



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.

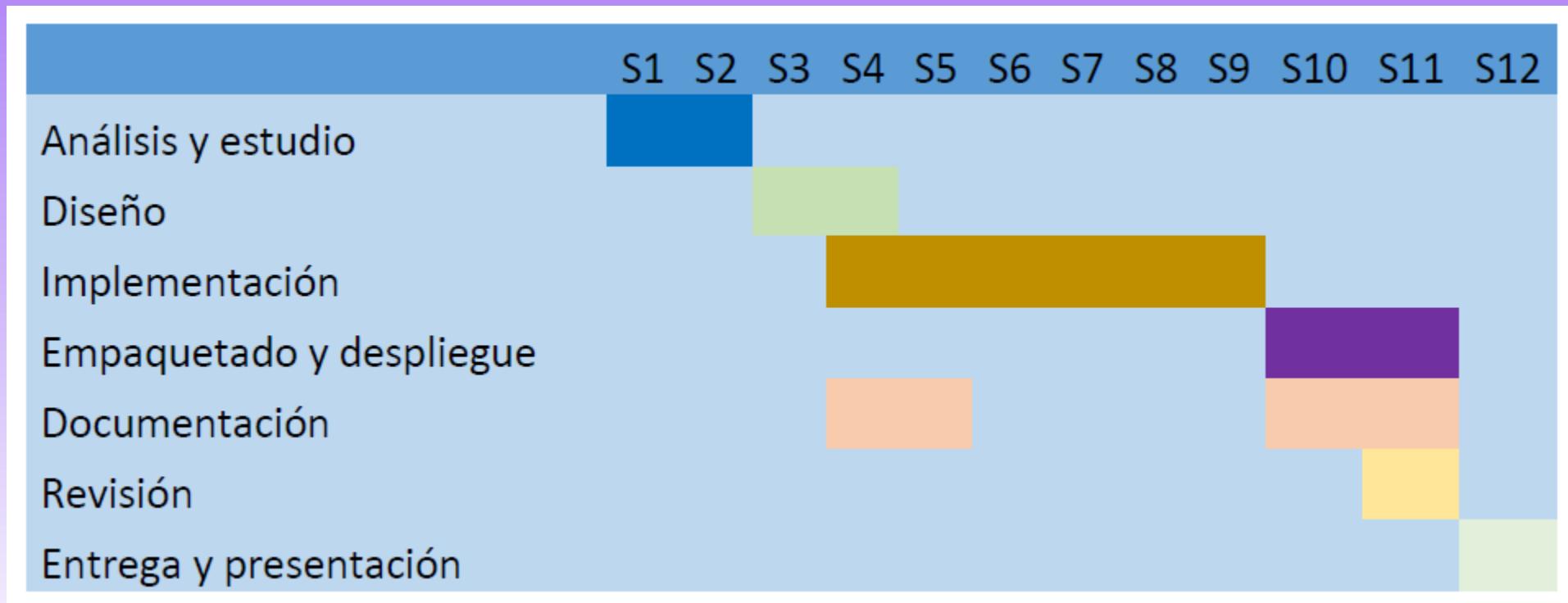
Propósito personal

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

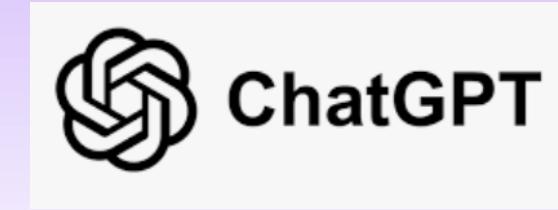
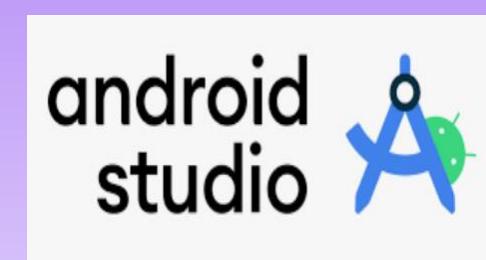
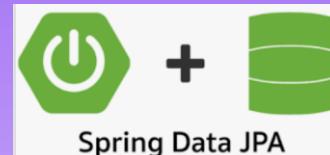
Resumen de requisitos

