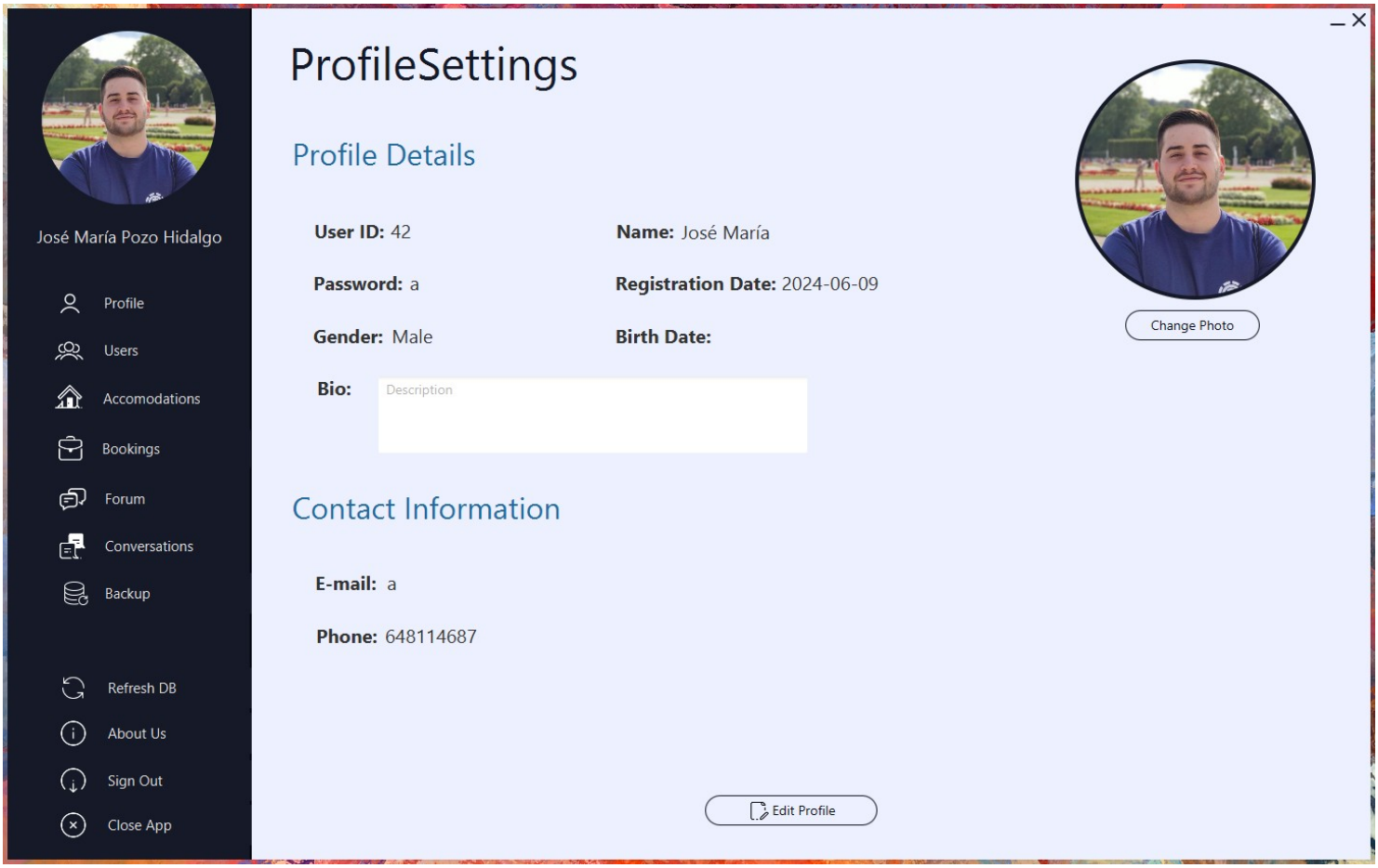
La pantalla de gestión de usuarios muestra una lista de usuarios registrados, con botones para añadir, editar o eliminar usuarios, y campos para buscar usuarios por diferentes criterios. La pantalla de gestión de alojamientos presenta una lista de alojamientos disponibles, con opciones para añadir, editar o eliminar alojamientos, y filtros para ordenar la lista. Por último, la pantalla de gestión de reservas permite ver y gestionar las reservas actuales y pasadas, con opciones para cancelar o modificar una reserva.

