NOVA SCHOOL OF SCIENCE & TECHNOLOGY

IADI 2023
Project Report

Autores:
Alexandre Oliveira 58059
André Cordeiro 57797
Bernardo Silva 57654

Table of Contents

Architecture Description ---------- *3*

Server-side application---------- 3

Client-side application ---------- 3

DataBase Schema---------- *4*

OpenAPI Summary ---------- *4*

Security rules---------- *5*

User Stories---------- *7*

IFML Diagrams ---------- *7*

Link to Commits ---------- *8*

Video ---------- *8*

Group Dynamics---------- *9*

Lessons Learned---------- *9*

Auto-evaluation---------- *9*

Architecture Description

Server-side application

Our server-side application implements a layered architecture using the Spring framework in Kotlin with the help of Spring Web, Spring Boot, Spring Security and Spring Data components. The layers comprised in our architecture are the following:

- **Data Layer**: comprised of our data package. Includes the DAO (Data Access Objects) of every entity on the application and their respective repositories.

- **Presentation Layer**: comprised of our presentation package. Includes the all the necessary DTOs (Data Transfer Objects) to present information to clients and the controllers with the backend logic for every API:
  - Apartment API (operations related to apartments)
  - Booking API (operations related to bookings)
  - Client API (operations related to clients)
  - Owner API (operations related to owners)
  - Period API (operations related to periods)
  - Review API (operations related to reviews)

- **Service Layer**: comprised of our service package. To each of the APIs mentioned above corresponds a service, which verifies the domain rules and interacts with the corresponding repository.