- 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

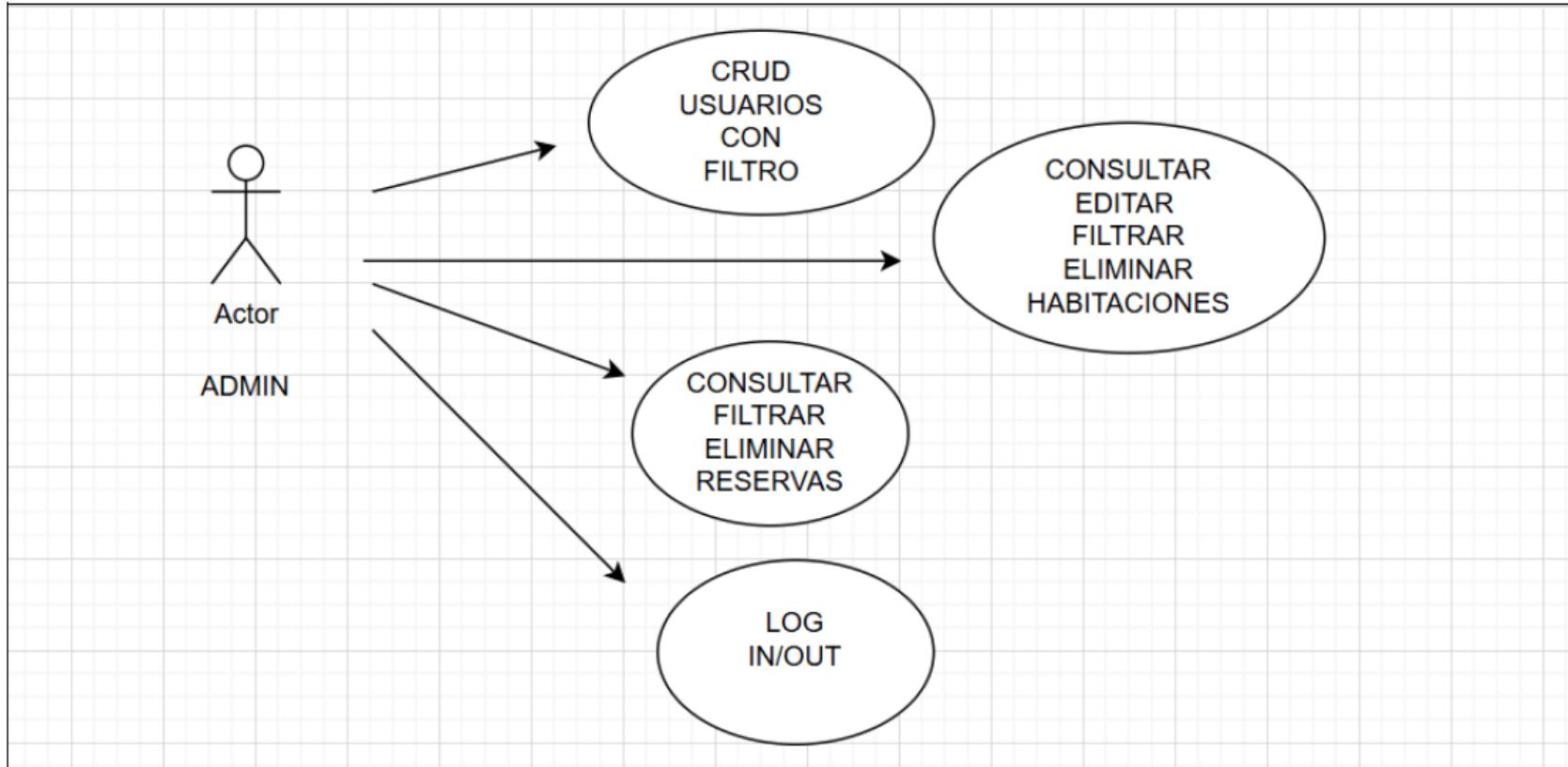


Análisis

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

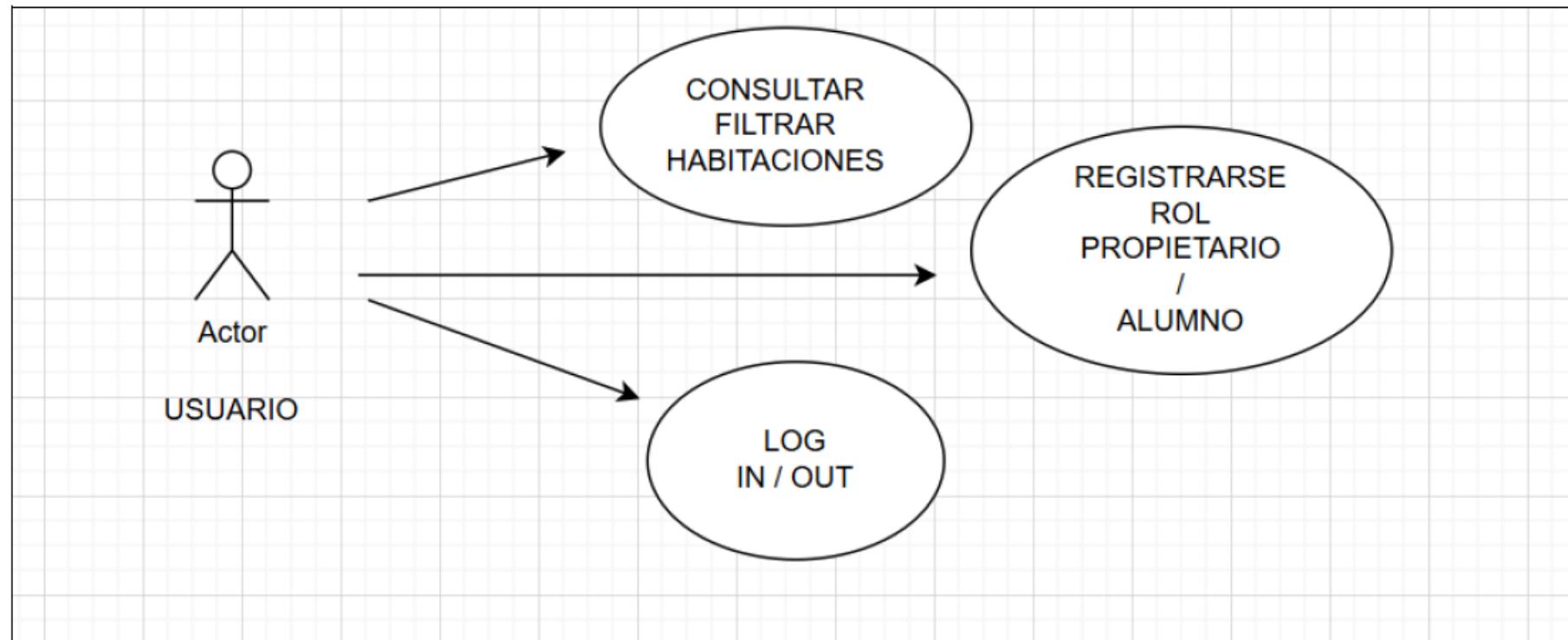
Casos de uso

Diagrama de casos de uso del administrador (ADMIN).



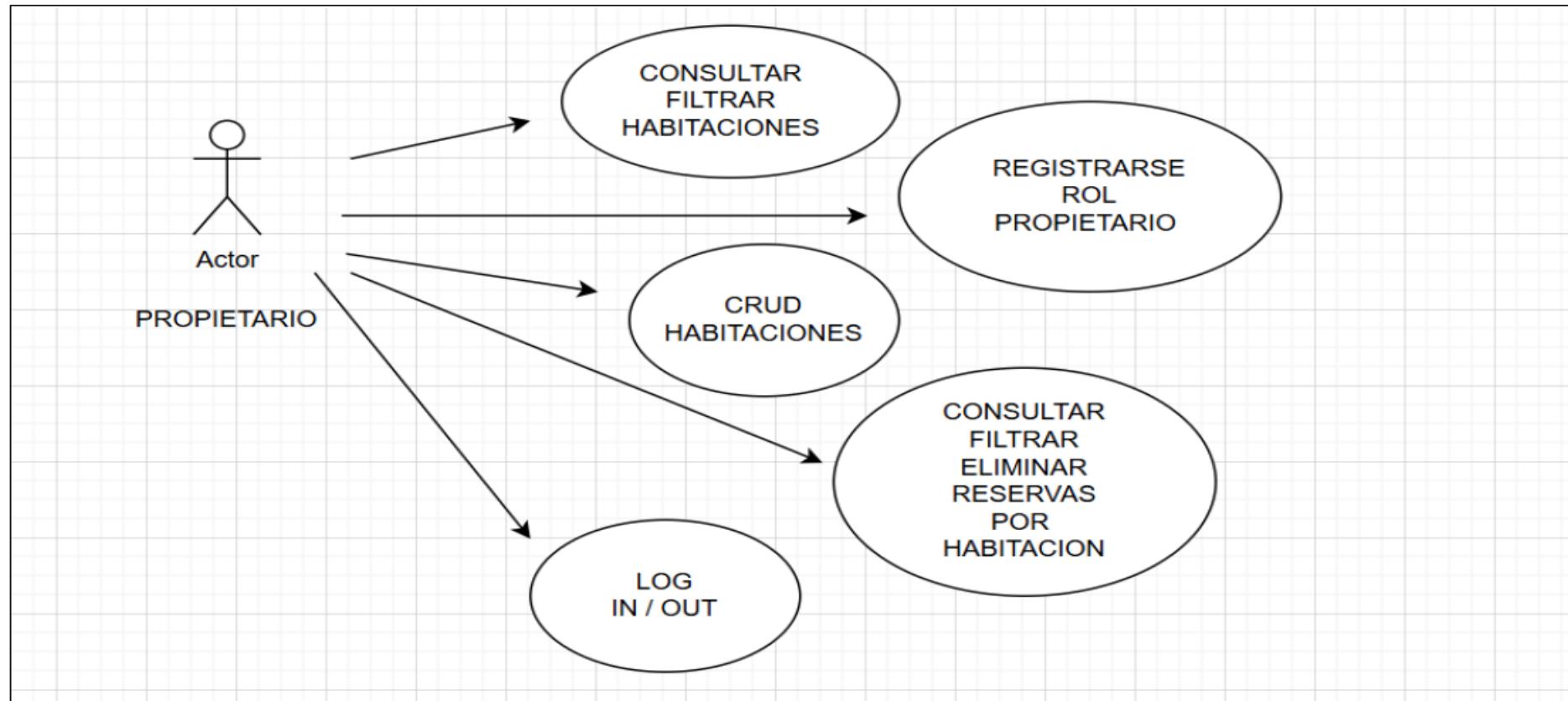
Casos de uso

Diagrama de casos de uso del USUARIO.



