StudentsRooms

Francisco Toribio Respaldo

Proyecto CFGS Desarrollo de Aplicaciones Multiplataforma

Propósito principal del proyecto

- Proponer una aplicación donde los estudiantes puedan encontrar habitaciones en alquiler cerca de sus centros de estudios y los propietarios de los pisos tengan un sitio donde ofertarlas.

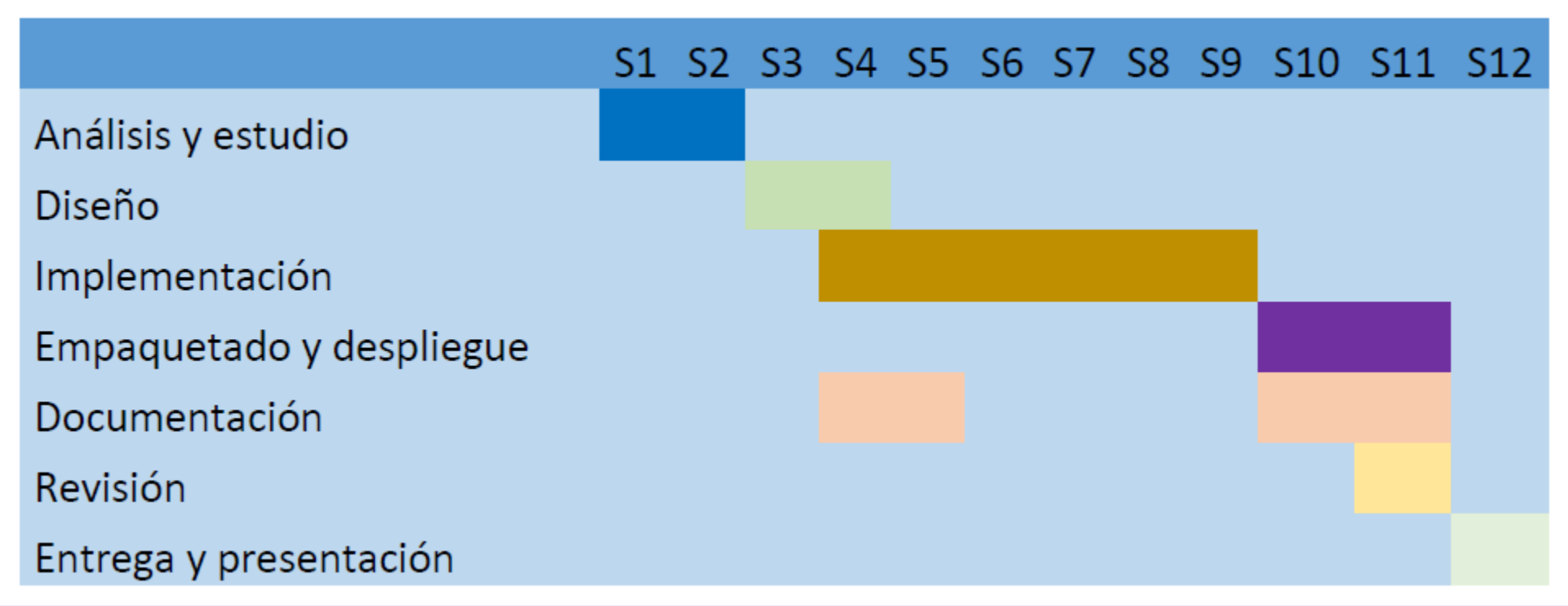
Propósito personal

- Realizar una presentación de las habilidades técnicas adquiridas durante la formación.

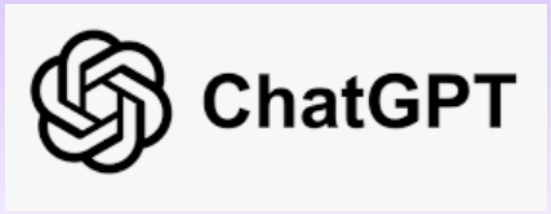
Resumen de requisitos

- Arquitectura robusta y escalable
- Integración sencilla con bases de datos
- Seguridad integrada con Spring Security
- Estandarización y buenas prácticas
- Compatibilidad multiplataforma
- Productividad y rapidez en el desarrollo
- Flexibilidad para escalar hacia otras plataformas (iOS, web) sin cambiar la lógica del servidor.

Planificación temporal de las tareas



Tecnologías usadas

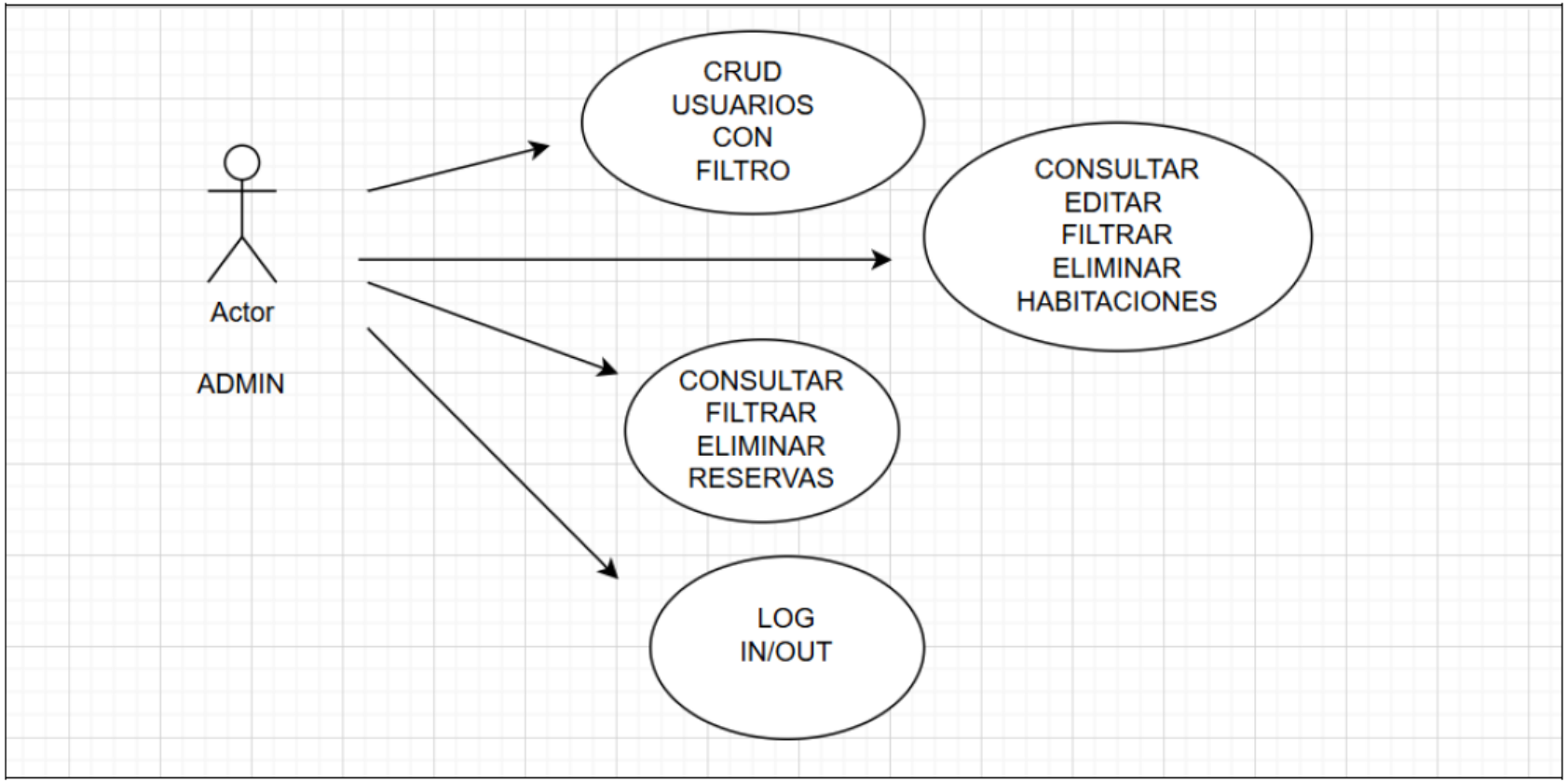


Análisis

- Casos de uso
- Modelo de datos
- Requisitos funcionales
- Requisitos no funcionales

