SOEN 390

Team 15 - Deliverable 1

Rently System Architecture Design

Winter 2023

Student	**Student ID**
Abdelkader Habel	40209153
Adam boucher	40165035
Adel Bouchatta	40175598
Anes Khadiri	40159080
Chems-Eddine Saidi	40192094
Francesco Ferrato	26642152
Omar Fares	40162541
Oussama Cherifi	40212275
Zakaria El Manar El Bouanani	40190432

Contents

1.Introduction

1.1 Identifying information

Throughout this report, the project will be referred to as “the project”, “the product” and “the platform”. It all refers to Rently, a condo management platform which will help management companies manage their properties and give access to their users to perform various actions on the platform.

1.2 Overview

The architecture for the system is separated into 3 main components. The frontend, the backend and the database. The database is a PostgreSQL database, the frontend is made with React and the backend in Spring Boot. The backend is to be split into several layers, going from a controller layer, a service layer and several layers up until the database.

2. Stakeholders and concerns

2.1 Stakeholders

The set of stakeholders that are impacted by this project contains owners and renters of units in buildings managed by companies who will use the product to trck the operations of their properties. It also includes the management companies, for whom this product is mainly made for. It includes the employees of said companies, who will interact with the system in their day-to-day operations. The stakeholders also include the members of the development team, who meet regularly and discuss with another stakeholder, the project owner – TA in this case – to advance this project.

2.2 Concerns

The project has several concerns.

- The end product should allow management companies to perform various operations on their properties, such as setting up common facilities for their occupants, answering their requests, welcoming new occupants and more.
- The end product should allow a renter or owner to have access to basic information about the unit they live in, give them access to an interface to make requests about their unit or even make reservations for a common facility.
- The deployment of the system is expected to be straightforward, with various parts of the architecture hosted on different providers’ platforms. The frontend is to be hosted on Netlify, the backend on Railways and the database on Neon.
- The system of interest is built at the backend with Spring Boot, allowing for a great separation of concerns and good modularity, thus allowing for good prospects for maintenance and scalability. Spring Boot is a framework that is

known and has been proven to work well due to various integrations that are available, especially with authentication and user-defined roles.

- The system is expected to be easily maintainable, due to how the backend is structured.

3. Viewpoint specification

3.1 Overview

This viewpoint focuses on the architectural, structural and behavioral aspects of the system, encompassing user interactions, property management, structure and deployment of the system.

3.2 Model kinds

- Domain model
- Component diagram
- Use case diagram
- Activity diagram
- Class diagram
- Backend Architecture diagram
- Deployment diagram