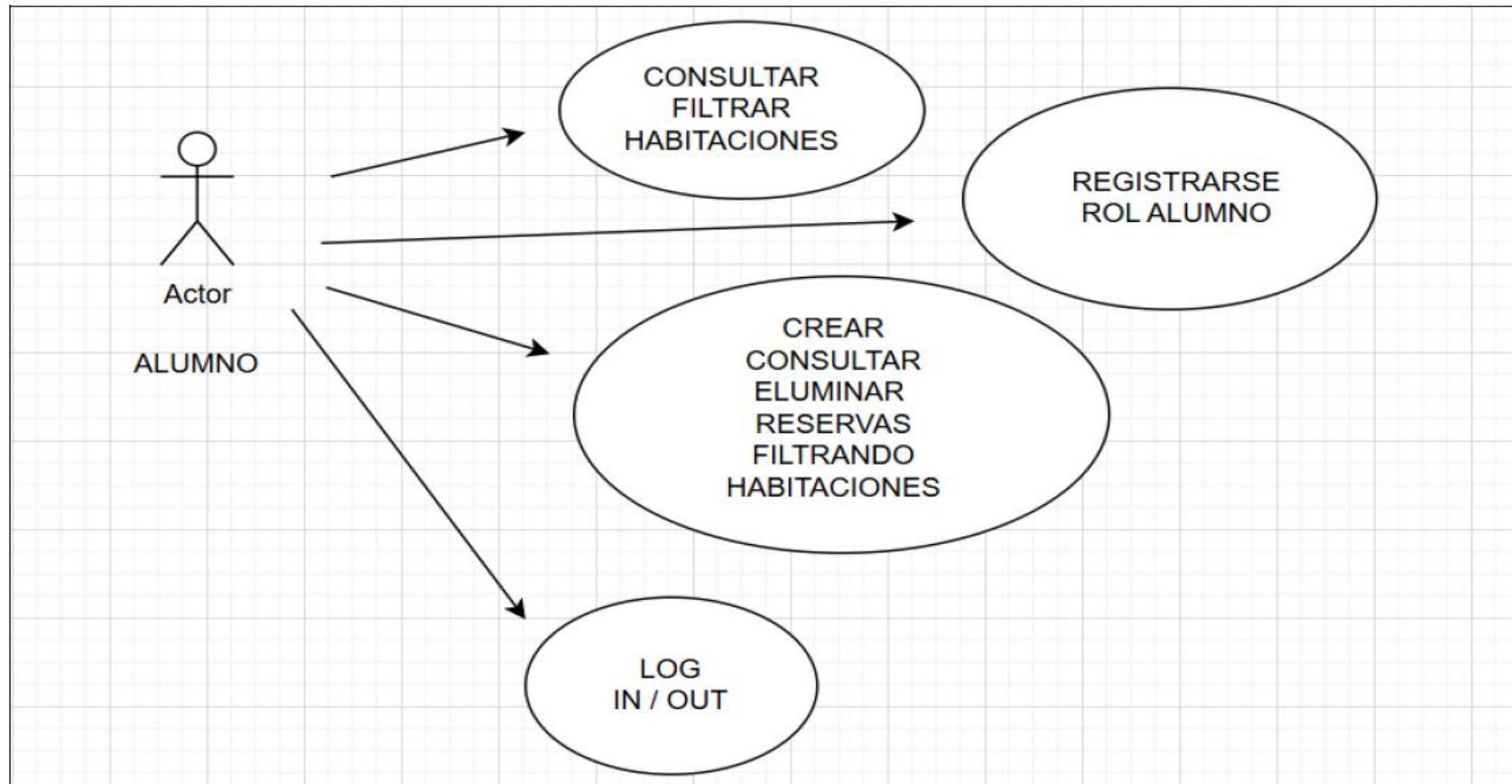
Casos de uso

Diagrama de casos de uso del PROPIETARIO.



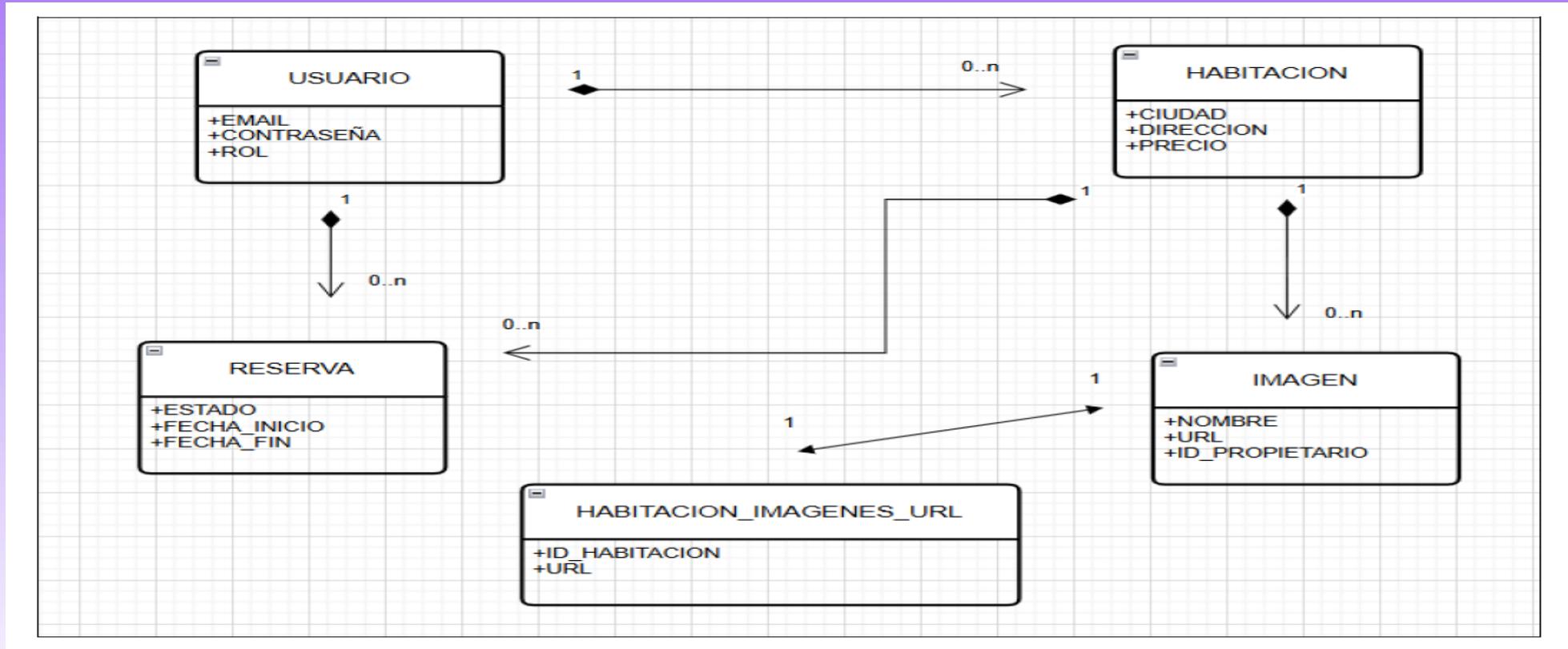
Casos de uso

Diagrama de casos de uso del ALUMNO.



