Institut supérieur d'électronique de Paris

II.2406 - TECHNOLOGIES WEB

REPORT

HOME EXCHANGE MANAGER

TEAM: LA VACA POLACA

Nizar Bikti	nizar.bikti@eleve.isep.fr
Magdalena Pakula	magdalena.pakula@eleve.isep.fr
Aleksandra Banasiak	aleksandra.banasiak@eleve.isep.fr
Mario Aroca Paéz	mario.aroca-paez@eleve.isep.fr

June 5th 2023

Abstract

This report presents an overview of the Home Exchange Manager project, a web application aimed at facilitating home exchanges for short vacations. The project's primary goal is to create a user-friendly platform that allows individuals to connect and exchange their homes, enabling them to explore new locations while reducing accommodation expenses. The report provides an overview of the project requirements, including free home exchanges, defined services and constraints, home description and media, messaging system, member registration and authentication, member profiles, booking functionality, detailed housing visualization, search functionality, rating system, and administrator back-office interface. The chosen technologies for the project are discussed, including Java, Spring Framework, Gradle, Thymeleaf, IntelliJ IDEA, MySQL, and GitHub. The report also explores the application architecture and detailed design, focusing on the implementation of the MVC pattern and the interactions between different software components. Additionally, the database design and project management aspects, such as planning, releases, and team member roles, are addressed. Overall, the Home Exchange Manager application offers a user-friendly and secure platform for facilitating home exchanges, leveraging modern web technologies and best practices in software development.

Table of Contents

1. Introduction....4
2. Project Requirements....4
3. Chosen technologies and motivations....5
4. Application Architecture and detailed design....6
    4.1. MVC pattern implementation....6
    4.2. Software components and their interactions....10
    4.3. Project's design....11
    4.4. Database design....16
5. Project management....17
    5.1. Planning....17
    5.2. Releases....17
    5.3. Team members roles....18
6. How to install....18
7. How to use....19
8. Conclusions....21
    8.1. Learned lessons....21
    8.2. Perspectives....22

1. Introduction

This report provides an overview of the Home Exchange Manager project, a web application designed to facilitate home exchanges for individuals during short vacations. The purpose of this report is to introduce the project, outlining its objectives and key features.

The Home Exchange Manager project aims to create a user-friendly web platform for individuals to connect and exchange their homes, allowing them to experience new locations while reducing accommodation costs. The application will provide features such as member registration, home description sharing, messaging system, booking functionality, search capabilities, and a rating system.

This report briefly discusses the project requirements, chosen technologies, and overall objectives. It provides a high-level understanding of the project and sets the stage for further discussions on the project's detailed design and implementation.

2. Project Requirements

The Home Exchange Manager project aims to develop a web application that facilitates home exchanges for individuals during short vacations, typically ranging from a day to a few weeks. The core requirements of the project are as follows:

1. **Free Home Exchanges:** The primary functionality of the application is to enable free home exchanges. Users can register and list their home for exchange, specifying the desired duration and availability.
2. **Defined Services and Constraints:** Homeowners have the ability to define specific services that guests must provide during their stay. These services can include tasks such as pet care, plant watering, and house cleaning. Additionally, homeowners can set constraints and limitations that guests must adhere to, such as no smoking in the accommodation, no noise after 23:00, and restrictions on the number of children or pets allowed.
3. **Home Description and Media:** To showcase their homes, homeowners are required to provide a detailed description along with at least three photos for each listing. This information helps potential guests get a clear understanding of the accommodation.
4. **Messaging System:** All communication between users take place through the web application's messaging system. This ensures a secure and centralized platform for discussing exchange details, clarifying queries, and finalizing arrangements.
5. **Member Registration and Authentication:** The application includes a registration and authentication system to ensure that only verified users can participate in home exchanges. This helps establish trust and maintain the integrity of the platform.
6. **Member Profiles:** Each member should have a profile where they can provide relevant information about themselves for potential exchange partners.
7. **Booking Functionality:** The application should provide a booking mechanism that allows users to request and confirm home exchanges. Homeowners have the flexibility to accept or reject guests before finalizing the exchange.
8. **Detailed Housing Visualization:** A detailed view of each housing listing should be available, including a comprehensive description, location details, the services offered, and the constraints or limitations imposed by the homeowner. This enables users to make informed decisions about potential exchanges.
9. **Search Functionality:** The application should offer both simple and advanced search options to help users find suitable home exchange opportunities based on their desired location and duration.