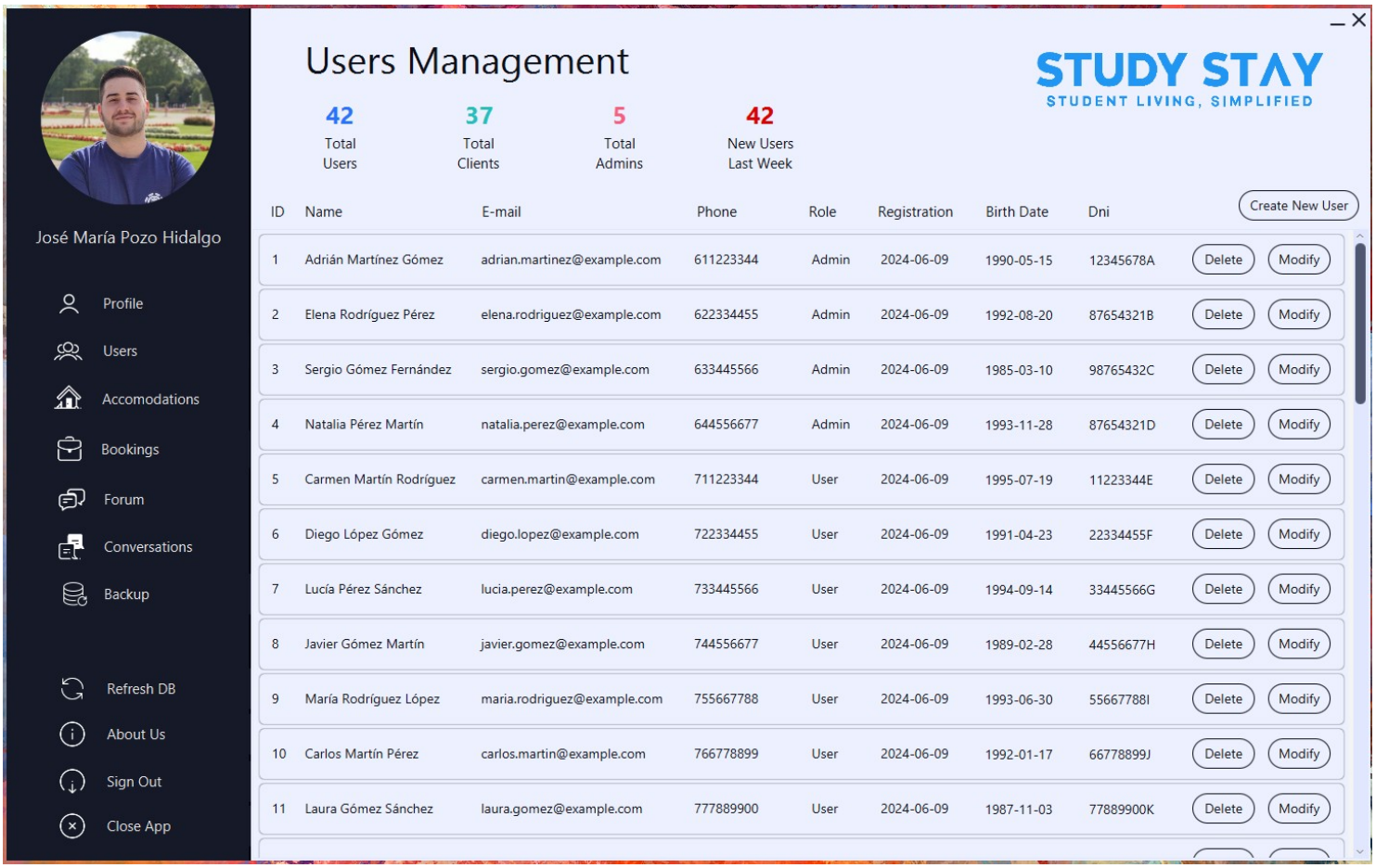




Estos son algunos bocetos básicos que se realizaron durante el diseño de interfaces de la aplicación de escritorio. Una vez se implementaron en javaFX este fue el resultado:


STUDY STAY
STUDENT LIVING, SIMPLIFIED
Don't have an account?
Register on StudyStay to find a room or to rent your own
Sign Up
Login
josemariph7@gmail.com
Sign In


José María Pozo Hidalgo
Profile
Users
Accomodations
Bookings
Forum
Conversations
Backup
Refresh DB
About Us
Sign Out
Close App
ProfileSettings