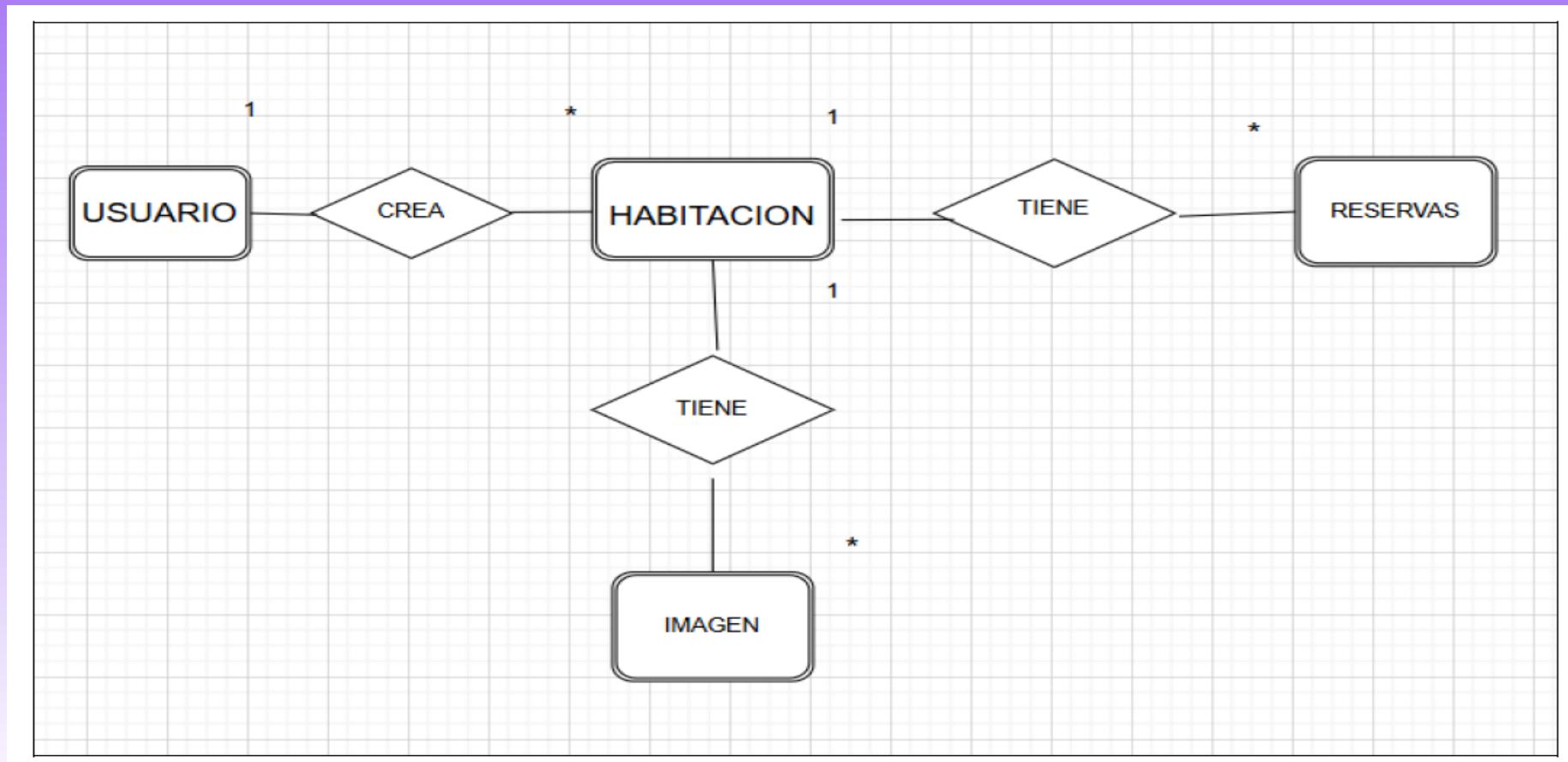
Modelo de datos

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 "fkfd19pxifrm9lnwi9s3hb83d6c" 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, 'CANCELADA'::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:
  "fkfd19pxifrmmy9lnwi9s3hb83d6c" 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 responsive 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

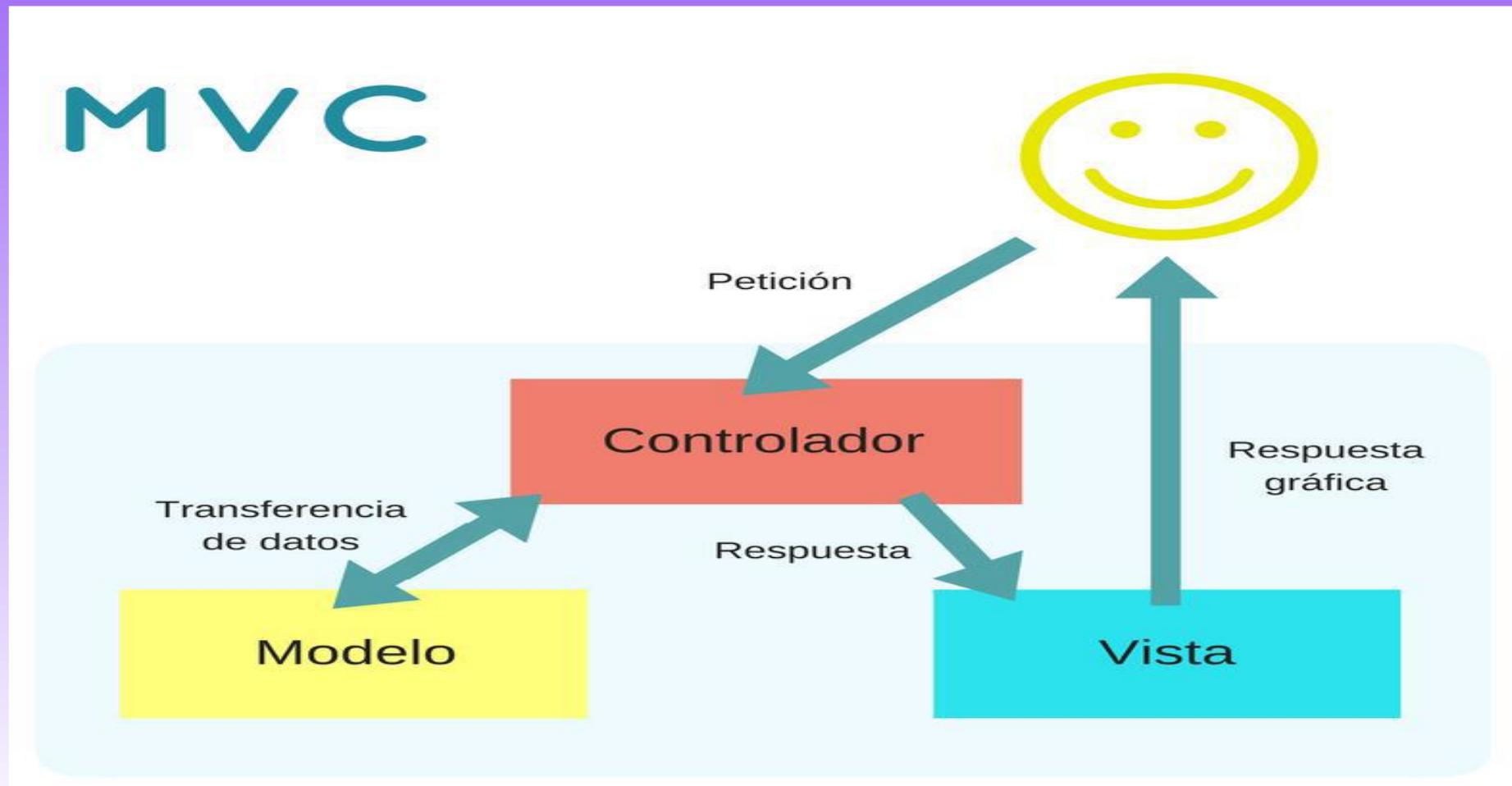
- **Cliente móvil (Android)** → envía peticiones HTTP/HTTPS (JSON).
- **Servidor backend (Spring Boot)** → recibe, procesa y responde a las solicitudes.
- **Base de datos (PostgreSQL en Docker)** → almacena y gestiona la información.
- Comunicación segura mediante **HTTP/HTTPS + JWT** para autenticación.