10. **Rating System:** A dedicated rating system allows users to provide feedback and rate their exchange experience. This helps in building a reputation system and fostering trust within the community.
11. **Administrator Back-Office Interface:** An administrator has access to a dedicated back-office interface that allows them to view and manage member profiles, listings, and messages. Additionally, the administrator has the ability to delete inappropriate or spam listings.

3. Chosen technologies and motivations

For the development of the Home Exchange Manager application, the team choose the following technologies:

1. **Java**: Widely adopted programming language known for its robustness, scalability, and extensive community support. It provides a secure and reliable foundation for building enterprise-level web applications.
2. **Spring Framework**: It was chosen due to its comprehensive features and modular architecture. Spring provides excellent support for building web applications, implementing the MVC (Model-View-Controller) pattern, and facilitating the integration of various components.
3. **Gradle**: Powerful build automation tool that offers flexibility and scalability for managing dependencies, compiling source code, and generating deployment artifacts. Its declarative configuration approach simplifies project management and enhances the overall development experience.
4. **Thymeleaf**: Thymeleaf is a modern server-side Java template engine that seamlessly integrates with Spring. It offers a natural and intuitive way to build dynamic web pages by combining HTML templates with server-side processing. Thymeleaf's versatility and ease of use make it an ideal choice for rendering views in our web application.
5. **MySQL**: MySQL, a widely used relational database management system, has been chosen for data storage and management. It offers reliability, scalability, and excellent performance, making it suitable for handling the application's data requirements.
6. **GitHub**: To ensure efficient version control and collaboration, we have chosen GitHub as our preferred platform. GitHub enables seamless code sharing, facilitates effective team collaboration, and provides robust version control features.
7. **IntelliJ IDEA**: We have selected IntelliJ IDEA as our Integrated Development Environment (IDE) for its powerful features, code analysis capabilities, and seamless integration with the chosen technologies. IntelliJ IDEA provides a user-friendly interface that boosts productivity and facilitates efficient development.

The motivations behind these technology choices are as follows:

- **Java and Spring**: Java's wide adoption and Spring's extensive features make them a reliable combination for building enterprise-grade web applications. The Spring Framework provides a modular architecture, dependency injection, and built-in support for MVC, making it a suitable choice for implementing the application's core functionalities.
- **Gradle**: Gradle simplifies project management and dependency handling. Its flexibility and scalability help streamline the build process, enabling efficient development and deployment.
- **Thymeleaf**: Thymeleaf's integration with Spring and its intuitive templating capabilities allow us to build dynamic web pages with ease. Its seamless integration with the chosen technologies ensures a smooth development experience.

- **MySQL**: MySQL's reliability, scalability, and performance make it a suitable choice for handling the application's data storage and management requirements. Its compatibility with Java and Spring further simplifies the integration process.
- **GitHub**: GitHub provides a robust version control system, facilitating collaborative development and effective team coordination. Its features, such as pull requests and issue tracking, enable seamless code sharing and efficient project management.
- **IntelliJ IDEA**: IntelliJ IDEA's rich feature set, code analysis tools, and seamless integration with the chosen technologies enhance productivity and facilitate efficient development.