Profile Details
User ID: 42
Name: José María
Password: a
Registration Date: 2024-06-09
Gender: Male
Birth Date:
Bio:
Description
Change Photo
Contact Information
E-mail: a
Phone: 648114687
Edit Profile



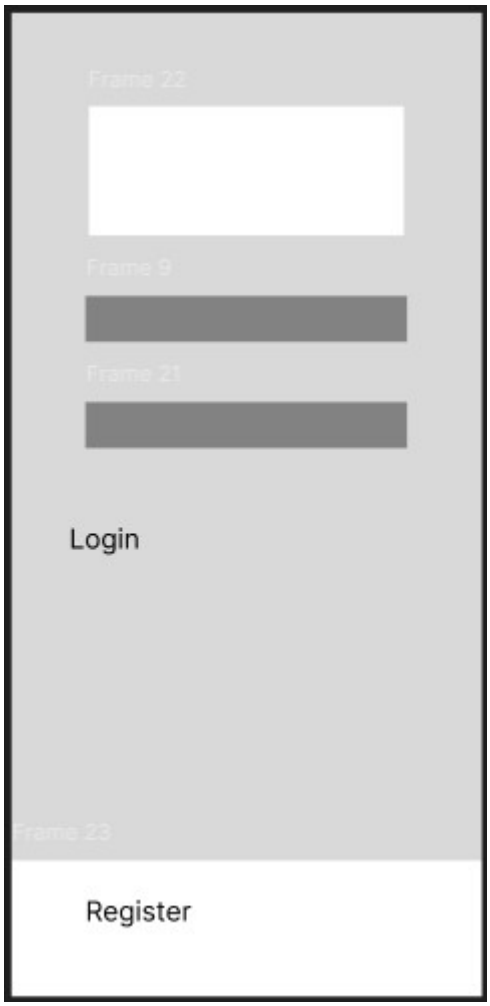
Aplicación de Android

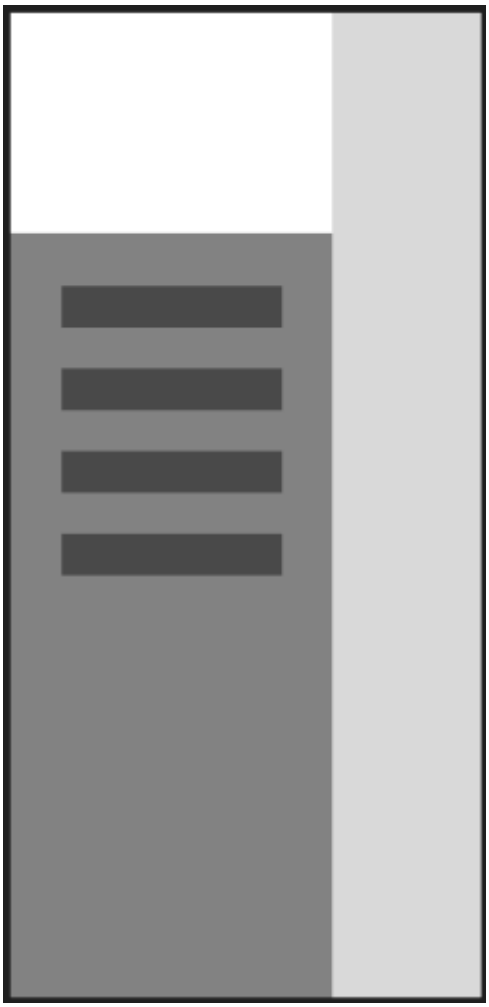
La pantalla de inicio de sesión (Login) de la aplicación de Android tiene campos de texto para el correo electrónico y la contraseña del usuario, un botón para iniciar sesión, y un enlace para registrarse o recuperar la contraseña.