Diseño, estructura y arquitectura

Componentes del sistema

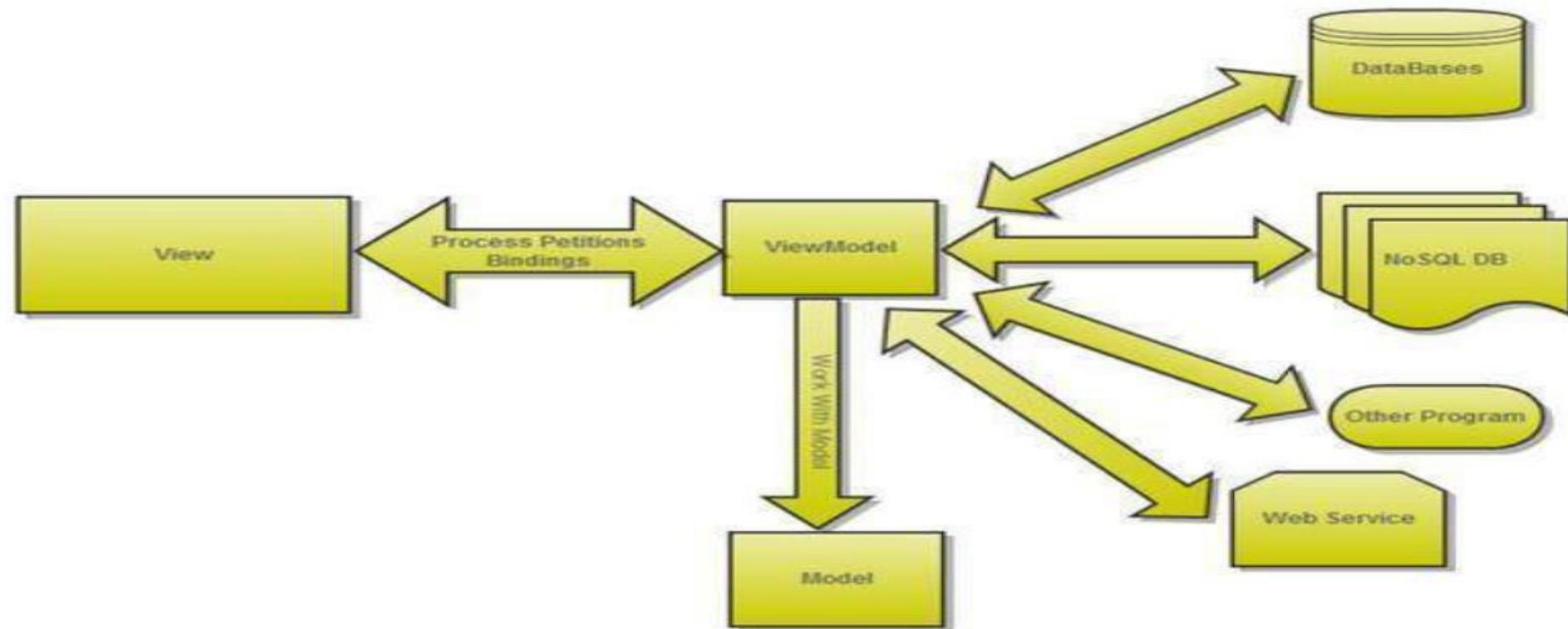
- **Backend (Spring Boot)**
- **Frontend (Android)**