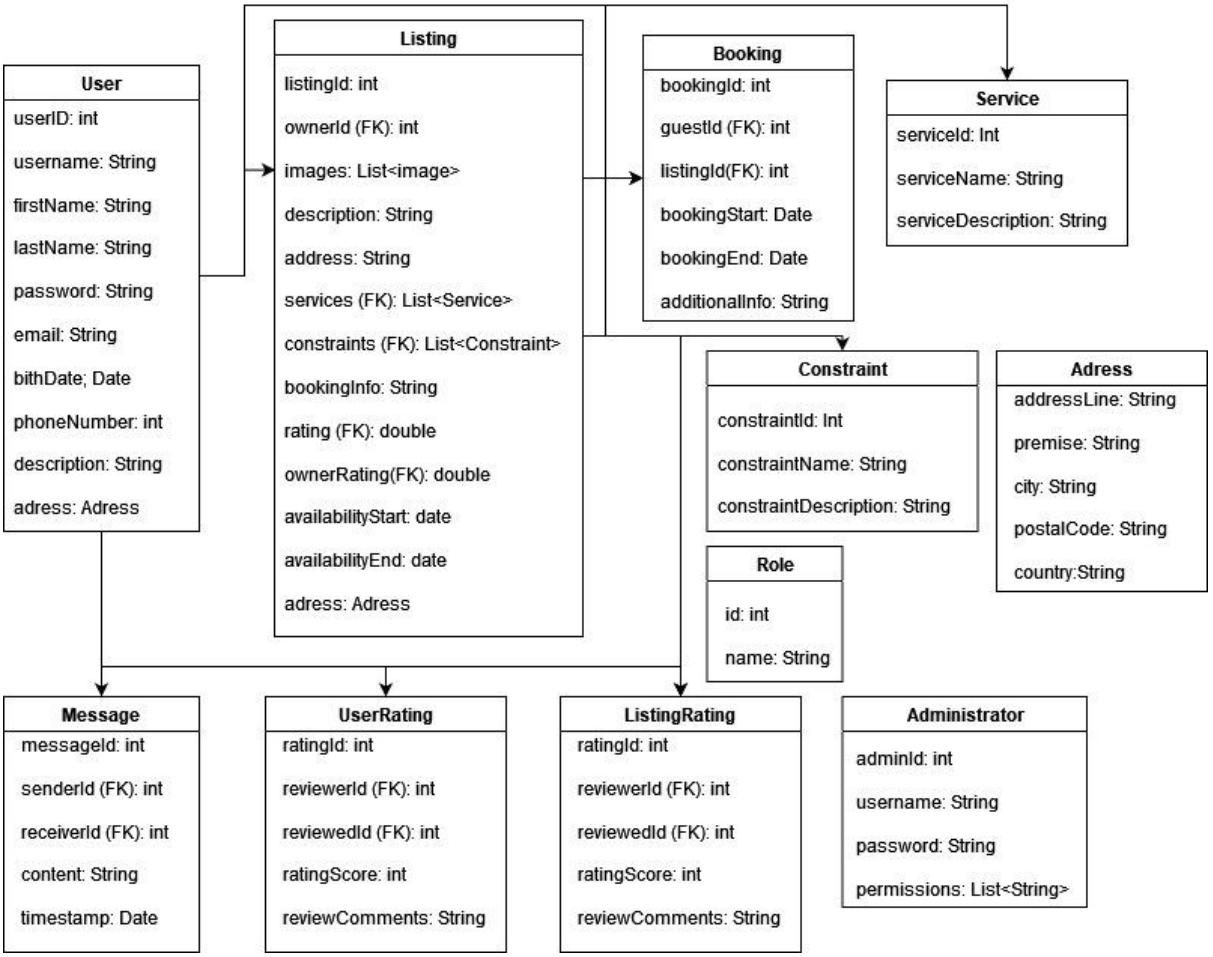
These technologies were chosen based on their proven track record, community support, and ability to meet the specific requirements of the Home Exchange Manager application. Their combination offers a solid foundation for building a scalable, secure, and user-friendly web application.

4. Application Architecture and detailed design

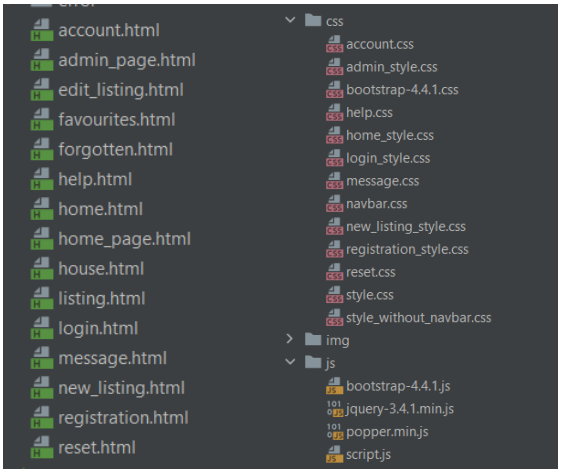
4.1. MVC pattern implementation

The Home Exchange Manager application follows the Model-View-Controller (MVC) architectural pattern, which separates the concerns of data management, user interface presentation, and user interaction. The MVC pattern provides a structured approach to designing and developing web applications, enhancing modularity, maintainability, and testability.