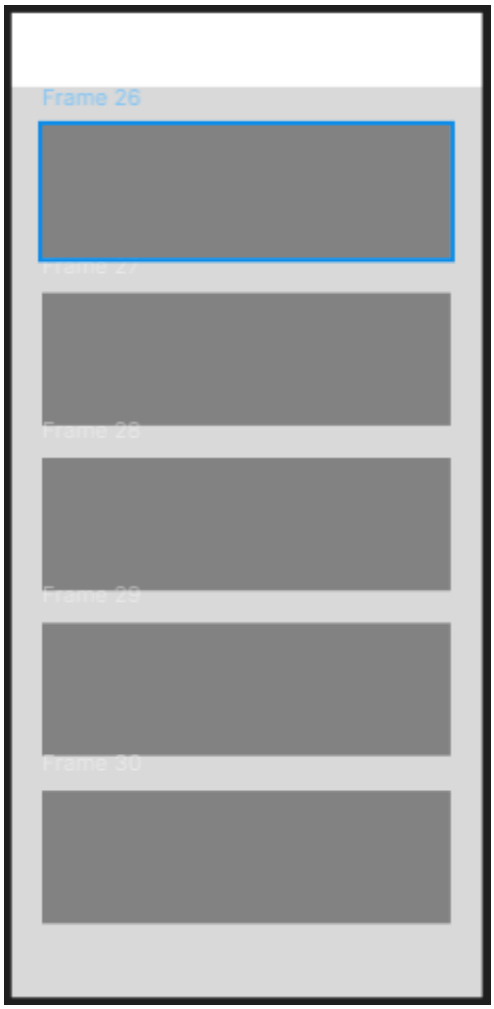
El activity principal permite el login y registro de usuarios. Al iniciar sesión, se accede a una activity que contiene un navigator view desde el cual se puede navegar por las principales secciones de la aplicación, como chat, foro, perfil, o alojamientos. En la sección de alojamientos, los usuarios pueden publicar alojamientos, alquilarlos o hacer reseñas de los mismos.

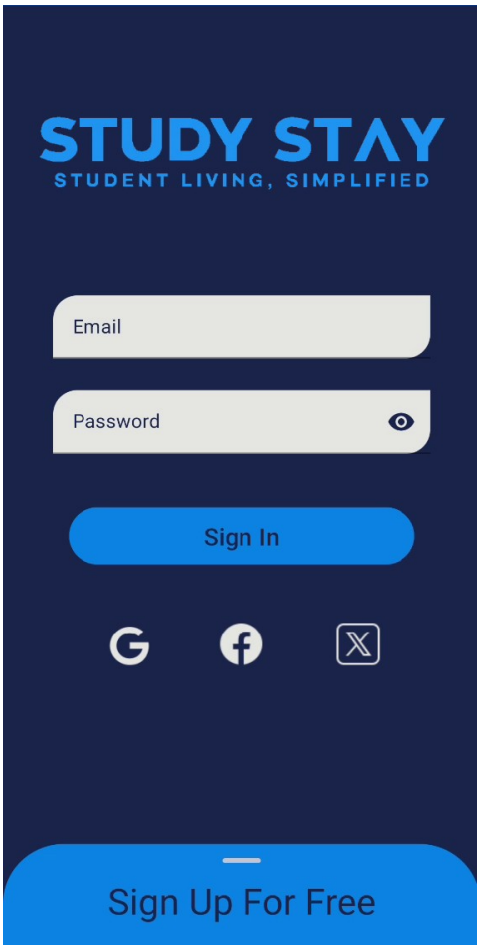
La pantalla de perfil permite al usuario ver y editar su información personal. La pantalla de chat muestra una lista de conversaciones con otros usuarios y botones para iniciar una nueva conversación. La pantalla de foro lista los temas del foro, con opciones para crear un nuevo tema o responder a temas existentes. La pantalla de publicación y alquiler de habitaciones permite a los usuarios ver las habitaciones disponibles, publicar nuevos anuncios o realizar reservas.