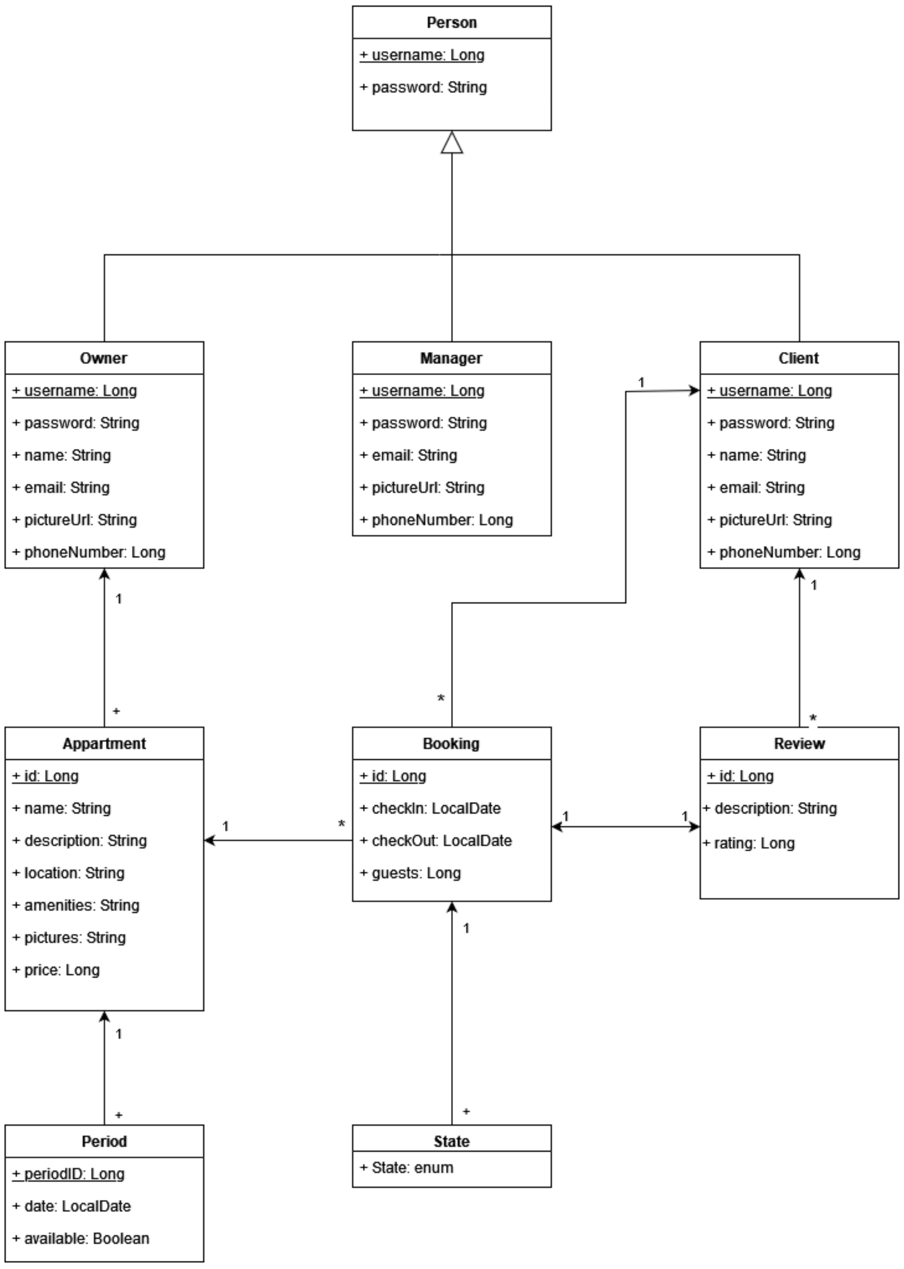
Client-side application

To design the client-side application we used Typescript recurring to the React library, as well as CSS to customize the look of the UI. We also used ChatGPT to help us write the CSS code significantly faster and save time.
The client-side application architecture has the following components:

- **HomePage:** Welcoming entry point to our application, providing users with a comprehensive overview.
- **Login:** Contains the login form, facilitating secure access to the application.
- **Apartments:** Displays a curated list of apartments tailored to the user's role—clients see all available apartments, while owners view their own properties.
- **ApartmentDetails:** Offers detailed information about a specific apartment, featuring role-specific functionalities such as period creation for owners and booking capabilities for users.
- **BookingsList:** Displays bookings based on user roles, clients will view their reservations, while owners can verify the bookings associated with each of his apartments.
- **BookingDetails:** Contains information about a specific booking, mentioning important information regarding not only the booking, but also about the apartment and client which the booking is about, as well as the review of the booking.

- **ReviewList:** Lists the reviews associated with a given apartment. It also displays certain information based on the user role
- **CreateReview:** Renders a form for the client to review his stay at a specific apartment.

DataBase Schema



OpenAPI Summary

Below is brief summary list of the APIs in our system but you can see the full specification in detail here.

- **Apartment API**:
    - Get all apartments registered in the system.
    - Get an apartment with a given id.
    - Get the period of an apartment given the apartment id and period id.
    - Get all periods of an apartment with a given id.
    - Get all available periods of an apartment with a given id.
    - Get all available apartments in a given time period.
    - Book an apartment with a given id.
    - Create a new period for an apartment with a given id.
    - Get all reviews of an apartment with a given id.

- **Booking API**:
    - Get a booking with a given id.
    - Get the review of booking with a given id.

- **Client API**:
    - Create a new client.
    - Get a client with a given id.
    - Get all bookings of a client with a given id.
    - Get all reviews of a client with a given id.
    - Check-in of the booking with a given id.
    - Check-out of the booking with a given id.
    - Cancel the booking with a given id.

- **Owner API**:
    - Get all apartments of an owner with a given id.
    - Get all bookings made in the apartments of an owner with a given id.
    - Get an owner with a given id.
    - Accept or reject a booking with a given id.

- **Period API**:
    - Get a period with a given id.

- **Review API**:
    - Create a review for a booking with a given id.
    - Get a review with a given id.
    - Get all reviews on the system.

Security rules

Below is a list with every security rule we defined for our system:

- **CanListApartments:** always true.

- **CanCreateApartmentPeriod**: verifies if the user trying to create a period for an apartment is the owner of the given apartment.

- **CanBookApartment**: verifies if the user trying to book an apartment is a client to which the shipment will belong (a client cannot create a shipment on behalf of another client).

- **CanGetApartmentPeriods:** always true.

- **CanListApartmentAvailablePeriods: always true.**

- **CanListApartmentReviews**: verifies if the user trying to list the reviews an apartment is the owner of the apartment or if it is a client.

- **CanChangeBookingStatus**: verifies if the user trying to accept or reject a booking is the owner of the apartment or if it is a manager.

- **CanGetOwnerApartments**: verifies if the user trying to see the apartments of an owner is that owner of them.

- **CanCheckIn**: verifies if the user trying to check-in a booking is the client who made the booking or if it is a manager.

- **CanCheckOut**: verifies if the user trying to check-out a booking is the client who made the booking or if it is a manager.

- **CanCancel**: verifies if the user trying to cancel a booking is the client who made the booking or if it is a manager.

- **CanGetClientBookings**: verifies if the user trying to see the bookings of a client is the client to whom the bookings belong to.

- **CanGetClientReviews**: verifies if the user trying to see the reviews of a client is the client to whom the reviews belong to.