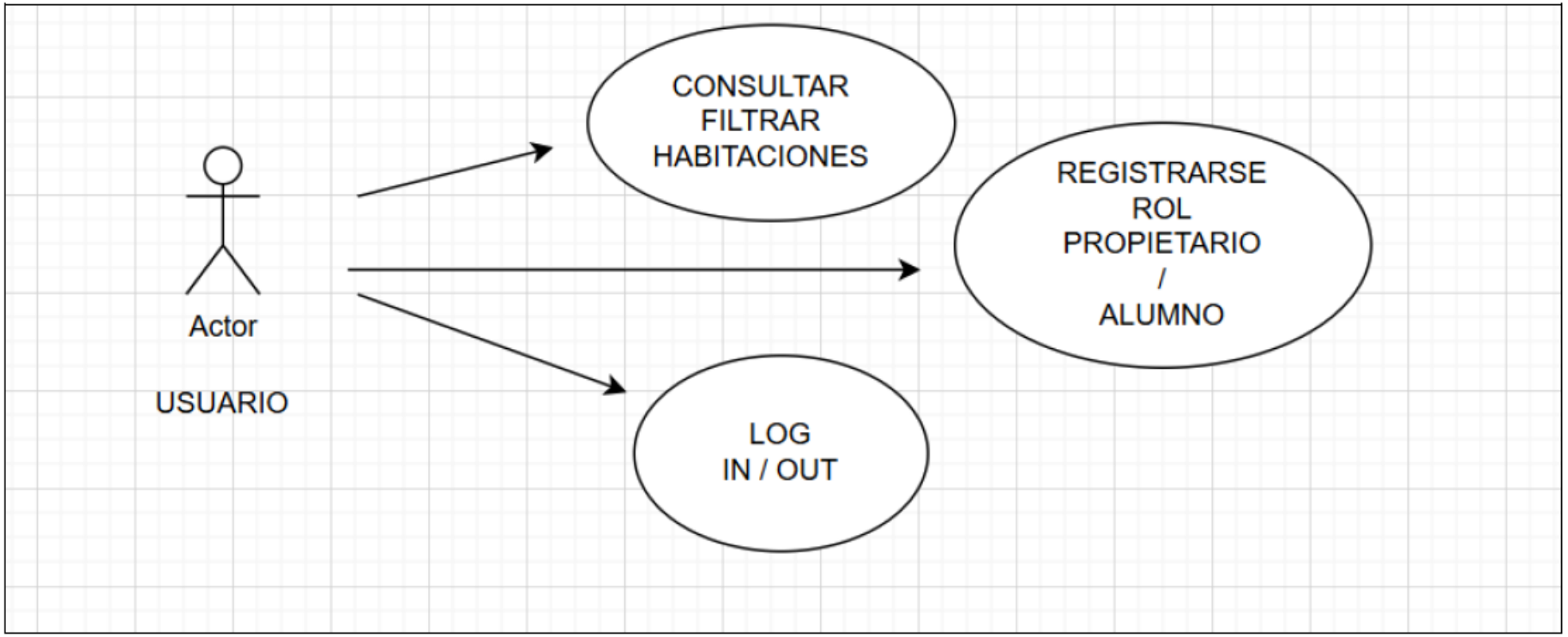
Casos de uso

Diagrama de casos de uso del administrador (ADMIN).



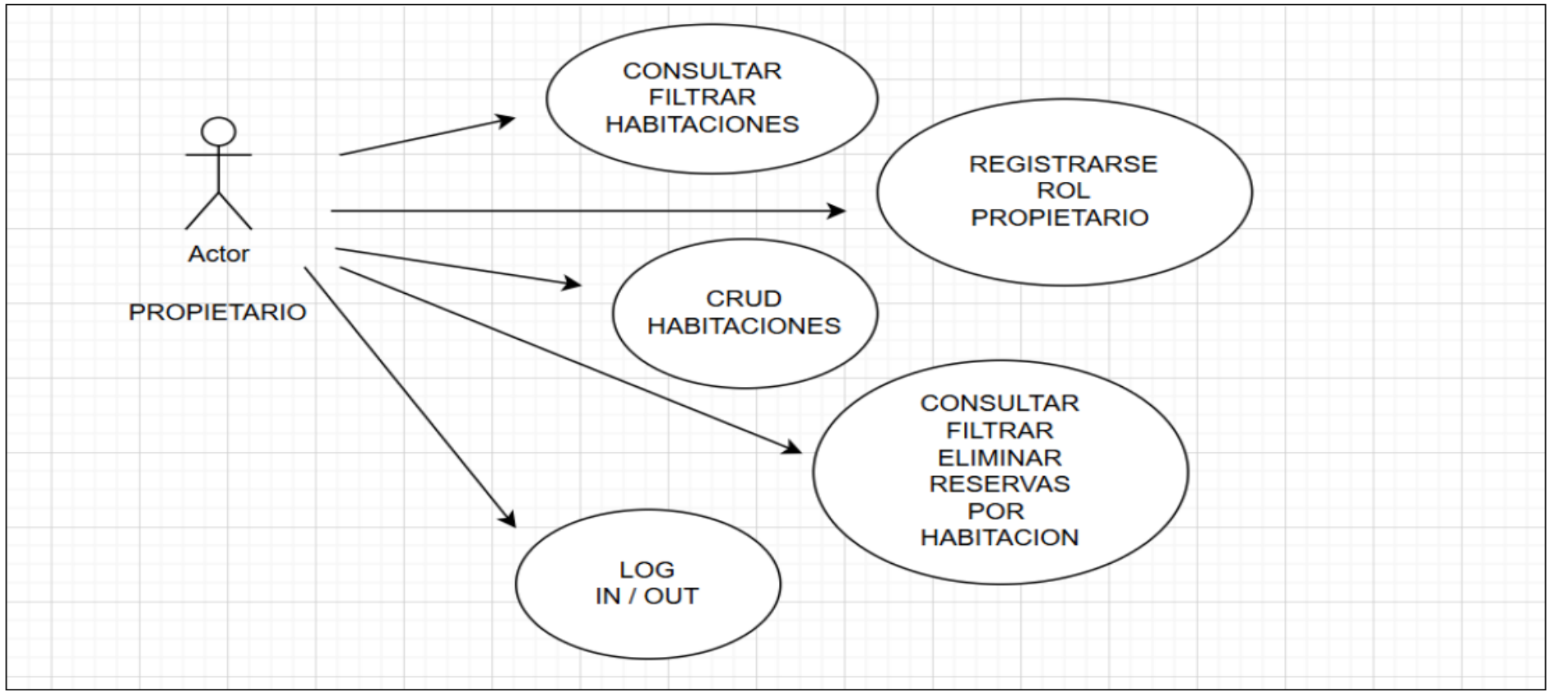
Casos de uso

Diagrama de casos de uso del USUARIO.



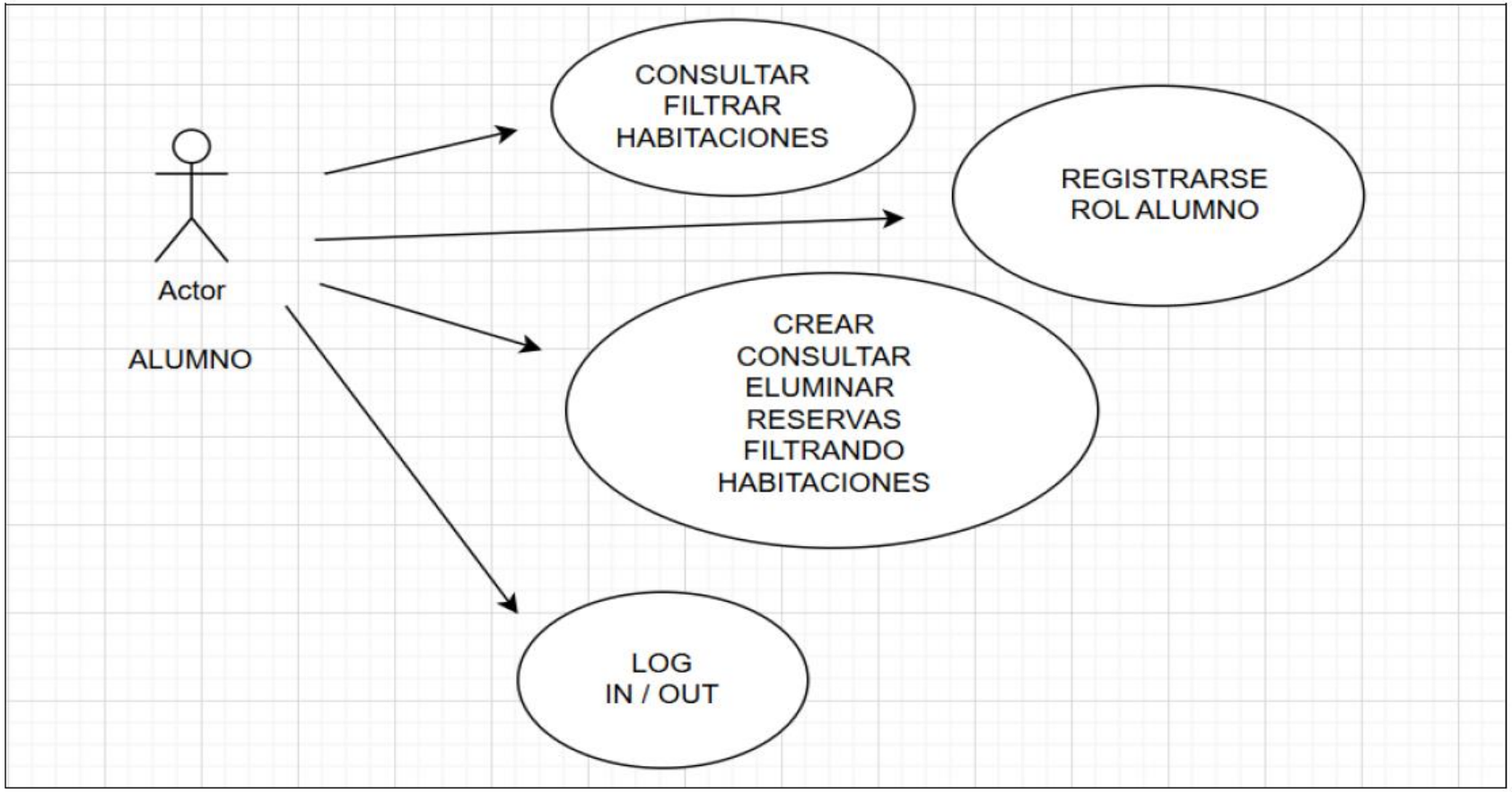
Casos de uso

Diagrama de casos de uso del PROPIETARIO.