A continuación se muestran algunos bocetos iniciales que sirvieron para establecer la distribución de los elementos principales en la interfaz de la app de android y acto seguido, algunas capturas de el resultado final de las mismas.

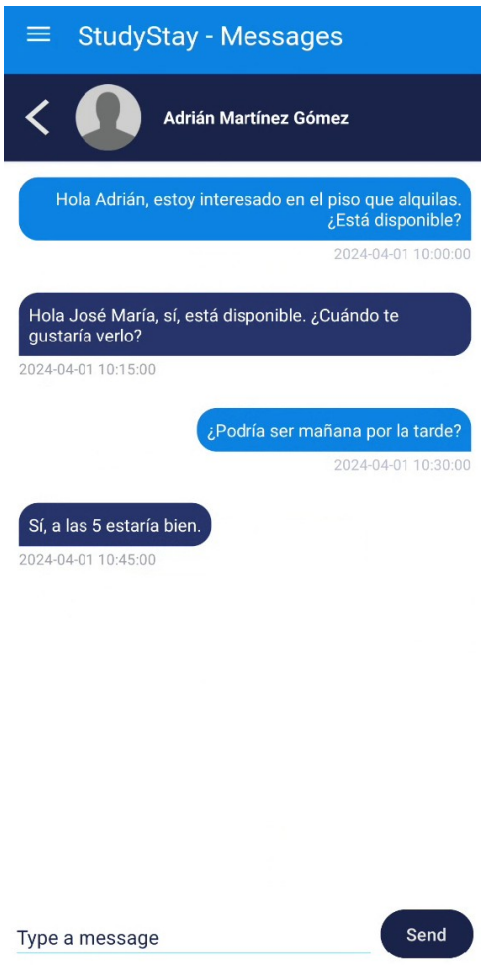

Frame 22
Frame 9
Frame 21
Login
Frame 23
Register

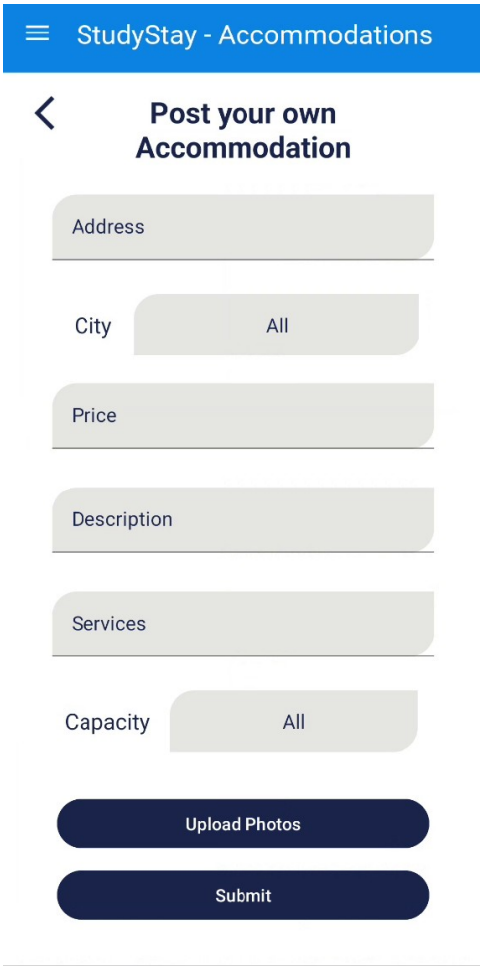



Frame 26
Frame 27
Frame 28
Frame 29
Frame 30


STUDY STAY
STUDENT LIVING, SIMPLIFIED
Email
Password
Sign In
Sign Up For Free


StudyStay
José María Pozo Hidalgo
josemariph7@gmail.com
1997-06-08
648114687
2024-03-05 17:00:00
53737432W
Account Settings
Rented
Listed


StudyStay - Messages
Adrián Martínez Gómez
Hola Adrián, estoy interesado en el piso que alquilas. ¿Está disponible?
2024-04-01 10:00:00
Hola José María, sí, está disponible. ¿Cuándo te gustaría verlo?
2024-04-01 10:15:00
¿Podría ser mañana por la tarde?
2024-04-01 10:30:00
Sí, a las 5 estaría bien.
2024-04-01 10:45:00
Type a message
Send


StudyStay - Accommodations
Post your own Accommodation
Address
City
All
Price
Description