4.Views

4.1 View: Condo management system architecture

4.1.1 Domain model

Governing model kind: Domain models

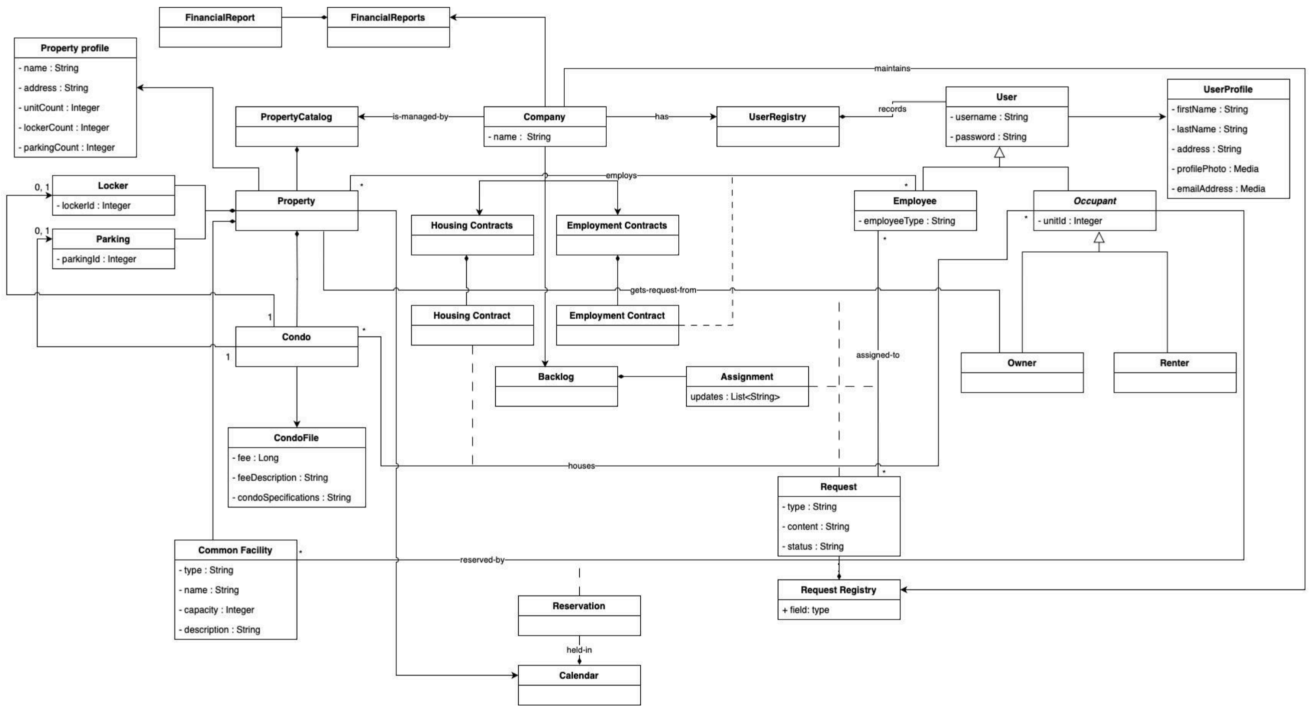

Figure 1: Domain model

This domain model represents the classes that represent real-life elements of the system. The company handles a few registries, making it a controller class and an entry point into the system. Association classes like “Assignment” or “Reservation” are connected to relations between other classes where the relation has a M-N multiplicity. The property class maintains several data structures for various objects, like condos, lockers or parkings. Several types of users require the use of inheritance to simplify development.