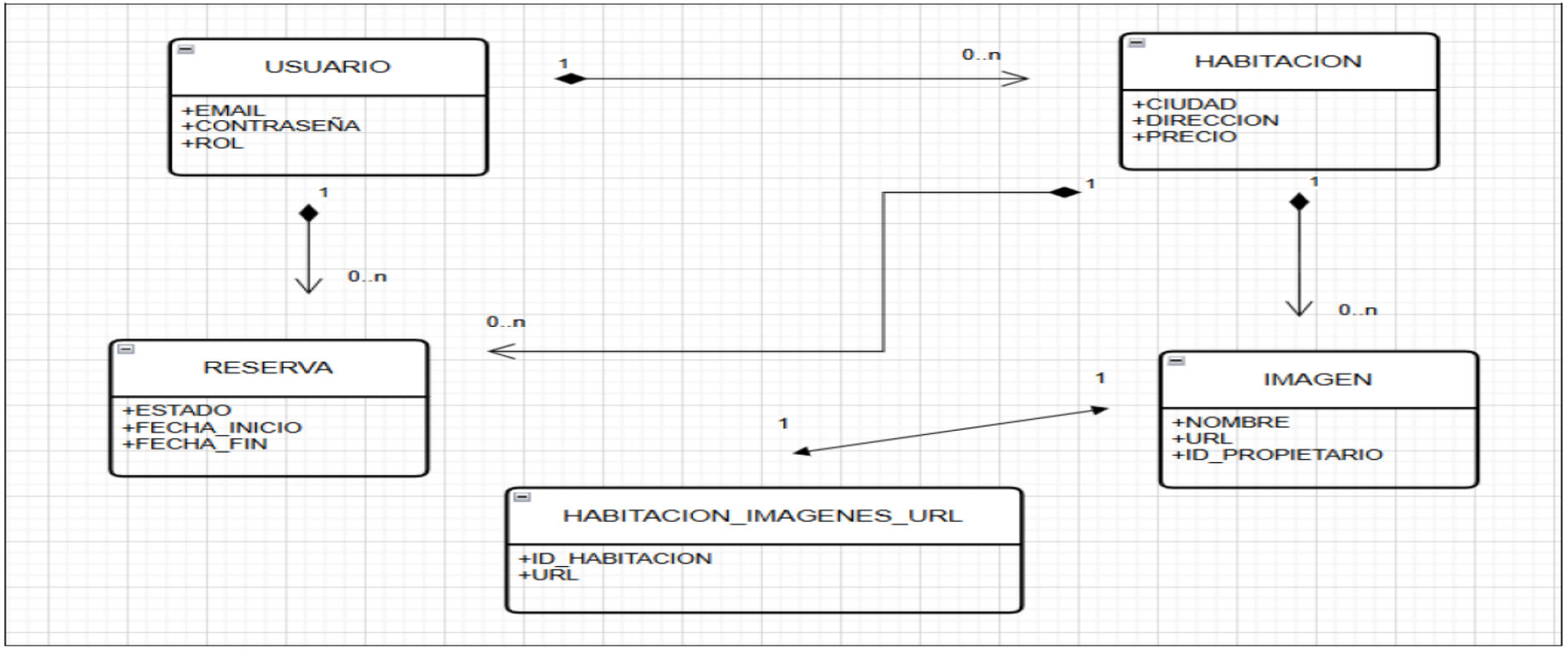
Casos de uso

Diagrama de casos de uso del ALUMNO.



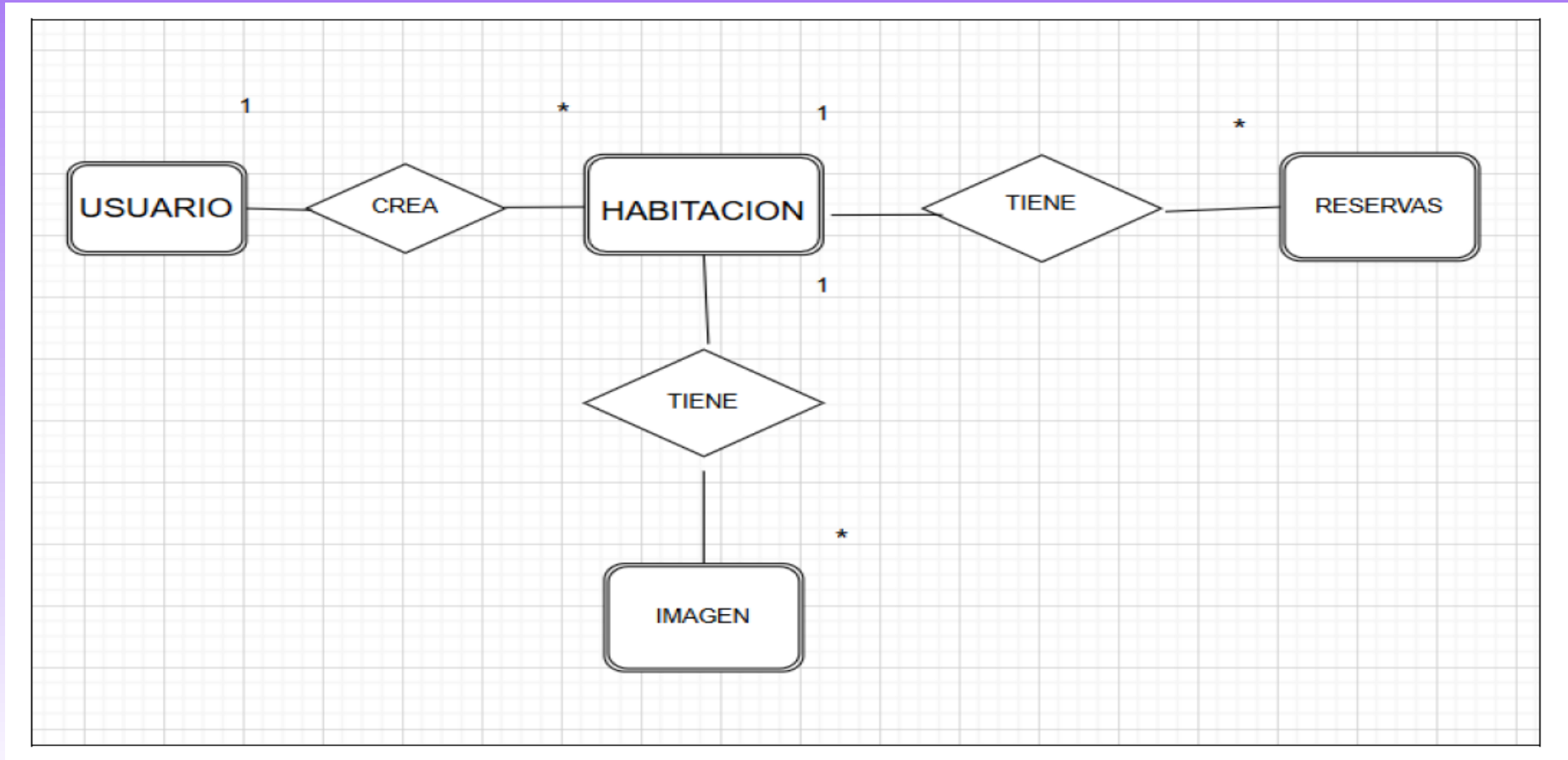
Modelo de datos

Diagrama de clases



Modelo de datos

Diagrama de entidad relación (E/R).



Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \dt
                List of tables
 Schema |          Name           | Type  | Owner
--------+-------------------------+-------+--------
 public | habitacion              | table | myuser
 public | habitacion_imagenes_url | table | myuser
 public | imagen                  | table | myuser
 public | reserva                 | table | myuser
 public | usuario                 | table | myuser
(5 rows)
```

Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \d usuario
                      Table "public.usuario"
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 id         | uuid                   |           | not null |
 contraseña | character varying(255) |           |          |
 email      | character varying(255) |           |          |
 nombre     | character varying(255) |           |          |
 rol        | character varying(255) |           |          |
Indexes:
    "usuario_pkey" PRIMARY KEY, btree (id)
Check constraints:
    "usuario_rol_check" CHECK (rol::text = ANY (ARRAY['ALUMNO'::character varying, 'PROPIETARIO'::character varying, 'ADMIN'::character varying]::
text[]))
Referenced by:
    TABLE "habitacion" CONSTRAINT "fk68vdor41uae3i4aq51j56cokn" FOREIGN KEY (propietario_id) REFERENCES usuario(id)
    TABLE "imagen" CONSTRAINT "fkd19pxifrmy9lnwi9s3hb83d6c" FOREIGN KEY (usuario_id) REFERENCES usuario(id)
    TABLE "reserva" CONSTRAINT "fkjar9wrcvgdxhp3o1pv7ge5x7d" FOREIGN KEY (propietario_id) REFERENCES usuario(id)
    TABLE "reserva" CONSTRAINT "fkpbab1y8d8vb54dk9wgx5w6pe6" FOREIGN KEY (alumno_id) REFERENCES usuario(id)
```

Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \d habitacion
                      Table "public.habitacion"
     Column     |          Type          | Collation | Nullable | Default
----------------+------------------------+-----------+----------+---------
 id             | uuid                   |           | not null |
 ciudad         | character varying(255) |           |          |
 descripcion    | character varying(255) |           |          |
 direccion      | character varying(255) |           |          |
 precio_mensual | numeric(38,2)          |           |          |
 titulo         | character varying(255) |           |          |
 propietario_id | uuid                   |           | not null |
Indexes:
    "habitacion_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fk68vdor41uae3i4aq51j56cokn" FOREIGN KEY (propietario_id) REFERENCES usuario(id)
Referenced by:
    TABLE "habitacion_imagenes_url" CONSTRAINT "fkk3hbofe0b80s7w5vyaqyr0boe" FOREIGN KEY (habitacion_id) REFERENCES habitacion(id)
    TABLE "reserva" CONSTRAINT "fktr5bg864m3dseko7gif2bl239" FOREIGN KEY (habitacion_id) REFERENCES habitacion(id)
```

Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \d reserva
                        Table "public.reserva"
     Column     |          Type          | Collation | Nullable | Default
----------------+------------------------+-----------+----------+---------
 id             | uuid                   |           | not null |
 estado_reserva | character varying(255) |           |          |
 fecha_fin      | date                   |           |          |
 fecha_inicio   | date                   |           |          |
 alumno_id      | uuid                   |           |          |
 habitacion_id  | uuid                   |           |          |
 propietario_id | uuid                   |           |          |
Indexes:
    "reserva_pkey" PRIMARY KEY, btree (id)
Check constraints:
    "reserva_estado_reserva_check" CHECK (estado_reserva::text = ANY (ARRAY['PENDIENTE'::character varying, 'CONFIRMADA'::character varying, 'CANC
ELADA'::character varying]::text[]))
Foreign-key constraints:
    "fkjar9wrcvgdxhp3o1pv7ge5x7d" FOREIGN KEY (propietario_id) REFERENCES usuario(id)
    "fkpbab1y8d8vb54dk9wgx5w6pe6" FOREIGN KEY (alumno_id) REFERENCES usuario(id)
    "fktr5bg864m3dseko7gif2bl239" FOREIGN KEY (habitacion_id) REFERENCES habitacion(id)
```

Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \d imagen
                      Table "public.imagen"
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 id         | uuid                   |           | not null |
 filename   | character varying(255) |           |          |
 url        | character varying(255) |           |          |
 usuario_id | uuid                   |           |          |
Indexes:
    "imagen_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fkd19pxifrmy9lnwi9s3hb83d6c" FOREIGN KEY (usuario_id) REFERENCES usuario(id)
```

Modelo de datos

Tablas, atributos, restricciones, claves foráneas y referencias