Diseño, estructura y arquitectura

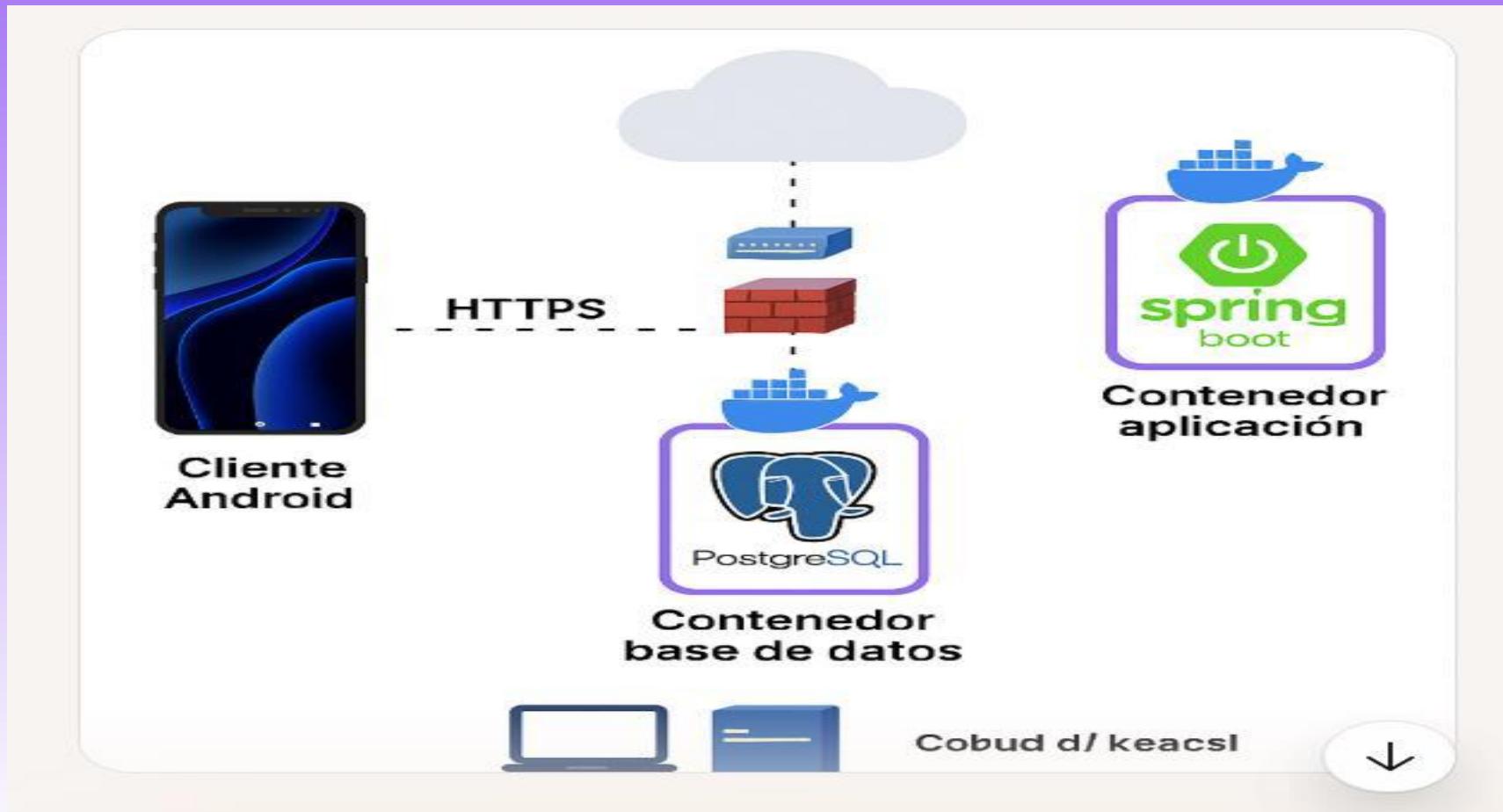


Diseño, estructura y arquitectura

Model View ViewModel



Diseño, estructura y arquitectura



Herramientas y tecnologías 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
Servidores	JSON	Intercambio de datos entre cliente y servidor
	Apache Tomcat (embebido)	Servidor web integrado en Spring Boot
	Docker host / Cloud VM	Despliegue local o en producción

Herramientas y tecnologías 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 responsive
	Room / DataStore	Persistencia local y sincronización offline

Herramientas y tecnologías 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/alquierapp-compose>

Configuraciones realizadas en el sistema

The image shows two terminal windows side-by-side. The left window is titled "Docker Compose - docker-co" and displays logs for a PostgreSQL container. The right window is titled "Spring Boot App - java -jar d" and displays logs for a Spring Boot application.

Docker Compose - docker-co

```
postgres-1 | 2025-12-08 14:25:48.141 UTC [51] LOG: background worker "logical replication launcher" (PID 60) exited with exit code 1
postgres-1 | 2025-12-08 14:25:48.142 UTC [55] LOG: shutting down
postgres-1 | 2025-12-08 14:25:48.145 UTC [55] LOG: checkpoint starting: shutdown immediate
postgres-1 | 2025-12-08 14:25:48.272 UTC [55] LOG: checkpoint complete: wrote 943 buffers (5.8%), wrote 3 SLRU buffers; 0 WAL file(s) added, 0 removed, 0 recycled; write=0.023 s, sync=0.098 s, total=0.131 s; sync files=303, longest=0.013 s, average=0.001 s; distance=4352 kB, estimate=4352 kB; lsn=0/1B9FBA8, redo lsn=0/1B9FBA8
postgres-1 | 2025-12-08 14:25:48.284 UTC [51] LOG: database system is shut down
postgres-1 | done
postgres-1 | server stopped
postgres-1 | PostgreSQL init process complete; ready for startup.
postgres-1 | 2025-12-08 14:25:48.366 UTC [1] LOG: starting PostgreSQL 18.1 (Debian 18.1-1.pgdg13+2) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
postgres-1 | 2025-12-08 14:25:48.367 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
postgres-1 | 2025-12-08 14:25:48.367 UTC [1] LOG: listening on IPv6 address ":::", port 5432
postgres-1 | 2025-12-08 14:25:48.373 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres-1 | 2025-12-08 14:25:48.381 UTC [73] LOG: database system was shut down at 2025-12-08 14:25:48 UTC
postgres-1 | 2025-12-08 14:25:48.386 UTC [1] LOG: database system is ready to accept connections
```

View in Docker Desktop View Config Enable Watch

Spring Boot App - java -jar d

```
PA EntityManagerFactory for persistence unit 'default'
2025-12-08T15:26:20.283+01:00  INFO 9072 --- [demo] [main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with UserDetailsService bean with name userDetailsService
2025-12-08T15:26:20.428+01:00  WARN 9072 --- [demo] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-12-08T15:26:20.760+01:00  INFO 9072 --- [demo] [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint beneath base path '/actuator'
2025-12-08T15:26:21.266+01:00  INFO 9072 --- [demo] [main] o.a.t.util.net.NioEndpoint.certificate : Connector [https-jsse-nio-0.0.0-8443], TLS virtual host [_default_], certificate type [UNDEFINED] configured from keystore [C:\Users\fran_\keystore] using alias [tomcat] with trust store [null]
2025-12-08T15:26:21.282+01:00  INFO 9072 --- [demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on ports 8443 (https), 8080 (http) with context path '/'
2025-12-08T15:26:21.297+01:00  INFO 9072 --- [demo] [main] com.example.demo.DemoApplication : Started DemoApplication in 5.369 seconds (process running for 5.805)
2025-12-08T15:26:21.445+01:00  DEBUG 9072 --- [demo] [main] org.hibernate.SQL : select u1_0.id,u1_0.contraseña,u1_0.email,u1_0.nombre,u1_0.rol from usuario u1_0
Hibernate: select u1_0.id,u1_0.contraseña,u1_0.email,u1_0.nombre,u1_0.rol from usuario u1_0
2025-12-08T15:26:21.586+01:00  DEBUG 9072 --- [demo] [main] org.hibernate.SQL : insert into usuario (contraseña,email,nombre,rol,id) values (?, ?, ?, ?, ?)
Hibernate: insert into usuario (contraseña,email,nombre,rol,id) values (?, ?, ?, ?, ?)
Usuario administrador creado.
```

Puebas

```
C:\Users\fran_\Desktop\Entrega_Francisco_Toribio\App>curl -X POST http://localhost:8080/auth/login -H "Content-Type: application/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": "eyJhbGciOiJIUzI1NjM4NCJ9.eyJzdWIiOiJhQGEuY29tIiwicm9sIjoiUk9MRV9BTFVNTk8iLCJpYXQiOjE3NjUyMDQ2MjYsImV4cCI6MTc2NTI5MTAyNn0.wlAffo8mtaeKhjFDitgJQYLhCkib7uMMaPtEf9uiLjwJuQUMoemaoFiZir8TfvW1",
  "rol": "ALUMNO",
  "id": "922986b7-8677-488f-b48e-3f4a0a5bf2e2"
}
```

Puebas

```
src > test > java > com > example > demo > service > HabitacionServiceTest.java > ...
1 package com.example.demo.service;
2
3 import com.example.demo.dto.CrearHabitacionDto;
4 import com.example.demo.model.Habitacion;
5 import com.example.demo.model.Usuario;
6 import com.example.demo.repository.HabitacionRepository;
7 import com.example.demo.repository.UsuarioRepository;
8 import org.junit.jupiter.api.Test;
9 import java.math.BigDecimal;
10 import java.util.Optional;
11 import java.util.UUID;
12 import static org.junit.jupiter.api.Assertions.*;
13 import static org.mockito.Mockito.*;
14
15 class HabitacionServiceTest {
16
17     @Test
18     void testCrearHabitacionAsignaPropietario() {
19         // Arrange
20         HabitacionRepository habitacionRepository = mock(HabitacionRepository.class);
21         UsuarioRepository usuarioRepository = mock(UsuarioRepository.class);
22
23         HabitacionService service = new HabitacionService(habitacionRepository, usuarioRepository)
24
25         UUID propietarioId = UUID.randomUUID();
26         Usuario propietario = new Usuario();
27         propietario.setId(propietarioId);
28         propietario.setEmail(email: "test@example.com");
29
30         when(usuarioRepository.findById(propietarioId)).thenReturn(Optional.of(propietario));
31     }
}
```

Puebas

The screenshot shows the Swagger UI interface for the **StudentsRooms API**. The top navigation bar includes the Swagger logo, the URL `/v3/api-docs`, and a green "Explore" button. The main title is **StudentsRooms API 1.0 (GAS 3.1)**. Below it, a subtitle reads "Documentación de la API StudentsRooms con seguridad JWT".