4.1.2 Component and architecture diagram

Governing model kind: Component Diagrams

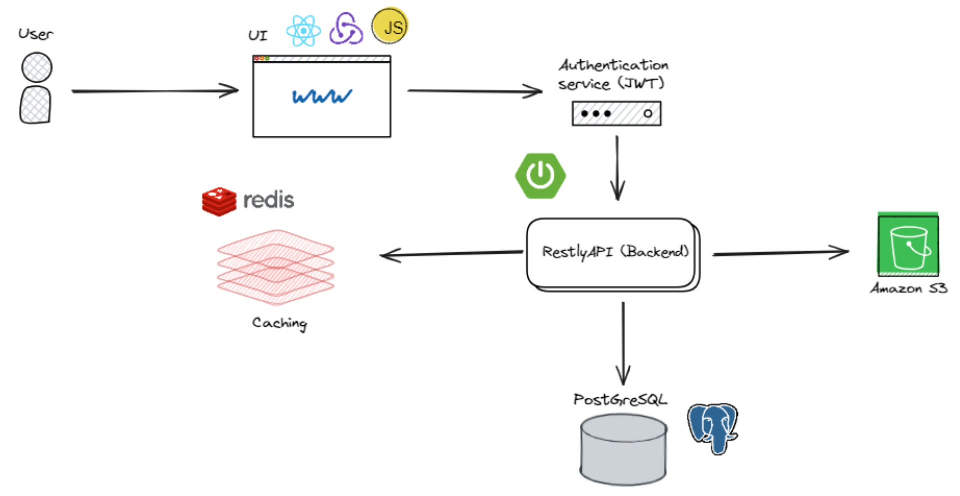

Figure 2: Component and architecture diagram

4.1.3 Use Case diagram

Governing model kind: Use case diagrams

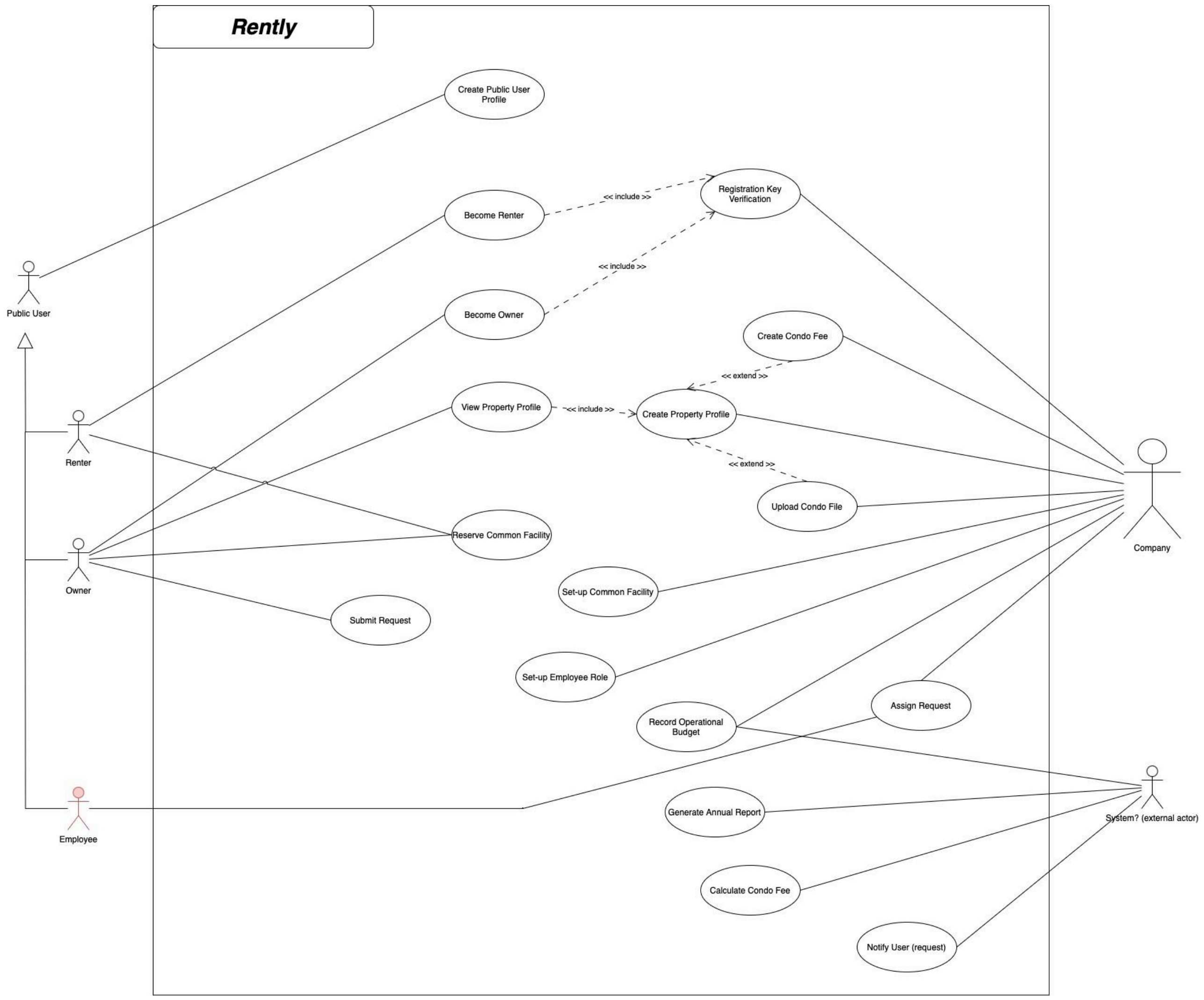

Figure 3: Use case diagram

The use case diagram shows all the actors that will interact with the system upon completion of the development. Three external types of actors, the renter, owner and employee fulfill various use cases, where some also involve the action of a system employee.