- **CanCreateReview**: verifies if the user trying to review a booking is the client who made the booking.
- **CanGetAllReviews: always true.**

- **CanGetReview: always true.**

- **CanGetBooking**: verifies if the user trying to see a booking is the client who made the booking or if it is the owner of the apartment booked.

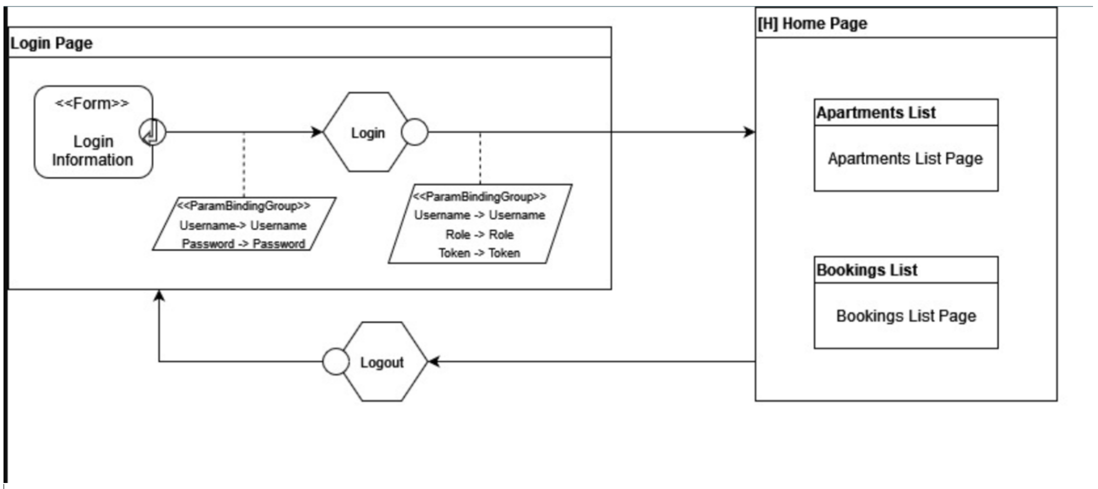
- **CanGetBookingReview**: verifies if the user trying to see the review of a booking is the client who made the booking or if it is the owner of the apartment booked.