Services
Capacity
All
Upload Photos
Submit

Planificación del Proyecto

El diagrama de Gantt se incluye en esta sección para ilustrar la planificación temporal del proyecto. Este diagrama visualiza las diferentes fases y tareas del proyecto a lo largo del tiempo, ayudando a entender el flujo y la coordinación de actividades.

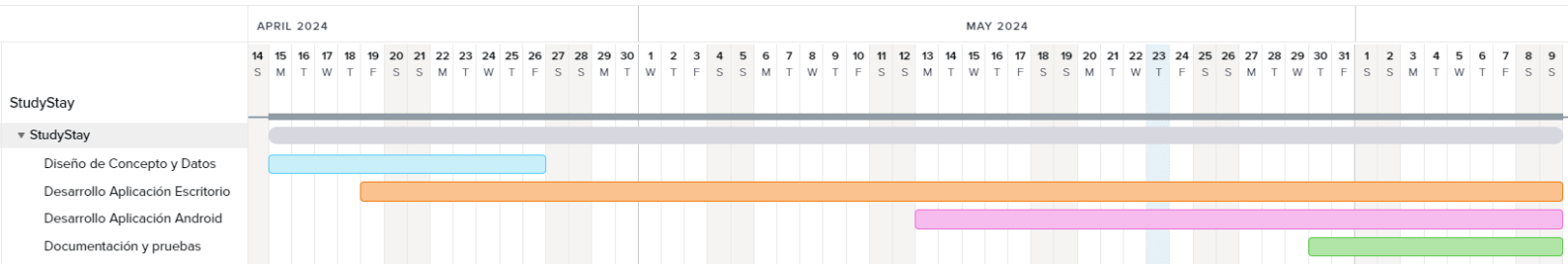
En primer lugar, se llevó a cabo el diseño de datos y el desarrollo del servidor. Esta fase incluyó la creación del modelo de datos, la implementación de la base de datos y la configuración del servidor web para alojar los scripts PHP.

A continuación, se procedió con el desarrollo de la aplicación de escritorio. Esta fase involucró el diseño y la implementación de las interfaces de usuario utilizando JavaFX, así como la integración con la base de datos mediante JDBC.

Seguidamente, se desarrolló la aplicación de Android. En esta fase se diseñaron e implementaron las interfaces móviles, se integró Google Maps y se configuraron las solicitudes de red utilizando Volley para comunicarse con el servidor.