```
mydatabase=# \d habitacion_imagenes_url
                Table "public.habitacion_imagenes_url"
    Column     |          Type          | Collation | Nullable | Default
---------------+------------------------+-----------+----------+---------
 habitacion_id | uuid                   |           | not null |
 imagenes_url  | character varying(255) |           |          |
Foreign-key constraints:
    "fkk3hbofe0b80s7w5vyaqyr0boe" FOREIGN KEY (habitacion_id) REFERENCES habitacion(id)
```

Principales requisitos funcionales

- ✓ Proporcionar un sistema de autorización y autenticación para todos los usuarios con JWT.
- ✓ Gestionar las entidades (CRUD completo: crear, leer, actualizar, eliminar).
- ✓ Exposición de endpoints RESTful para consumo desde clientes móviles.
- ✓ Guardar las contraseñas encriptadas en la base de datos.
- ✓ Persistencia de datos en base de datos PostgreSQL con JPA/Hibernate.
- ✓ Proporcionar un sistema de búsqueda y filtros.
- ✓ Registro de actividad y errores mediante logs.
- ✓ Interfaz adaptativa y responsiva para distintos tamaños de pantalla.
- ✓ Manejo de errores y mensajes de estado (conexión, validación, etc.).

Principales requisitos no funcionales

- ✓ La aplicación debe cargarse de forma fluida bajo condiciones normales de uso.
- ✓ La aplicación debe ser capaz de manejar múltiples usuarios concurrentes sin degradar el rendimiento.
- ✓ Escalabilidad: capacidad para manejar aumento de usuarios y peticiones.
- ✓ Seguridad: protección contra ataques comunes (XSS, CSRF, inyecciones SQL).
- ✓ Alta disponibilidad con despliegue en contenedores Docker.
- ✓ Mantenibilidad: código modular, bien documentado y con buenas prácticas.
- ✓ La aplicación debe tener un diseño atractivo que facilite su usabilidad.
- ✓ La aplicación debe estar abierta a futuras mejoras y a su ampliación.
- ✓ Portabilidad: despliegue en distintos entornos (local, cloud, CI/CD).
- ✓ Trazabilidad: logs estructurados y trazas para auditoría.
- ✓ Compatibilidad: soporte para múltiples versiones de Android.
- ✓ Seguridad: almacenamiento seguro de credenciales y tokens.

Diseño, estructura y arquitectura

- Android App UI adaptativa, gestión de sesión JWT.
- Spring Boot API Endpoints REST, seguridad, validación, logs.
- PostgreSQL en Docker Base de datos persistente, aislada en contenedor.
- Docker Compose Orquestación de API + DB.

Diseño, estructura y arquitectura

Arquitectura de red

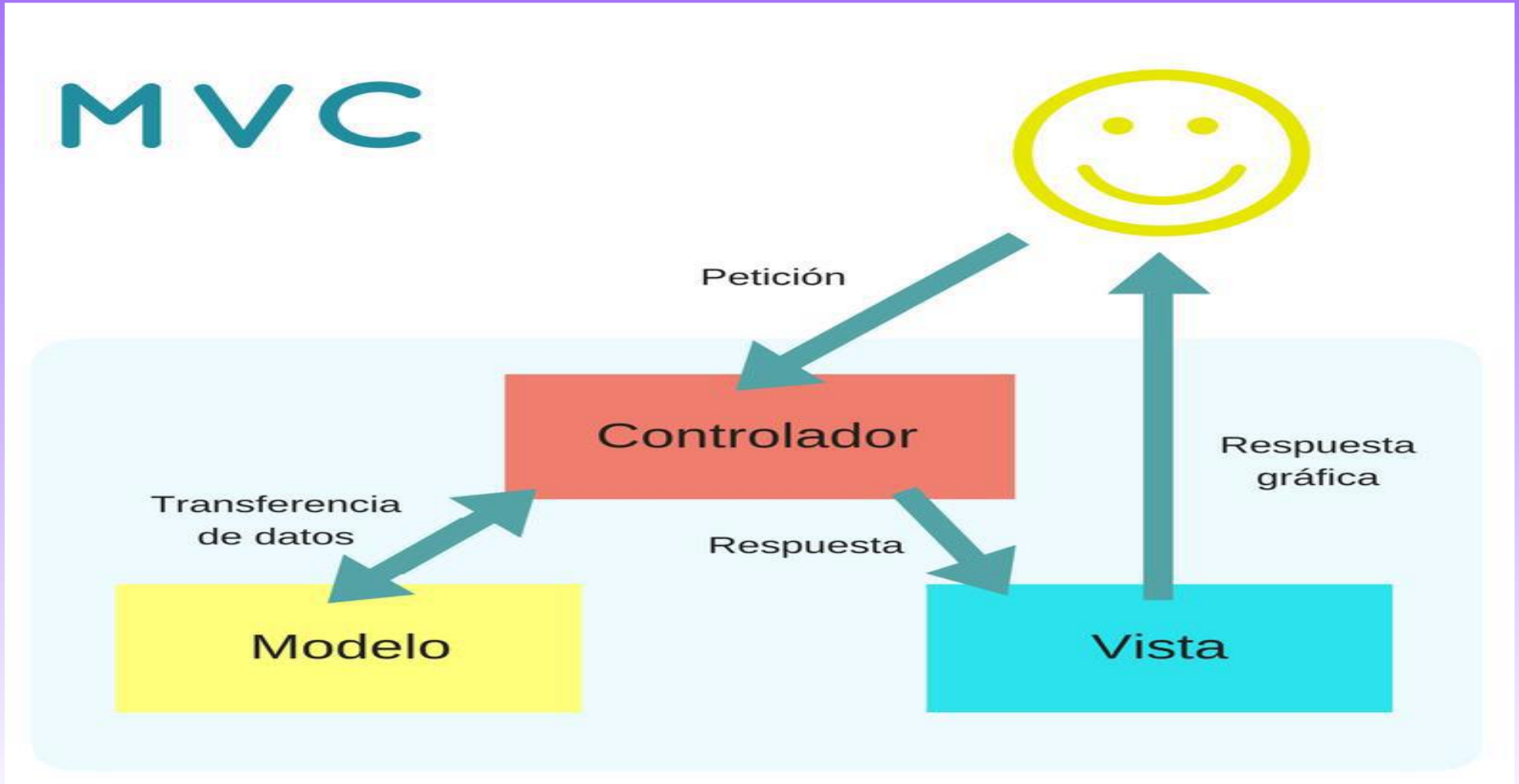
- **Cliente móvil (Android)** → envía peticiones HTTP/HTTPS (JSON).
- **Servidor backend (Spring Boot)** → recibe, procesa y responde a las solicitudes.
- **Base de datos (PostgreSQL en Docker)** → almacena y gestiona la información.
- Comunicación segura mediante **HTTP/HTTPS + JWT** para autenticación.

Diseño, estructura y arquitectura

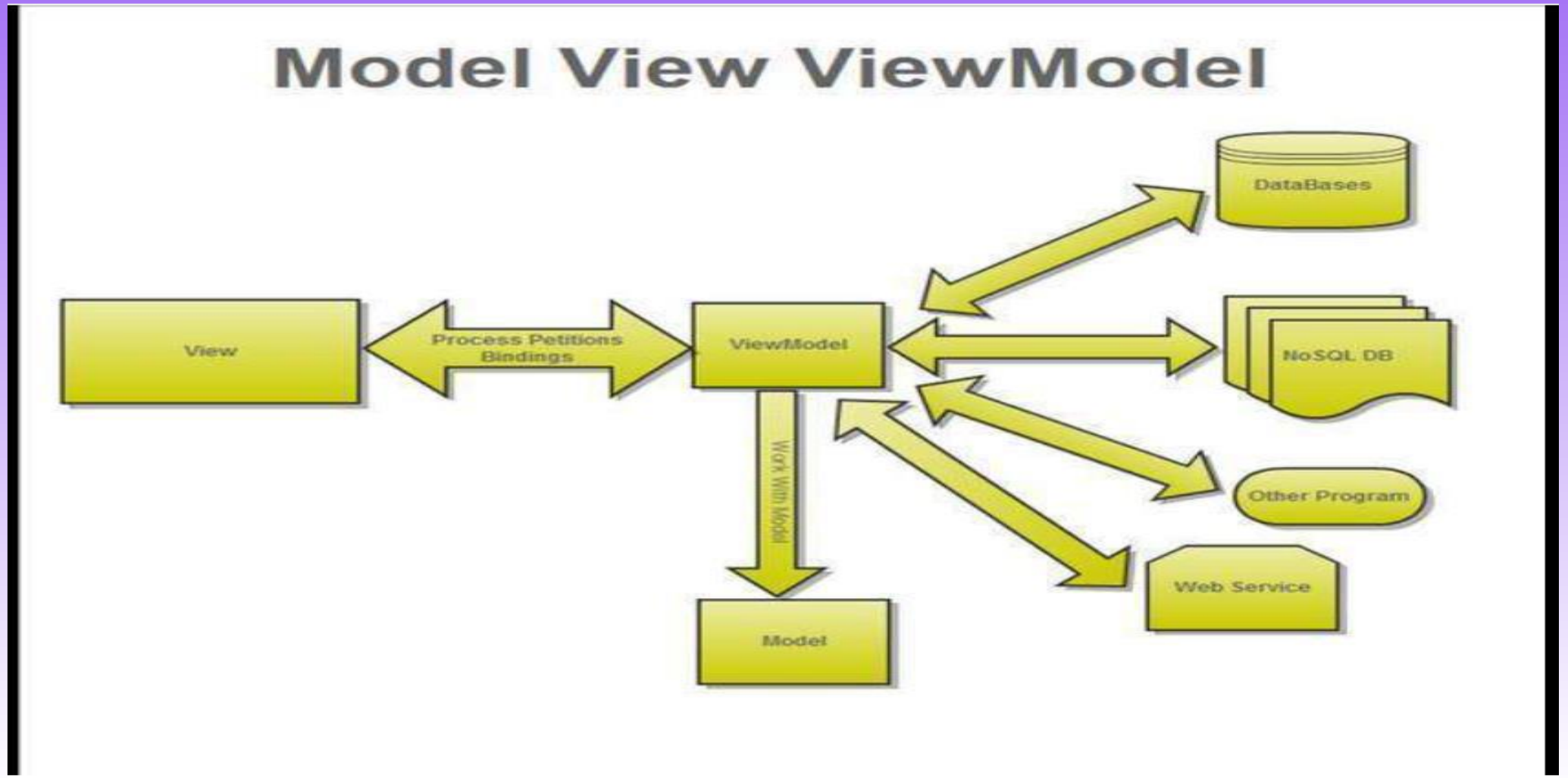
Componentes del sistema

- **Backend (Spring Boot)**
- **Frontend (Android)**