4.1.4 Activity diagram

Governing model kind: Activity diagrams

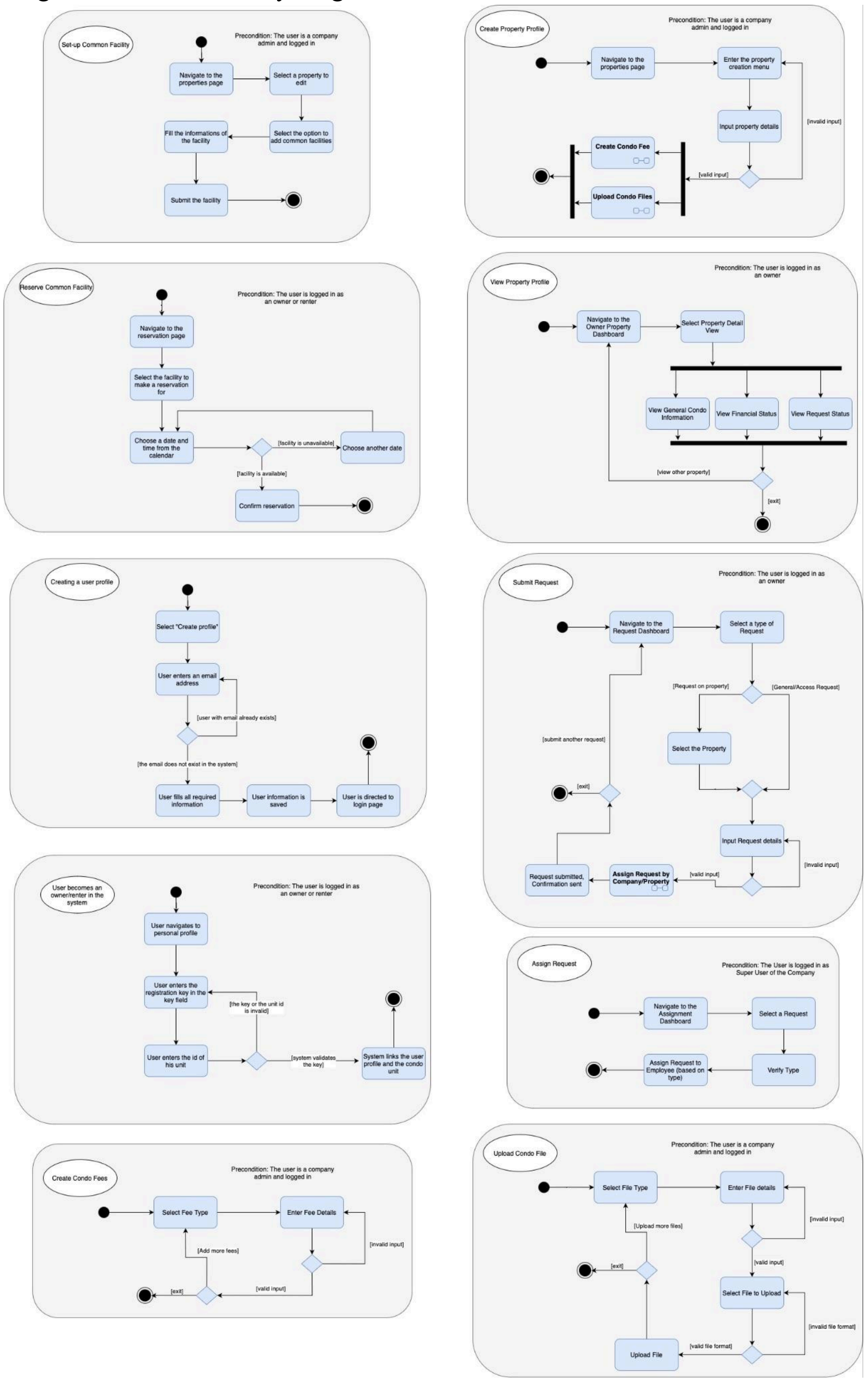
Set-up Common Facility
Precondition: The user is a company admin and logged in
Navigate to the properties page
Select a property to edit
Select the option to add common facilities
Fill the informations of the facility
Submit the facility
Create Property Profile
Precondition: The user is a company admin and logged in
Navigate to the properties page
Enter the property creation menu
Input property details
[invalid input]
[valid input]
Create Condo Fee
Upload Condo Files
Reserve Common Facility
Precondition: The user is logged in as an owner or renter
Navigate to the reservation page
Select the facility to make a reservation for
Choose a date and time from the calendar
[facility is unavailable]
Choose another date
[facility is available]
Confirm reservation
View Property Profile
Precondition: The user is logged in as an owner
Navigate to the Owner Property Dashboard
Select Property Detail View
View General Condo Information
View Financial Status
View Request Status
[view other property]
[exit]
Creating a user profile
Select "Create profile"
User enters an email address
[user with email already exists]
[the email does not exist in the system]
User fills all required information
User information is saved
User is directed to login page
Submit Request
Precondition: The user is logged in as an owner
Navigate to the Request Dashboard
Select a type of Request
[Request on property]
[General/Access Request]
Select the Property
[submit another request]
[exit]
Input Request details
[invalid input]
[valid input]
Assign Request by Company/Property
Request submitted, Confirmation sent
User becomes an owner/renter in the system
Precondition: The user is logged in as an owner or renter
User navigates to personal profile
User enters the registration key in the key field
[the key or the unit id is invalid]
User enters the id of his unit
[system validates the key]
System links the user profile and the condo unit
Assign Request
Precondition: The User is logged in as Super User of the Company
Navigate to the Assignment Dashboard
Select a Request
Verify Type
Assign Request to Employee (based on type)
Create Condo Fees
Precondition: The user is a company admin and logged in
Select Fee Type
Enter Fee Details
[invalid input]
[Add more fees]
[valid input]
[exit]
Upload Condo File
Precondition: The user is a company admin and logged in
Select File Type
Enter File details
[invalid input]
[Upload more files]
[valid input]
[exit]
Select File to Upload
[invalid file format]
[valid file format]
Upload File