Finalmente, se llevó a cabo la documentación del proyecto. Esta fase incluyó la elaboración de manuales de usuario y programador, la creación de diagramas UML y E/R, y la compilación de toda la información relevante sobre el desarrollo y uso de la aplicación.

Mientras que algunos diagramas, como los UML y el E/R, se desarrollaron durante el transcurso de todo el proceso, se hicieron ajustes y refinamientos continuos basados en el progreso y los cambios en el proyecto.



6. Configuración del Entorno de Desarrollo

El desarrollo de StudyStay requiere de un conjunto específico de herramientas de software que facilitan la creación, prueba y despliegue de la aplicación.

Aplicación de Escritorio

IDE: IntelliJ IDEA (versión 2021.1 o superior)

- Configuración: Descarga e instala IntelliJ IDEA desde aquí. Configura IntelliJ IDEA para utilizar JDK 21. Ve a File > Project Structure > Project y selecciona JDK 21.

DK: OpenJDK 21

- Configuración: Asegúrate de que JAVA_HOME esté configurado correctamente en tu sistema y que el JDK 21 esté seleccionado en la configuración del proyecto en IntelliJ IDEA.

JavaFX: Versión 15 o superior

- Configuración: Descarga JavaFX SDK desde Gluon. Añade las librerías de JavaFX en las dependencias del proyecto en IntelliJ IDEA:

```
<dependency>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-controls</artifactId>
        <version>15.0.1</version>
</dependency>
<dependency>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-fxml</artifactId>
        <version>15.0.1</version>
</dependency>
```

MySQL: Versión 8.0

- Configuración: Instala MySQL Server y configura una base de datos para la aplicación. Asegúrate de que el servidor MySQL esté en funcionamiento y accesible.

Driver JDBC: MySQL Connector/J 8.0

- Configuración: Incluye el driver JDBC en las dependencias del proyecto para permitir la conexión con la base de datos MySQL:

```
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.23</version>
</dependency>
```

Scene Builder:

- Configuración: Descarga e instala Scene Builder desde Gluon. Configura IntelliJ IDEA para utilizar Scene Builder: ve a File > Settings > Languages & Frameworks > JavaFX y configura la ruta a Scene Builder.

Maven:

- Configuración: Utiliza Maven para la gestión de dependencias. Asegúrate de tener el plugin de Maven configurado en IntelliJ IDEA. Configura el archivo pom.xml con las dependencias necesarias:

```
<dependencies>
        <dependency>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-controls</artifactId>
                <version>15.0.1</version>  </dependency>
        <dependency>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-fxml</artifactId>
                <version>15.0.1</version>
        </dependency>
        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>8.0.23</version>
        </dependency>
</dependencies>
```

Aplicación de Android

IDE: Android Studio (versión Arctic Fox 2020.3.1 o superior)

- Configuración: Descarga e instala Android Studio desde aquí. Configura Android Studio para utilizar el JDK 8 integrado.

JDK: OpenJDK 8

- Configuración: Android Studio incluye un JDK 8 integrado. Asegúrate de que el proyecto utilice esta versión de JDK.

SDK de Android:

- Configuración: Abre el SDK Manager desde `File > Settings > Appearance & Behavior > System Settings > Android SDK`. Instala las últimas versiones del SDK de Android y las herramientas de compilación necesarias.

Librerías de Android:

- Configuración: Añade las siguientes dependencias en el archivo `build.gradle` del módulo:

```
dependencies {
        implementation 'com.android.volley:volley:1.2.1'
        implementation 'com.google.android.gms:play-services-maps:17.0.0'
        implementation 'com.github.bumptech.glide:glide:4.12.0'
        annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
}
```

Servidor Web: Apache y PHP

- Configuración: Instala Apache y PHP en un servidor. Configura el servidor para alojar los archivos PHP que manejarán las solicitudes de la aplicación de Android. Asegúrate de que el servidor MySQL esté configurado y accesible.