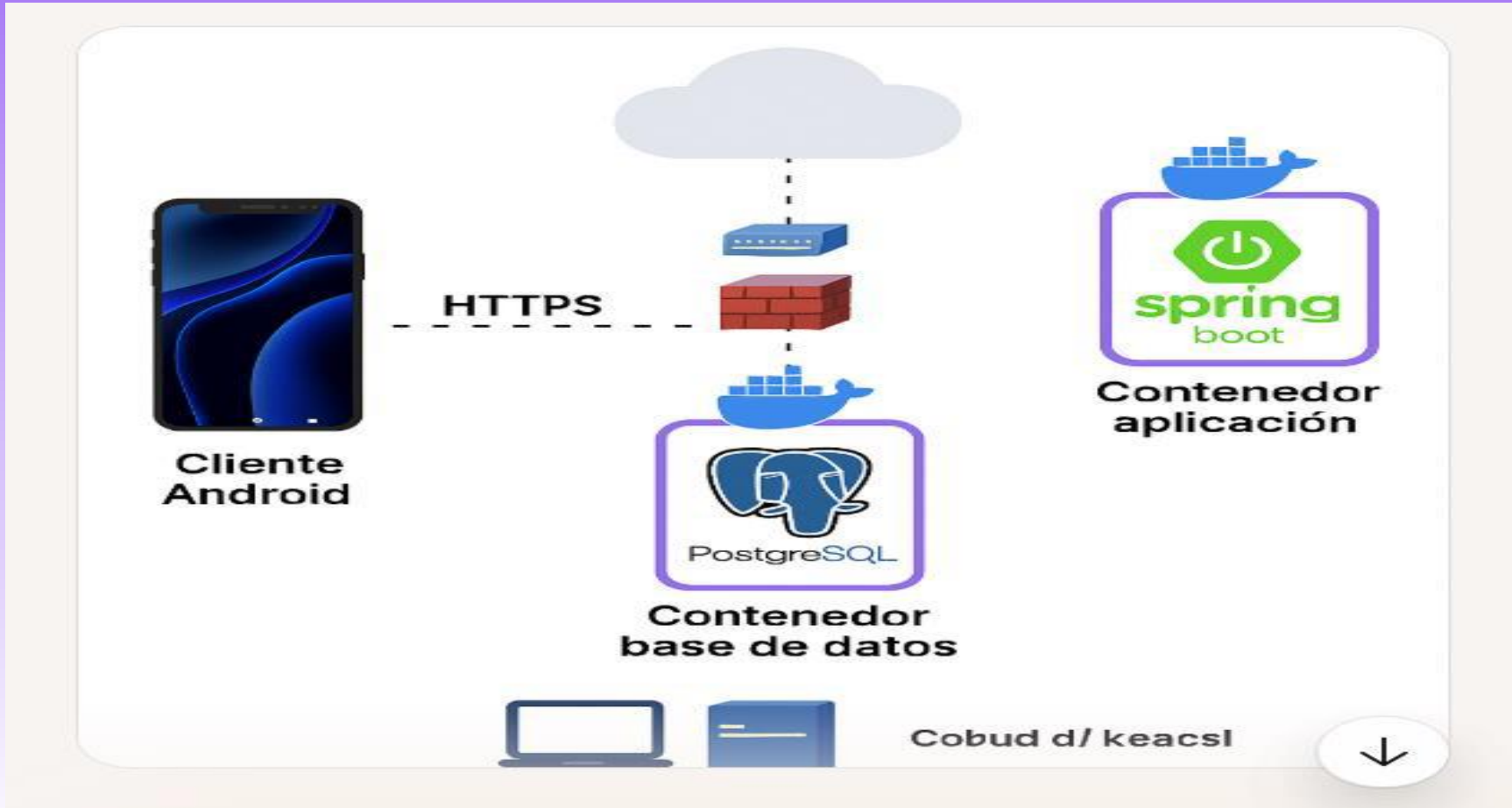
Diseño, estructura y arquitectura



Diseño, estructura y arquitectura



Diseño, estructura y arquitectura



Herramientas y teconologías utilizadas

Categoría	Tecnología / Herramienta	Descripción / Uso
Plataformas	Android OS	Sistema operativo móvil para la app cliente
	Docker	Contenedores para backend y base de datos
	Docker Compose	Orquestación de servicios (API + DB)
	PostgreSQL	Motor de base de datos relacional
	GitHub / GitLab	Control de versiones y colaboración
Lenguajes	Java	Backend con Spring Boot
	Kotlin	Desarrollo de la app Android
	SQL	Consultas en PostgreSQL
	YAML	Configuración de Spring Boot y Docker Compose
	JSON	Intercambio de datos entre cliente y servidor
Servidores	Apache Tomcat (embebido)	Servidor web integrado en Spring Boot
	Docker host / Cloud VM	Despliegue local o en producción

Herramientas y teconologías utilizadas

Categoría	Tecnología / Herramienta	Descripción / Uso
Frameworks Backend	Spring Boot	Framework principal para la API REST
	Spring Data JPA	Acceso a datos con PostgreSQL
	Spring Security	Autenticación y autorización con JWT
	Hibernate	ORM para mapear entidades Java a tablas SQL
Frameworks Android	Android Jetpack	Arquitectura MVVM y navegación
	Retrofit + OkHttp	Consumo de API REST desde Android
	Gson / Moshi	Serialización/deserialización de datos JSON
	Material Design Components	UI moderna y responsiva
	Room / DataStore	Persistencia local y sincronización offline

Herramientas y teconologías utilizadas

Categoría	Tecnología / Herramienta	Descripción / Uso
Entornos de desarrollo	Android Studio	IDE oficial para desarrollo Android
	IntelliJ IDEA / VS Code	IDE para desarrollo backend con Spring Boot
	Postman / Insomnia	Pruebas y depuración de endpoints REST
	Docker CLI / Docker Desktop	Gestión de contenedores y redes
	Git + GitHub/GitLab CLI	Control de versiones y trabajo colaborativo

Implementación

Enlaces a los repositorios con el código de la aplicación:

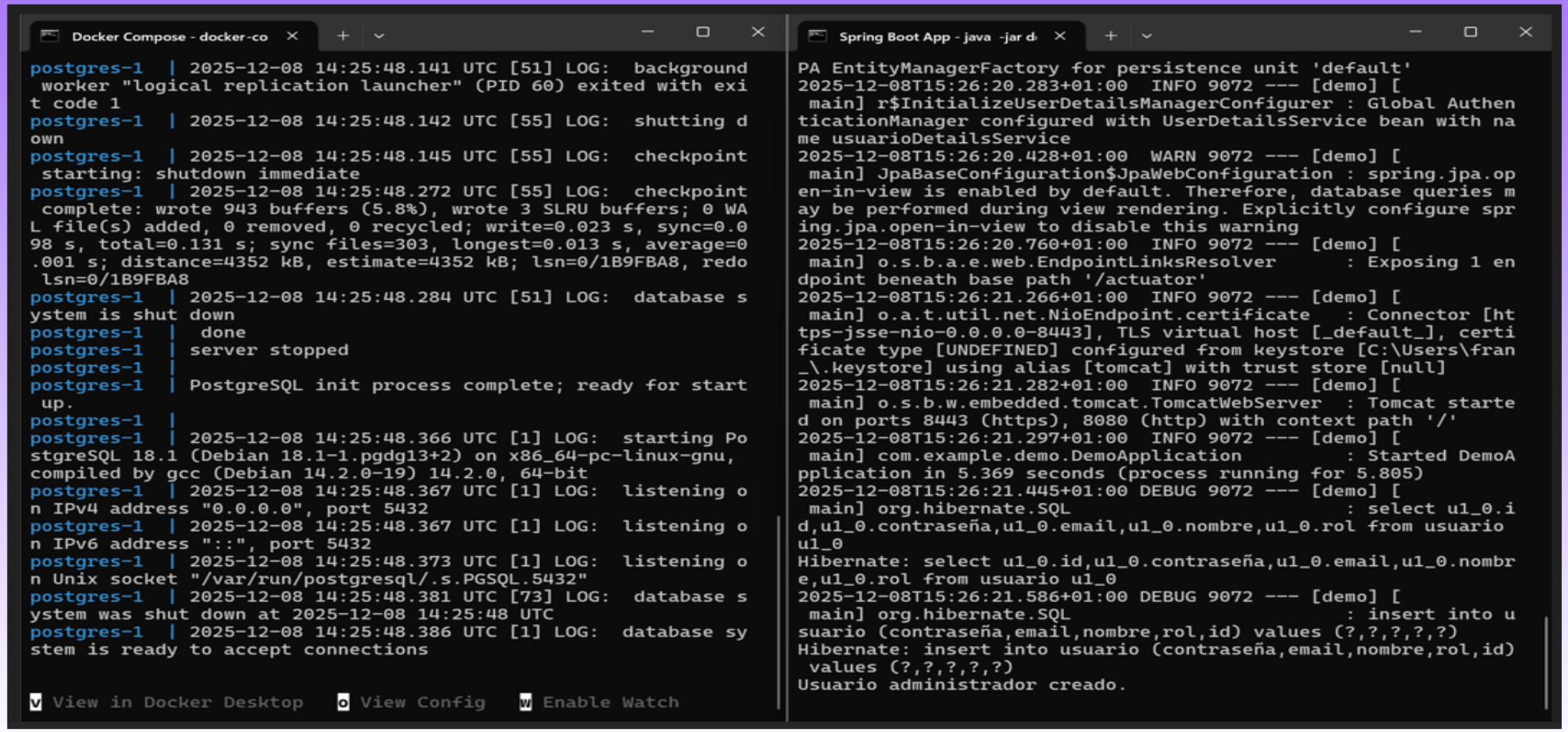
API SPRING

https://github.com/frantoribio/demo

APP ANDROID

https://github.com/frantoribio/alquilerapp-compose

Configuraciones realizadas en el sistema



Puebas

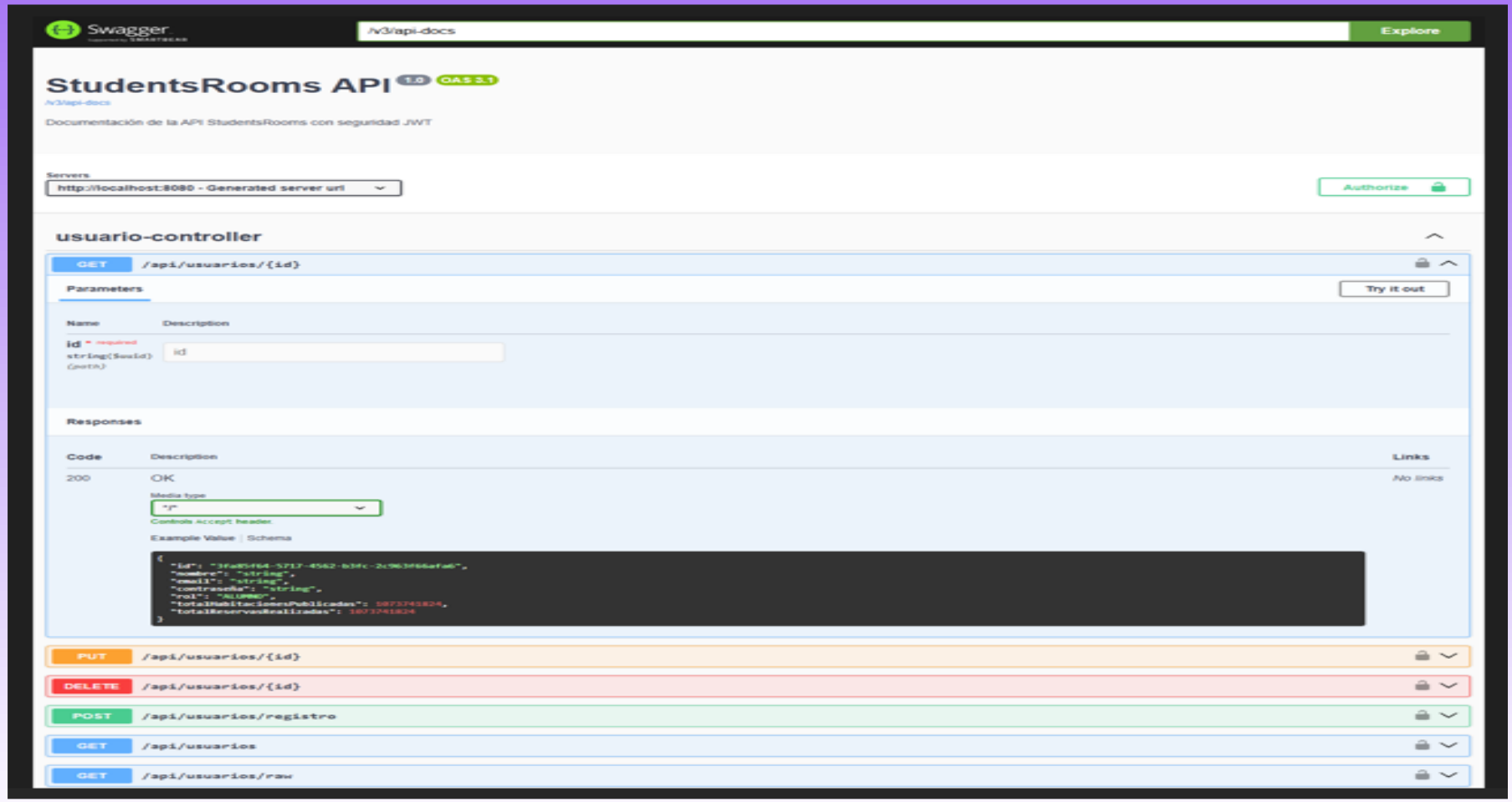
```
C:\Users\fran_\Desktop\Entrega_Francisco_Toribio\App>curl -X POST http://localhost:8080/auth/login -H "Content-Type: app
lication/json" -d "{\"email\":\"a@a.com\",\"contraseña\":\"a\"}" | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   307    0   270  100    37   3096    424 --:--:-- --:--:-- --:--:--  3569
{
  "email": "a@a.com",
  "token": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJhQGEuY29tIiwicm9sIjoiUk9MRV9BTFVNTk8iLCJpYXQiOjE3NjUyMDQ2MjYsImV4cCI6MTc2NTI
5MTAyNn0.wlAffo8mtaeKhjFDitgJQYLhCkib7uMMaPtEf9uiLjwJuQUMoemaoFiZir8TfvW1",
  "rol": "ALUMNO",
  "id": "922986b7-8677-488f-b48e-3f4a0a5bf2e2"
}
```

Puebas

src > test > java > com > example > demo > service > HabitacionServiceTest.java > ...