Figure 4: Activity diagrams

Figure 4 represents the activity diagrams for a few use cases in the system. Several of them have preconditions about the existence and authentication status of the actor. They serve the purpose of explaining the flow of events for a given use case.

4.1.5 Class diagram

Governing model kind: Class diagrams

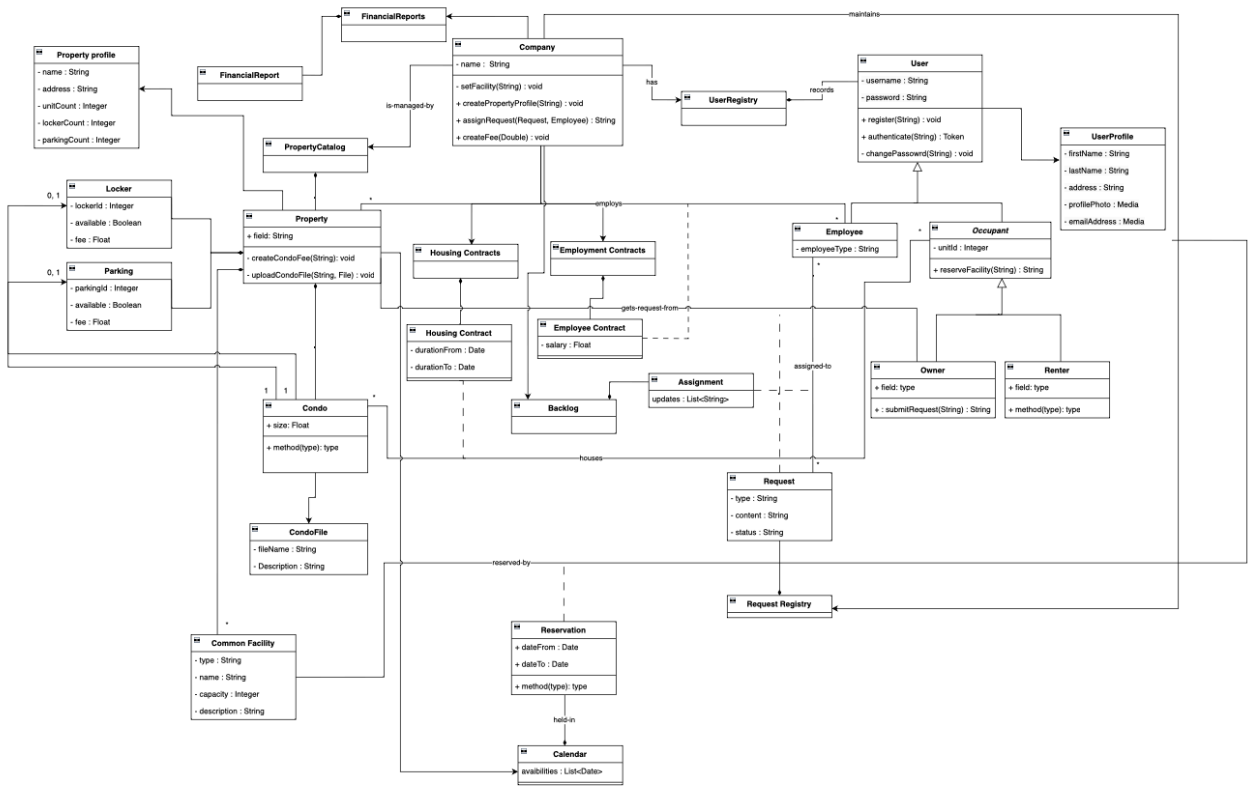

Figure 5: Class diagram

The above diagram is the class diagram, which has the functions that the classes will implement, as well as more attributes of the classes.

4.1.6 Backend architecture diagram

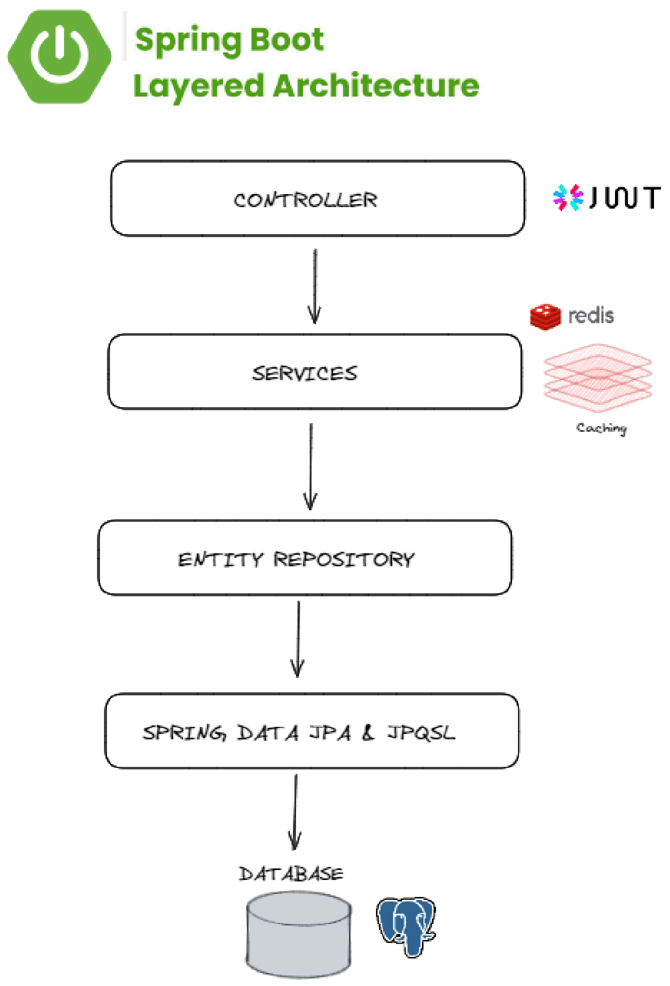

Figure 6: Backend architecture diagram

Figure 6 is a representation of how a Spring Boot backend is structured, going from the controller up to the Postgres database.