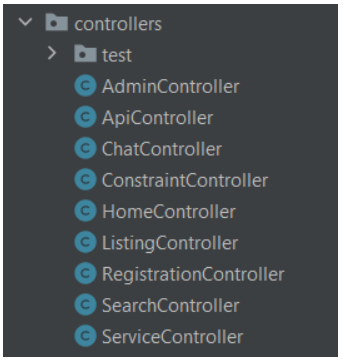
1. **Model**: The Model represents the application's data and business logic. In our implementation, the Model consists of Java classes that define the data entities, such as User, Listing , Booking, and Message. These classes encapsulate the state and behavior of the respective entities and provide methods to interact with the underlying data storage. The different models used for this project are defined in the following entity diagram:



2. **View**: The View is responsible for presenting the user interface to the application's users. In our implementation, we utilize Thymeleaf templates to generate dynamic HTML pages. The Thymeleaf templates, combined with CSS for styling and JavaScript for interactivity, enable the rendering of user-friendly and visually appealing views. Here it is possible to view all the different views created for this project:



3. **Controller**: The Controller acts as an intermediary between the Model and the View, handling user requests, updating the Model, and selecting the appropriate View to present the response. In our implementation, Spring's MVC framework is leveraged to define controllers that handle specific routes and request methods. These controllers receive user input, interact with the Model to retrieve or update data, and select the appropriate View to render the response.

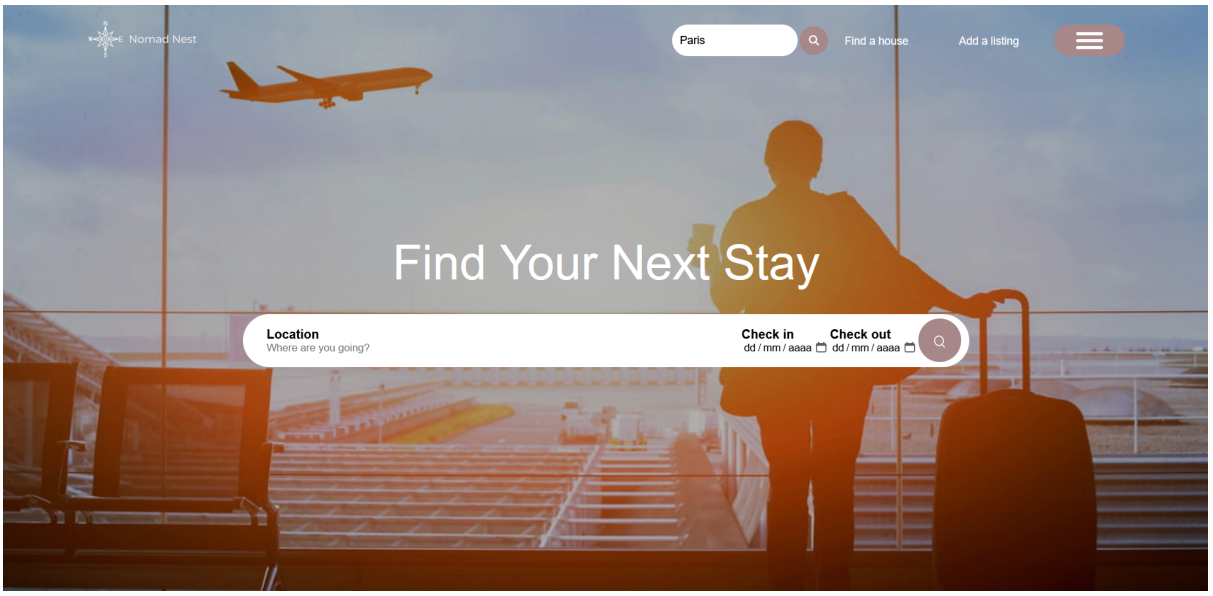


The interaction among these components in the MVC pattern follows a defined flow:

1. The user interacts with the application through the user interface (View), triggering a request.
2. The request is received by the corresponding controller, which processes the user input, validates it, and performs any necessary operations on the Model.
3. The controller updates the Model with the modified data and selected attributes. Then it selects an appropriate View to render the response.
4. The selected View accesses the Model to retrieve the required data and dynamically generates the HTML response, incorporating the data into the Thymeleaf templates.
5. The rendered HTML response is sent back to the user's browser, completing the request-response cycle.

Let's see one practical example of this functioning with the Search by city operation:

First the user is in the home page, equivalent to the home_page.html file. He will insert the name of a city in the search bar, in this case Paris in order to search for available listings in that specific city



By doing so when clicking the search button the user sends a GET request to the controller with the specific url of **/listings/search/simple**

```
<form th:action="@{/listings/search/simple}" method="get" class="form">
    <input type="search" placeholder="Search in a city..." class="search-field" name="citySimple">
    <button type="submit" class="search-button"><i class="fa fa-search"></i></button>
</form>
```