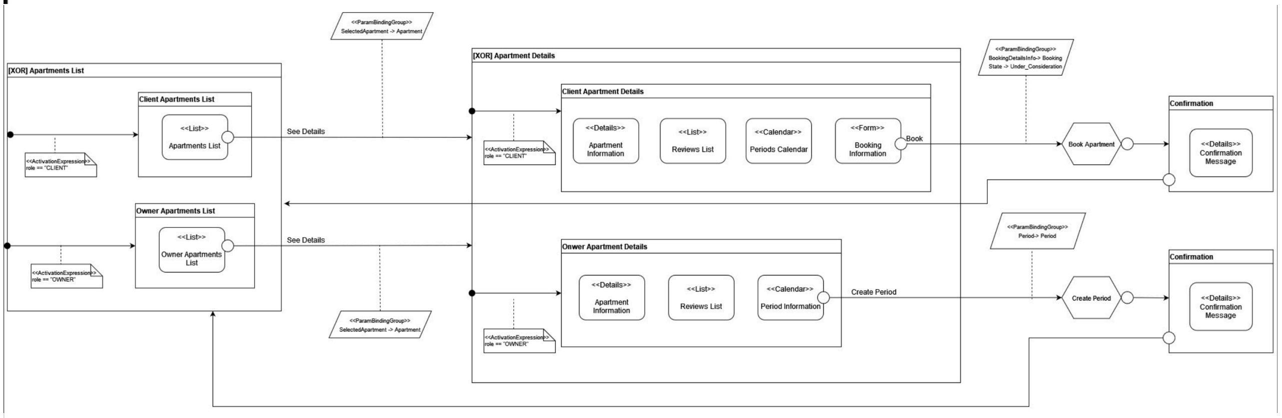
User Stories

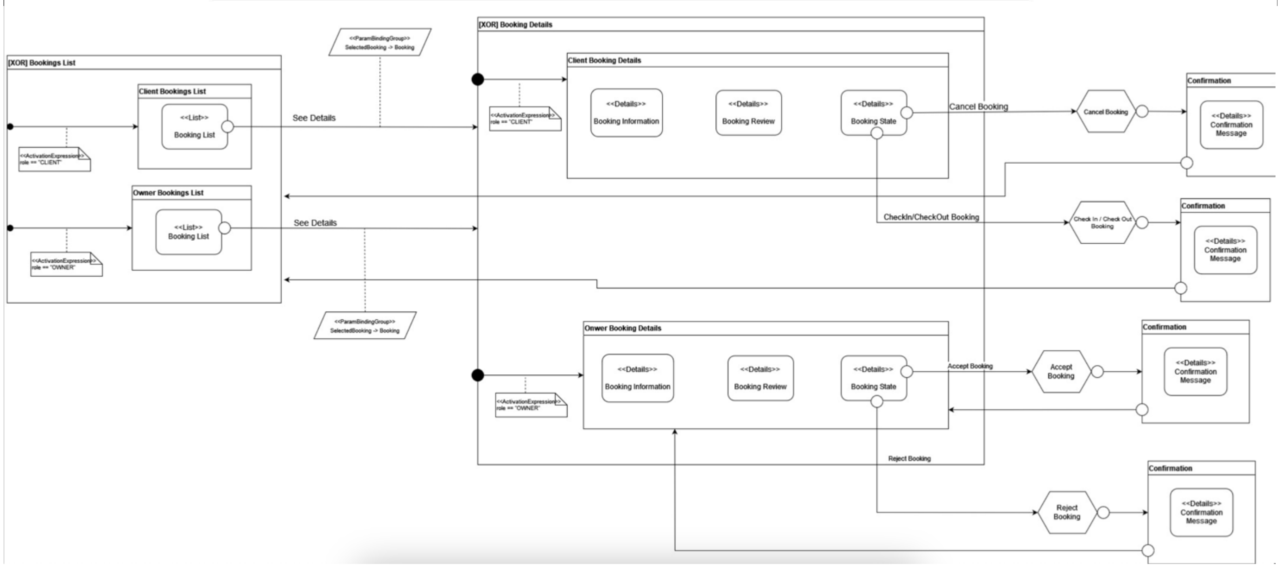
The following user-stories were implemented:

- As a client, I want to see the first page of a list of apartments so that I can choose one to rent and see its details.
- As a client, I want to see the next page in the list of apartments so that I can choose one to rent and see its details.
- As a client, I want to see the details of an apartment so that I can read its description and amenities and decide if I want to rent it.
- As a client, I want to see the calendar of an apartment so that I can decide when to rent it.
- As a client, I want to see the reviews of an apartment so that I can decide if I want to rent it.
- As a client, I want to select a starting date and an ending date in the calendar of an apartment to start making a reservation.
- As a client, I want to fill a form with the number of people to finish making a reservation.
- As a client, I want to see the details of a reservation so that I can see if it was accepted.
- As a client, I want to see the details of a reservation so that I can cancel it and see the result in the details of the reservation.
- As a client, I want to see the details of a reservation so that I can check-in and see the result in the details of the reservation.
- As a client, I want to see the details of a reservation so that I can check-out and see the result in the details of the reservation.
- As an owner, I want to see the list of my apartments so that I can select one apartment and see its details.
- As an owner, I want to see the details of an apartment so that I can add a period of availability and see the result in the calendar.
- As an owner, I want to see the details of an apartment so that I can see the list of reservations for it.
- As an owner, I want to see the details of a reservation so that I can accept it and see the result in the list of reservations.

IFML Diagrams







Link to Commits

1. First submission
2. Final submission

Video

To watch a quick video demonstrating our application click here: link

Group Dynamics

Our group was comprised of 3 students who equally contributed to the development of the whole application, distributing tasks across all group members and with every member having a completed understanding of the work done.

Lessons Learned

Throughout the development of this project, several valuable lessons have been learned:

- Developing the server-side application using the Spring framework and Kotlin provided insights into concise coding, scalability, and integration ease.

- Creating comprehensive OpenAPI documentation emerged as a crucial practice, serving as a guide for future developers and facilitating smooth backend-frontend communication.

- The design of the database schema underscored the importance of careful consideration for data relationships, integrity, and efficiency.

- Implementing the client-side application with React, Redux and TypeScript provided practical insights into state management, component-based architecture, and the advantages of static typing.

- Integration of automated tests into the development workflow ensured code reliability and facilitated early issue detection.

- Creating IFML specifications for the user interface improved planning and visualization, aligning development efforts with user stories.

Auto-evaluation

Regarding auto-evaluation, our Server-side application supports all required operations plus a few more that we found necessary or interesting to have. We also implemented all the necessary security rules to avoid unwarranted access to certain endpoints, as well as the use of JWT tokens for authentication. On the Client-side application we implemented every mandatory use case through an intuitive and minimalist UI. Having a fully functional and tested application with every mandatory feature and implementation guideline, as well as all the other deliverables, we believe we deserve 19.