Integración de Google Maps:

- Configuración: Configura un proyecto en Google Cloud Console y habilita las APIs necesarias. Añade la clave de la API de Google Maps en el archivo AndroidManifest.xml de la aplicación:

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="YOUR_API_KEY"/>
```

Pruebas y Emuladores:

- Configuración: Configura emuladores de Android desde el AVD Manager en Android Studio para probar la aplicación en diferentes dispositivos y versiones de Android.

Esta configuración asegura que el entorno de desarrollo esté correctamente preparado para trabajar con las aplicaciones de escritorio y Android de StudyStay, proporcionando todas las herramientas y librerías necesarias para un desarrollo eficiente y efectivo.

7. Pruebas

El apartado de Pruebas detalla las metodologías y herramientas utilizadas para asegurar que las aplicaciones de escritorio y Android de StudyStay funcionan correctamente y cumplen con los requisitos especificados. Se describen las pruebas unitarias, de integración y manuales realizadas durante el desarrollo.

Pruebas unitarias

En ambos proyectos, de escritorio y Android, se han utilizado pruebas unitarias con JUnit para verificar la correcta funcionalidad de las clases y métodos. Estas pruebas se han agrupado en un paquete específico dentro de cada proyecto, facilitando su organización y ejecución.

Pruebas Funcionales

Las pruebas funcionales se realizaron para asegurar que todas las funcionalidades de la aplicación cumplen con los requisitos especificados y funcionan según lo esperado. Estas pruebas se llevaron a cabo ejecutando la aplicación y utilizando todas sus funcionalidades para detectar errores y mejorar la experiencia de usuario.

Pruebas Funcionales Realizadas:

- Pruebas de Funcionalidad: Se probó cada funcionalidad de la aplicación, como la gestión de usuarios, la publicación y alquiler de alojamientos, la participación en foros y el uso del chat.
- Pruebas de Interfaz de Usuario: Se verificó que todas las interfaces fueran intuitivas y fáciles de usar, y que los elementos visuales se mostraran correctamente en diferentes resoluciones y tamaños de pantalla.
- Pruebas de Rendimiento: Se evaluó el rendimiento de la aplicación bajo diferentes condiciones de carga para asegurar que funcionara de manera eficiente.
- Pruebas de Seguridad: Se realizaron pruebas básicas de seguridad, como la validación de entradas para prevenir inyecciones SQL y asegurar que los datos sensibles estuvieran protegidos.

8. Conclusiones y Trabajo Futuro

Este proyecto, el más grande que he realizado hasta ahora, ha sido una experiencia que me ha permitido investigar y aprender más allá de lo que hubiera podido lograr con cualquier otro método. La complejidad y la escala del proyecto me han llevado a expandir mis conocimientos y capacidades de una manera mucho más rápida y efectiva.

Realizar el proyecto que tenía pensado desde un principio durante las prácticas de empresa no ha sido tarea fácil. Sin embargo, a pesar de los desafíos y del hecho de que quizás no pude profundizar

tanto como me hubiera gustado en algunos aspectos de las aplicaciones, el estado en el que consigo entregarlas me deja satisfecho. Siento que he logrado un balance adecuado entre funcionalidad y calidad, considerando las limitaciones de tiempo.

Soy plenamente consciente de que hay áreas mejorables en el proyecto. La eficiencia del sistema y la optimización del acceso a datos son dos de las áreas más evidentes para futuras mejoras. Implementar un sistema de paginación para mostrar listas de elementos y refactorizar algunos procesos que se llevan a cabo durante la ejecución de las aplicaciones son pasos claros a seguir para mejorar el rendimiento y la eficiencia.

También en el aspecto funcional tengo ideas que no han podido ser implementadas y que ofrecerían una capa mas de profundidad en la plataforma, sin embargo, considero que el balance ha sido muy positivo, pues el reto me ha llevado a desarrollar la aplicación más extensa y compleja que he realizado hasta día de hoy, ampliando mis conocimientos y empujándome a demostrar de lo que soy capaz.