Then that specific URL is handled by the listingsByCity method thanks to the **GetMapping** annotation inside the **SearchController.java** class. It receives the parameter citySimple which is the city name inserted by the user and filters from the database the listings from that city. The retrieved listings are added to the Model object using the **model.addAttribute** method. This makes the listings available to the view template for rendering. The method returns **/listing**, which represents the logical view name. This indicates that the view template to be rendered for this request is named listing.html which will be the file where the listings list will be displayed

```java
@GetMapping("/listings/search/simple")
public String listingsByCity(@RequestParam String citySimple, Model model) {
    List<Listing> listings = listingService.findByCity(citySimple);

    model.addAttribute( attributeName: "listingsSimpleSearch", listings);
    model.addAttribute( attributeName: "simpleSearch",  attributeValue: true);

    return "/listing";
}
```

Finally in the listing.html file we can find a conditional block that checks if there are any search results available from the controller results. If there are no results (listingsSimpleSearch is null or empty), the message "No listings found in this city.." is displayed.

If there are search results available, the code enters the second div. Within this div, the code loops through the listingsSimpleSearch and populates the HTML dynamically. Note how in the controller the attributes *listingsSimpleSearch* which is the actual result and *simpleSearch* which is a boolean to indicate that the search has been made, are the ones used in the html file in order to show the results

```html
<!-- Listings by simple search -->
<div class="left-col" th:if="${simpleSearch}">
    <h2> Simple search results: </h2>
    <div class="left-col" th:if="${listingsSimpleSearch == null or listingsSimpleSearch.isEmpty()}" >
        <p> No listings found in this city.. </p>
    </div>
```

```html
    <!-- Looping through the listings and populating the HTML dynamically -->
    <div th:if="${listingsSimpleSearch != null}" th:each="listing : ${listingsSimpleSearch}" class="house">
        <a th:href="@{'/house/' + ${listing.listingId}}">CLICK HERE FOR MORE INFO</a>
        <div th:each="image : ${listing.getImages}" class="house-img">
            <img th:src="@{${image.getUrl}}" alt="Listing Image">
        </div>
        <div class="house-info">
            <p th:text="${listing.addressLine}"></p>
            <h3 th:text="${listing.description}"></h3>
            <!-- Rating stars -->
            <span th:each="i : ${#numbers.sequence(1, listing.rating)}">
        <i class="fas fa-star"></i>
    </span>
            <span th:each="i : ${#numbers.sequence(1, 5 - listing.rating)}">
        <i class="far fa-star"></i>
    </span>
        </div>
    </div>
</div>
```

By implementing the MVC pattern, we achieve a clear separation of concerns, enabling the independent development and testing of each component. The Model encapsulates the data and business logic, promoting reusability and modularity. The View focuses on presenting a visually appealing and interactive user interface. The Controller handles user input, coordinates the flow of data, and selects the appropriate View for rendering the response.

The MVC pattern implementation in the Home Exchange Manager application ensures a structured and organized approach to development, enhancing maintainability, scalability, and code readability. It also facilitates collaboration among team members, as the separation of responsibilities allows for parallel development and easy integration of different components.

4.2. Software components and their interactions

The Home Exchange Manager application comprises several software components that work together to provide the desired functionalities. Each component has specific responsibilities and interacts with others to achieve the overall system behavior. The following are the key components and their interactions:

1. **User Management:**
    - Responsible for user registration, authentication, and profile management.
    - Interacts with the Database component to store and retrieve user information.
2. **Listing Management:**
    - Handles the creation, updating, and retrieval of home listings for exchange.
    - Interacts with the Database component to store and retrieve listing information.
    - Utilizes the Messaging System component to communicate with users regarding listing inquiries and booking requests.
3. **Booking System:**
    - Manages the booking process, allowing users to request and confirm home exchanges.
    - Interacts with the User Management and Listing Management components to validate user credentials and check listing availability.
4. **Messaging System:**
    - Facilitates communication between users through an integrated messaging platform.
    - Provides a secure and centralized environment for users to discuss exchange details and finalize arrangements.
5. **Search System:**
    - Enables users to search for suitable home exchange opportunities based on location and duration preferences.
    - Interacts with the Database component to retrieve listings matching search criteria.
    - Integrates with the User Management component to filter search results based on user preferences and constraints.
6. **Administrator Back-Office Interface:**
    - Provides an administrative interface for managing member profiles, listings, and messages.
    - Interacts with the Database component to perform CRUD (Create, Read, Update, Delete) operations on system data.
    - Utilizes the Messaging System component to monitor and moderate user communication.

These components work together through well-defined APIs and data flows. For example, the user management component interacts with the database component to obtain user information during authentication. The list management component communicates with the messaging system component to process user requests and update list availability. Additionally, the search system component interacts with the database component to retrieve list data based on the user's search criteria.