The main content area is titled **usuario-controller**. It displays the **GET /api/usuarios/{id}** endpoint. The "Parameters" section shows a required parameter `id` of type `string(Swift.String)`. The "Responses" section shows a successful response (200 OK) with a media type of `"json"`. An example value is provided:

```
{  
    "id": "3f6d1f64-5717-4562-b3fc-2c963f66ef0e",  
    "nombre": "Juan Pérez",  
    "email": "juan.perez@example.com",  
    "contraseña": "12345678",  
    "rol": "Administrador",  
    "totalHabitacionesPublicadas": 5073743824,  
    "totalReservasRealizadas": 1873741824  
}
```

Below this, a "Try it out" button is visible. A sidebar on the right contains "Links" and "No links". At the bottom, several other endpoints are listed: **PUT /api/usuarios/{id}** (disabled), **DELETE /api/usuarios/{id}** (disabled), **POST /api/usuarios/registro** (green background), **GET /api/usuarios** (disabled), and **GET /api/usuarios/raw** (disabled).

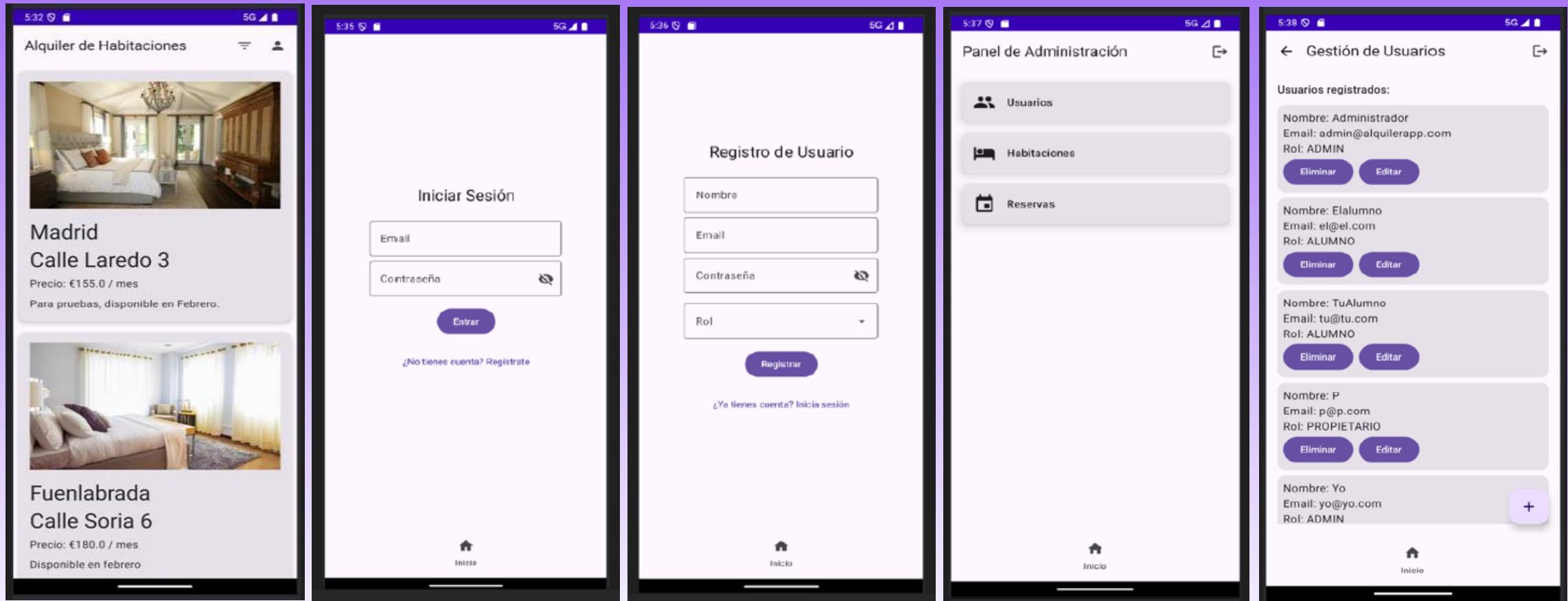
Manual de usuario

Instalación y uso

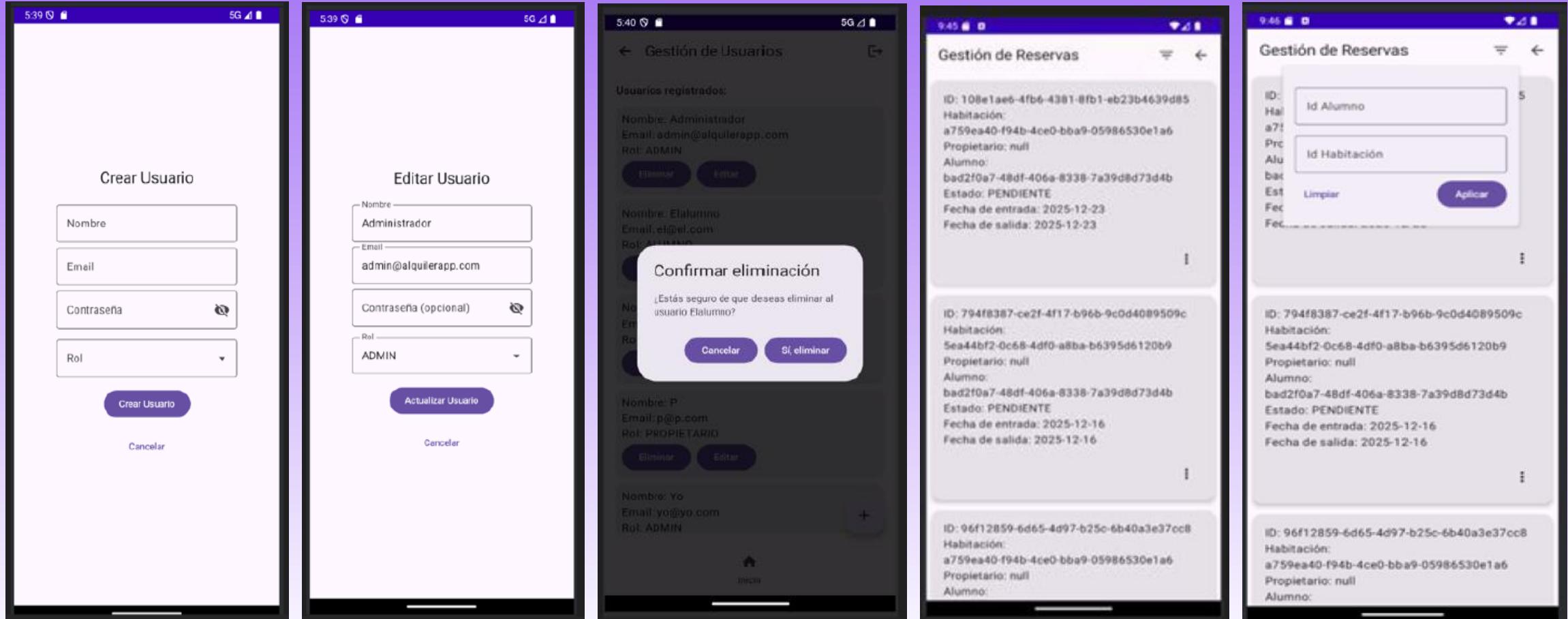
El proyecto se entrega en una carpeta comprimida que contiene:

- Las instrucciones de instalación.
- La memoria.
- Una carpeta con los archivos necesarios:
 - .apk, application.properties, docker-compose.yaml, .jar y .bat

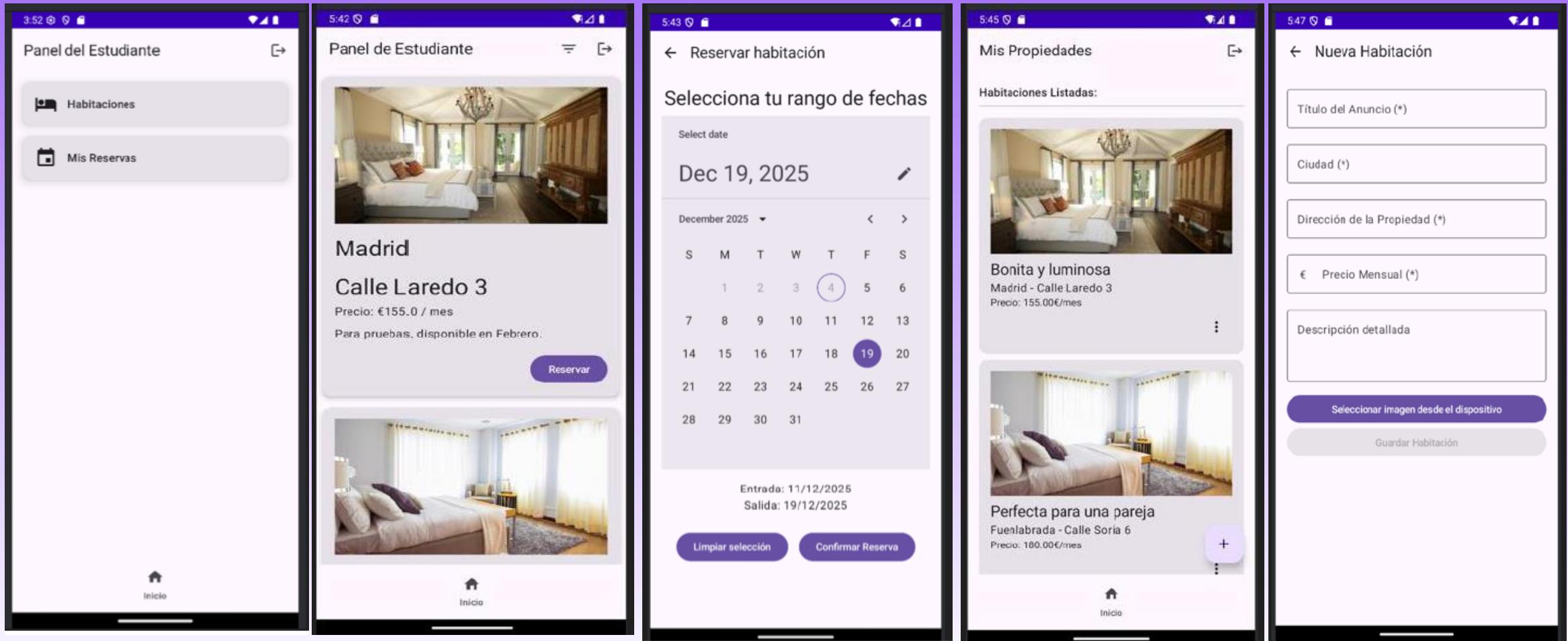
Instrucciones de uso



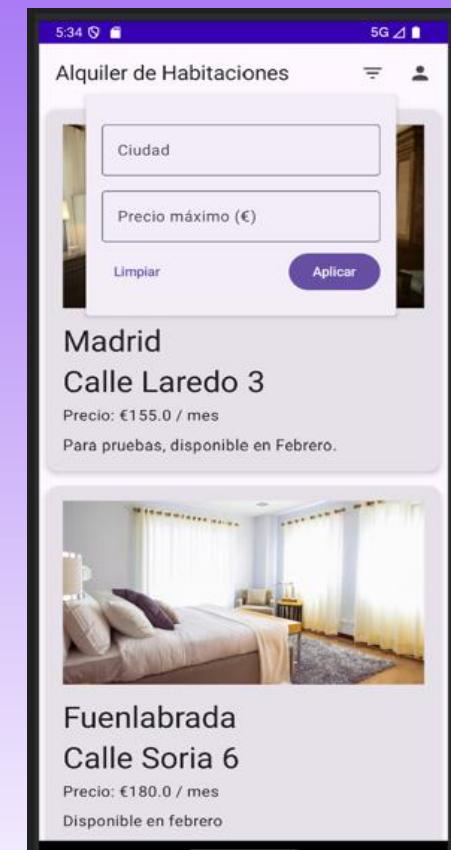
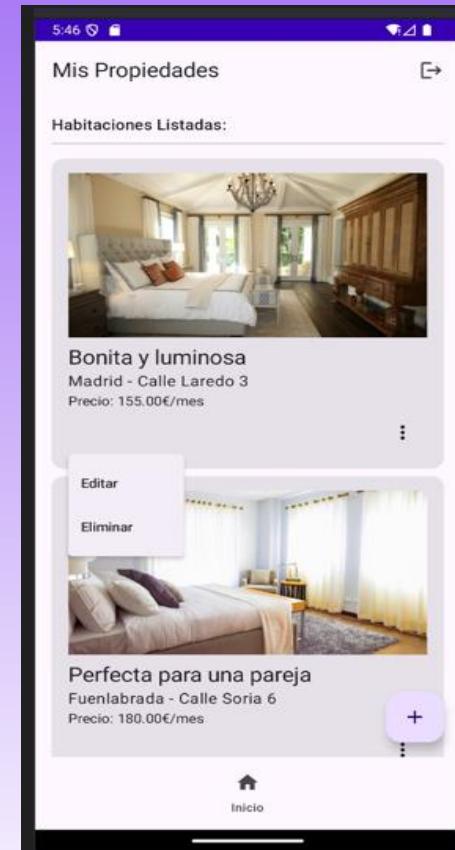
Instrucciones de uso



Instrucciones de uso



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>

Curriculum: **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