```java
package com.example.demo.service;

import com.example.demo.dto.CrearHabitacionDto;
import com.example.demo.model.Habitacion;
import com.example.demo.model.Usuario;
import com.example.demo.repository.HabitacionRepository;
import com.example.demo.repository.UsuarioRepository;
import org.junit.jupiter.api.Test;
import java.math.BigDecimal;
import java.util.Optional;
import java.util.UUID;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

class HabitacionServiceTest {

    @Test
    void testCrearHabitacionAsignaPropietario() {
        // Arrange
        HabitacionRepository habitacionRepository = mock(HabitacionRepository.class);
        UsuarioRepository usuarioRepository = mock(UsuarioRepository.class);

        HabitacionService service = new HabitacionService(habitacionRepository, usuarioRepository)

        UUID propietarioId = UUID.randomUUID();
        Usuario propietario = new Usuario();
        propietario.setId(propietarioId);
        propietario.setEmail(email: "test@example.com");

        when(usuarioRepository.findById(propietarioId)).thenReturn(Optional.of(propietario));

```

Puebas


Swagger
/v3/api-docs
Explore
StudentsRooms API
1.0
OAS 3.1
/v3/api-docs
Documentación de la API StudentsRooms con seguridad JWT
Servers
http://localhost:8080 - Generated server url
Authorize
usuario-controller
GET /api/usuarios/{id}
Parameters
Try it out
Name
Description
id
required
string($uuid)
(path)
id
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value | Schema
PUT /api/usuarios/{id}
DELETE /api/usuarios/{id}
POST /api/usuarios/registro
GET /api/usuarios
GET /api/usuarios/raw

Manual de usuario

Instalación y uso