In order to understand better the interactions here we can see some examples of the Backend controllers endopoitns:

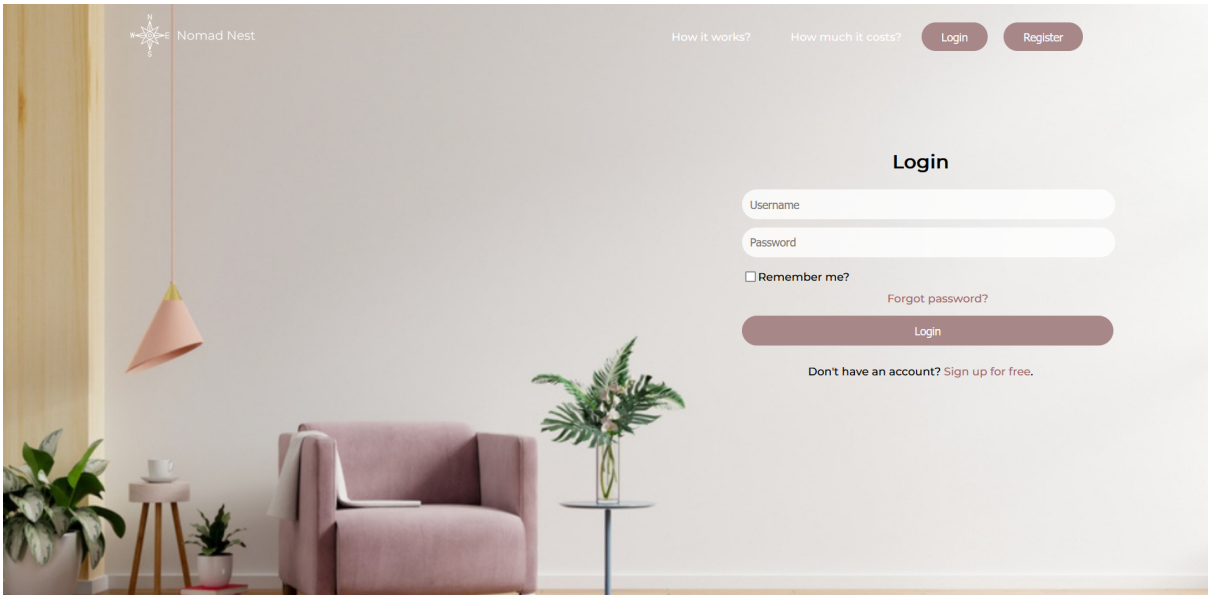
Controller Name	Endpoint	Request Type	Attributes	Description
Home Controller	/success	GET	role	Redirect to home page depending on role

Home Controller	/listing	GET	allListings	Return all Listings from home page
Home Controller	/home_page	GET	listings	Return all Listings from listings page
Listing Controller	/new_listing	GET	constraints & services	Provide necessary data to create new listing
Listing Controller	/new_listing	POST	listing	Create new listing with provided data
Listing Controller	/listing/delete/{id}	POST	listing	Delete 1 listing
Listing Controller	/listing/edit/{id}	POST	listing	Edit 1 listing
Admin Controller	/admin/delete/listing/{id}	POST	listing	Admin delete listing
Admin Controller	/admin/delete/user/{id}	POST	user	Admin delete user