4.1.7 Deployment diagram

Governing model kind: Deployment diagrams

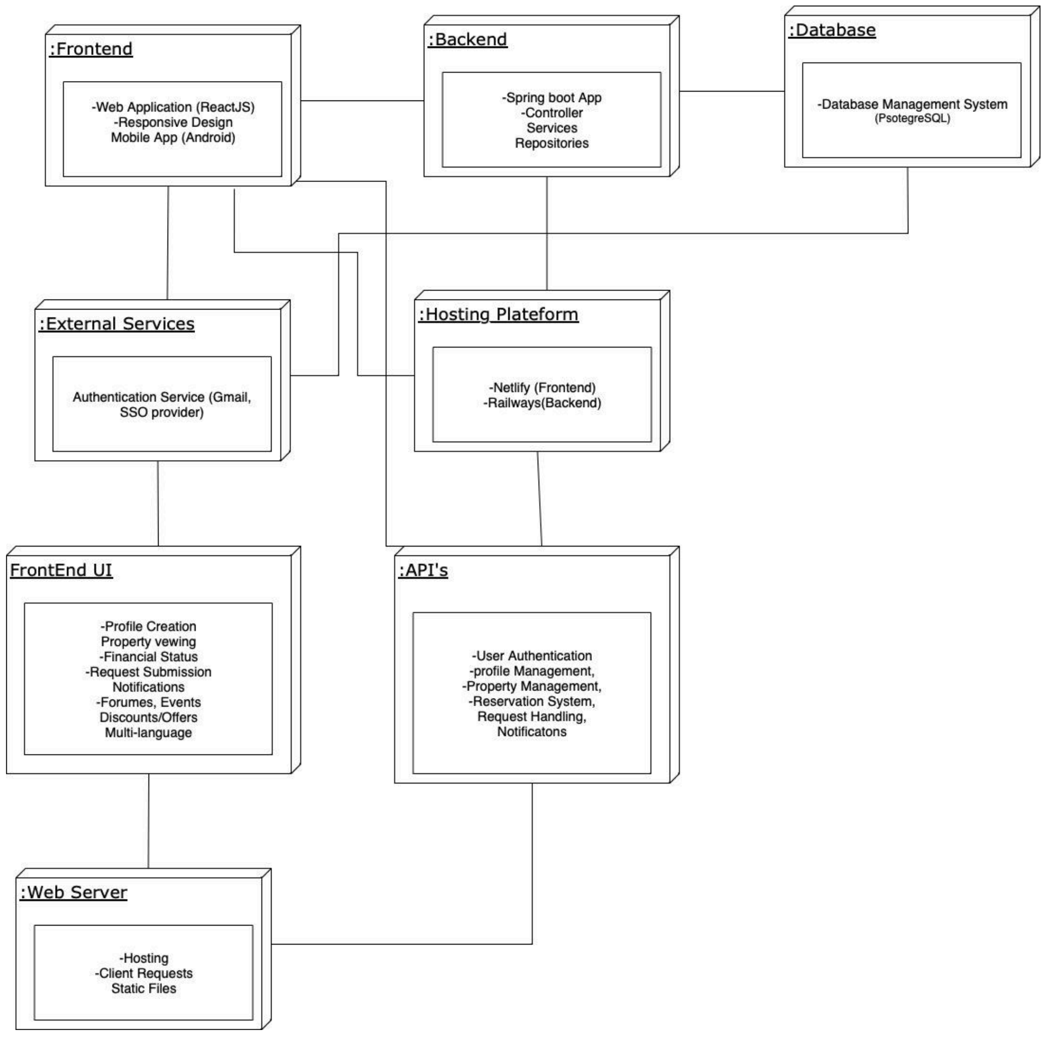

Figure 7: Deployment diagram

Figure 7 is a representation of how various parts of the platform are interconnected once they are hosted. It depicts the relationship between the hosting platform for the backend and frontend is linked to the backend for example, or how the external SSO login service is connected to the database.