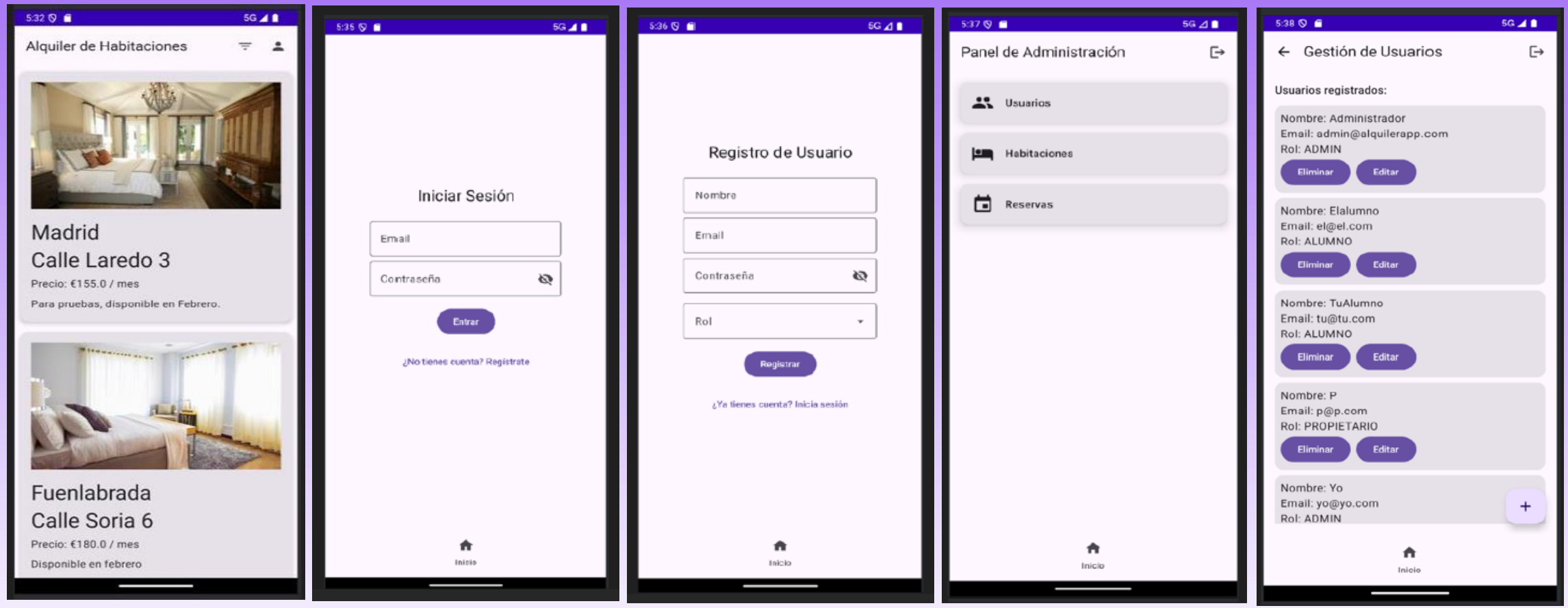
El proyecto se entrega en una carpeta comprimida que contiene:

- Las instrucciones de instalación.
- La memoria.
- Una carpeta con los archivos necesarios:

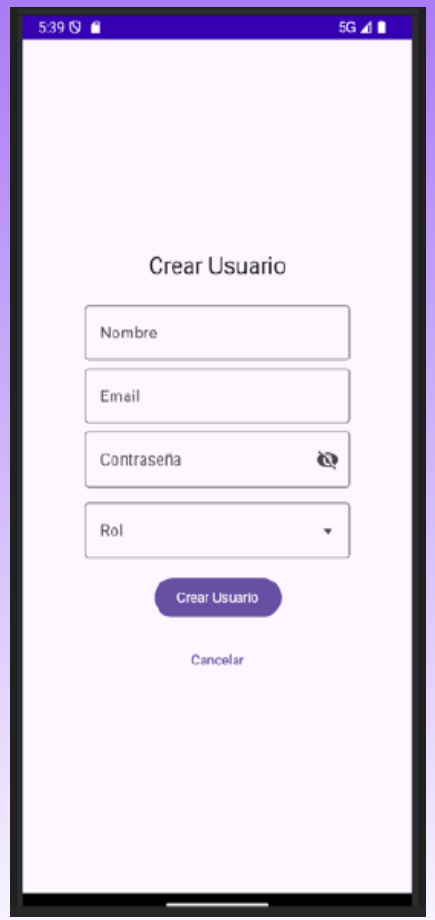
  .apk, application.properties, docker-compose.yaml,

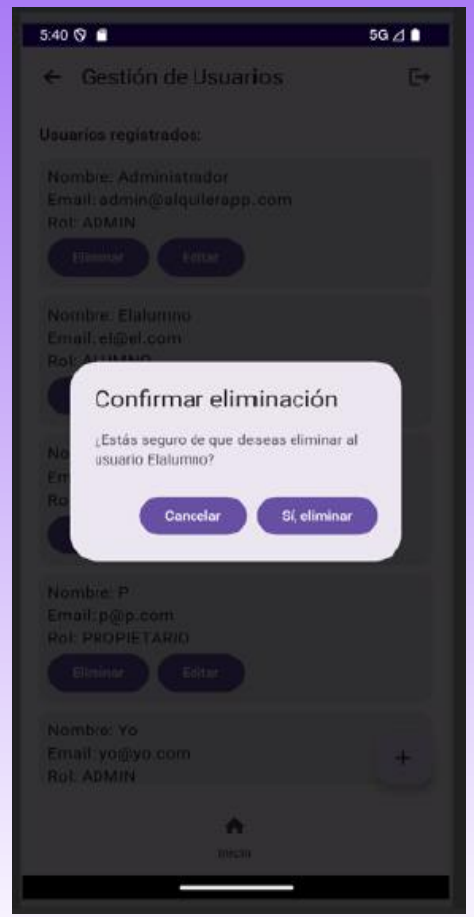
  .jar y .bat

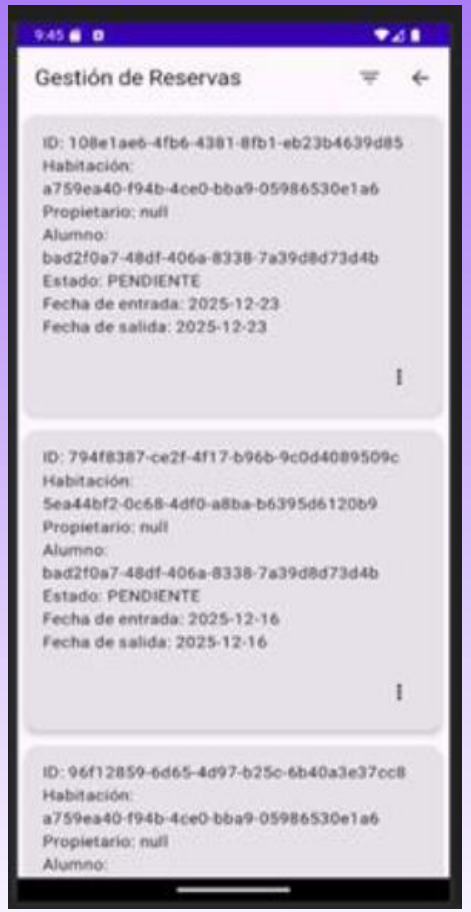
Instrucciones de uso

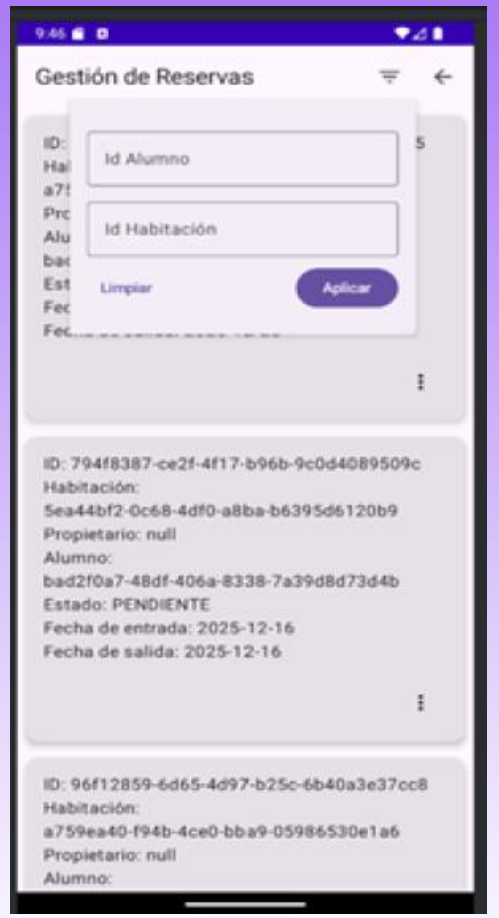


Instrucciones de uso

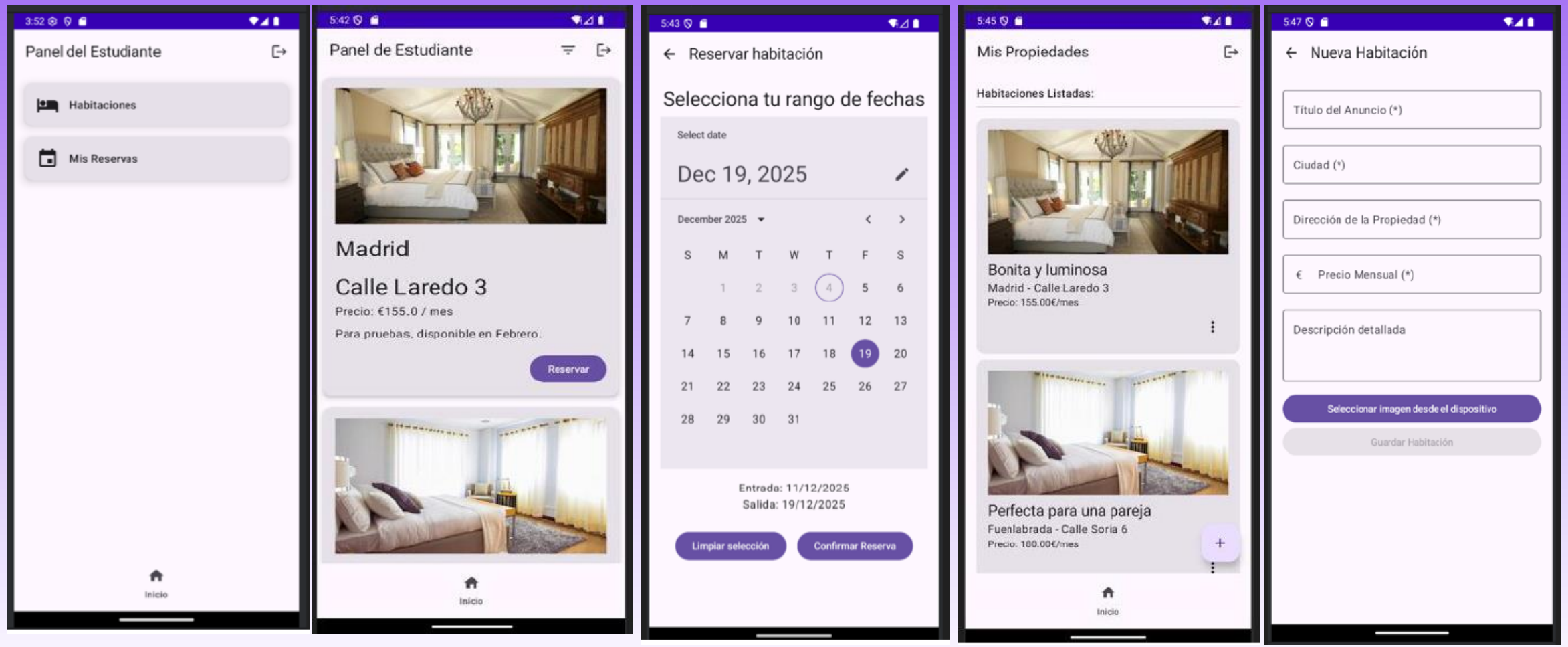






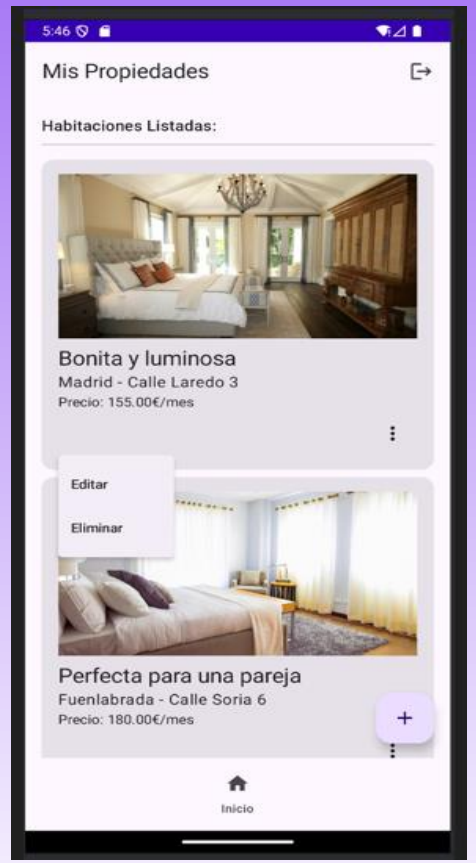


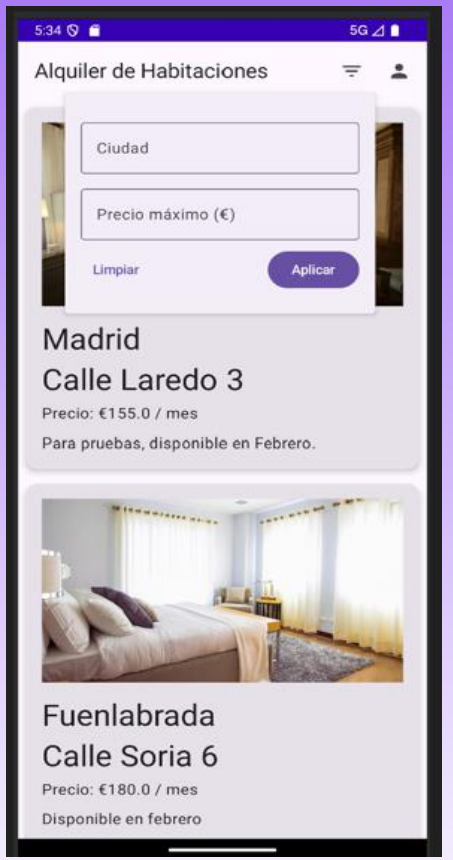
Instrucciones de uso



IES Rey Fernado VI - Francisco Toribio Respaldo

Instrucciones de uso





DEMOSTRACIÓN

Conclusiones

Integración exitosa entre cliente y servidor

Arquitectura modular y escalable

Persistencia y gestión de datos eficiente

Seguridad y control de acceso

Interfaz intuitiva y experiencia de usuario

Pruebas y validación del sistema

Aprendizaje y aplicación de buenas prácticas

Posibilidades de mejora y ampliación

Legislación

Enseñanzas mínimas**: Real Decreto 450/2010, de 16 de abril (BOE 20/05/2010)**

https://www.boe.es/boe/dias/2010/05/20/pdfs/BOE-A-2010-8067.pdf

Currículo: **D. 3/2011, de 13 de enero (BOCM 31/01/2011)**

https://www.comunidad.madrid/sites/default/files/doc/educacion/fp/FP-Ensenanza-IFCS02-LOE-Curriculo-D20110003.pdf

Bibliografía

https://spring.io/projects/spring-boot

https://developer.android.com/compose

https://www.postgresql.org/docs/

https://docs.docker.com/

https://www.jwt.io/introduction

https://swagger.io/docs/

Agradecimientos

Al sistema de educación pública