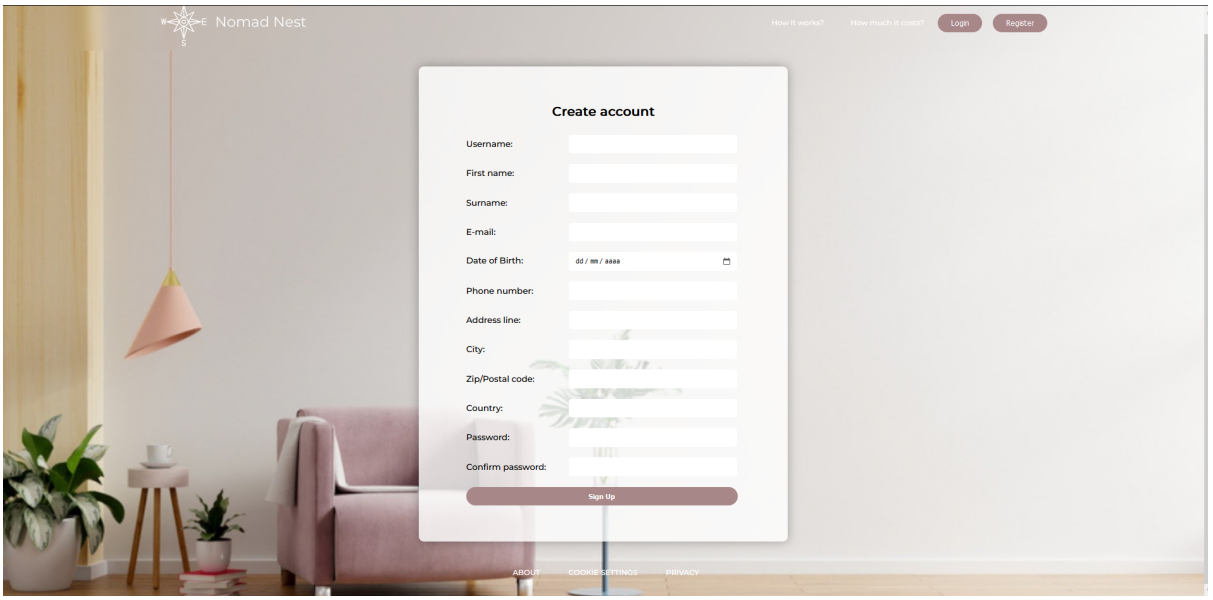
The architecture and design of the Home Exchange Manager application promotes modularity and maintainability. Each component focuses on specific responsibilities and interacts with other components in isolation. This separation of concerns allows independent development and testing of components, improves code reusability, and facilitates collaboration among team members.
By leveraging interactions between these software components, the Home Exchange Manager application provides a seamless user experience, efficient data management, and effective communication between users and administrators.

4.3. Project's design

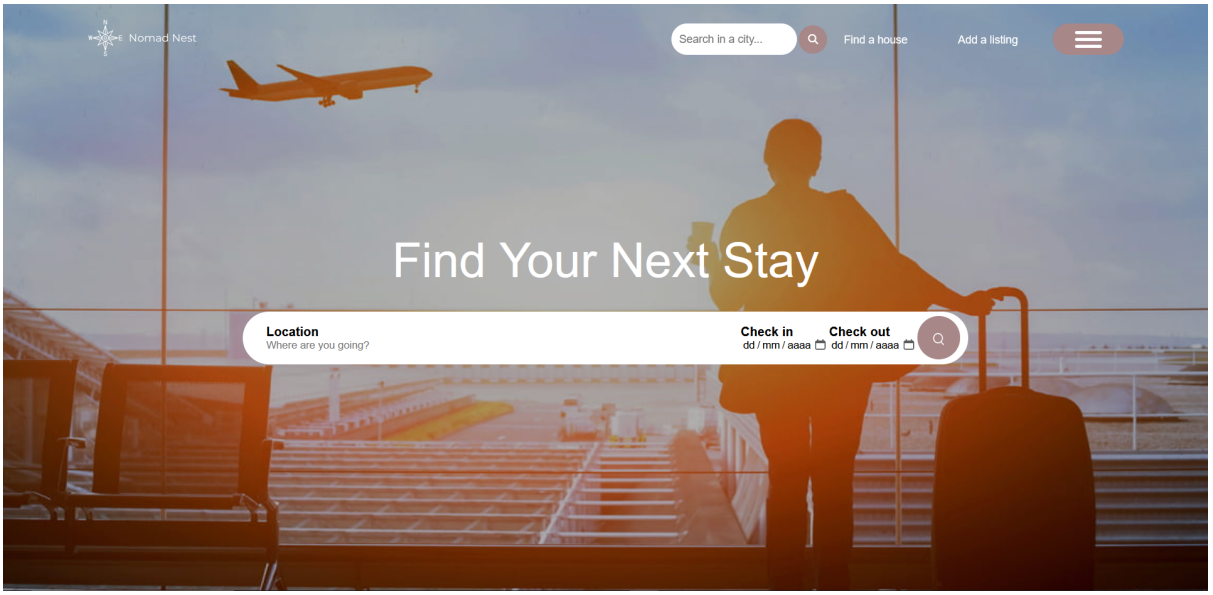
Below is presented the UI of the project.
Login Page:



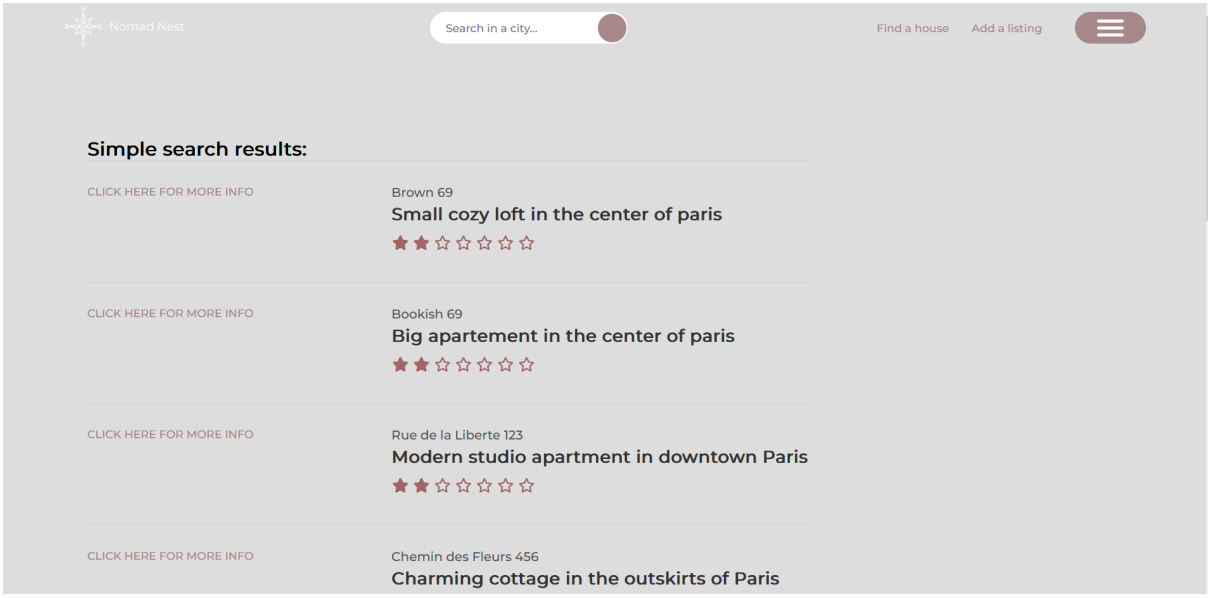
Register Page:



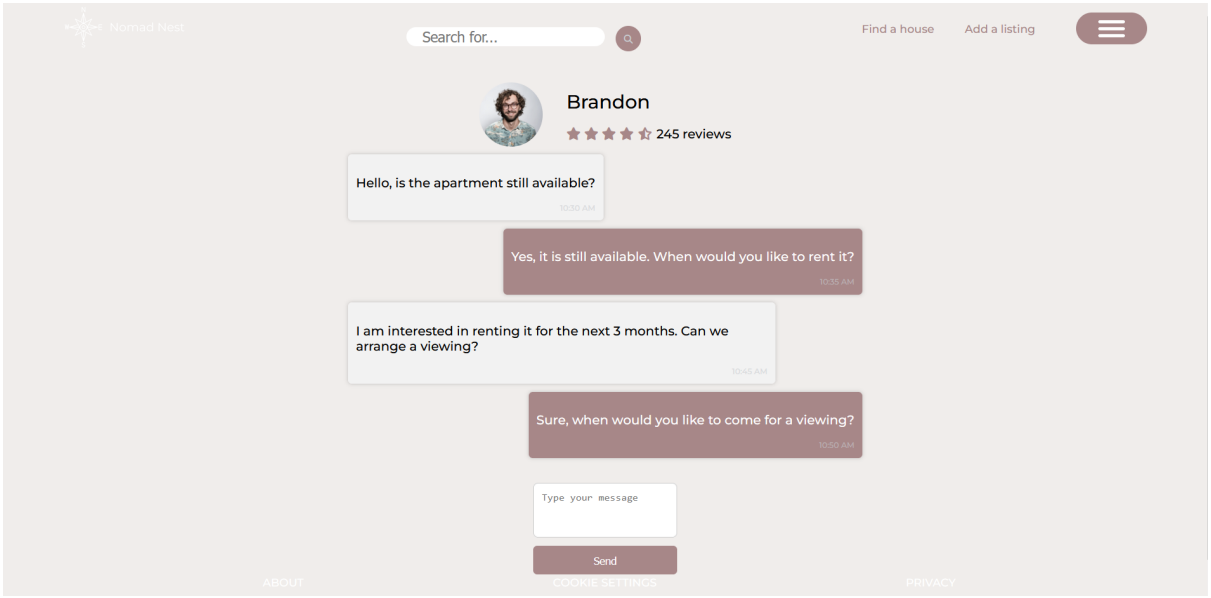
Home Page:



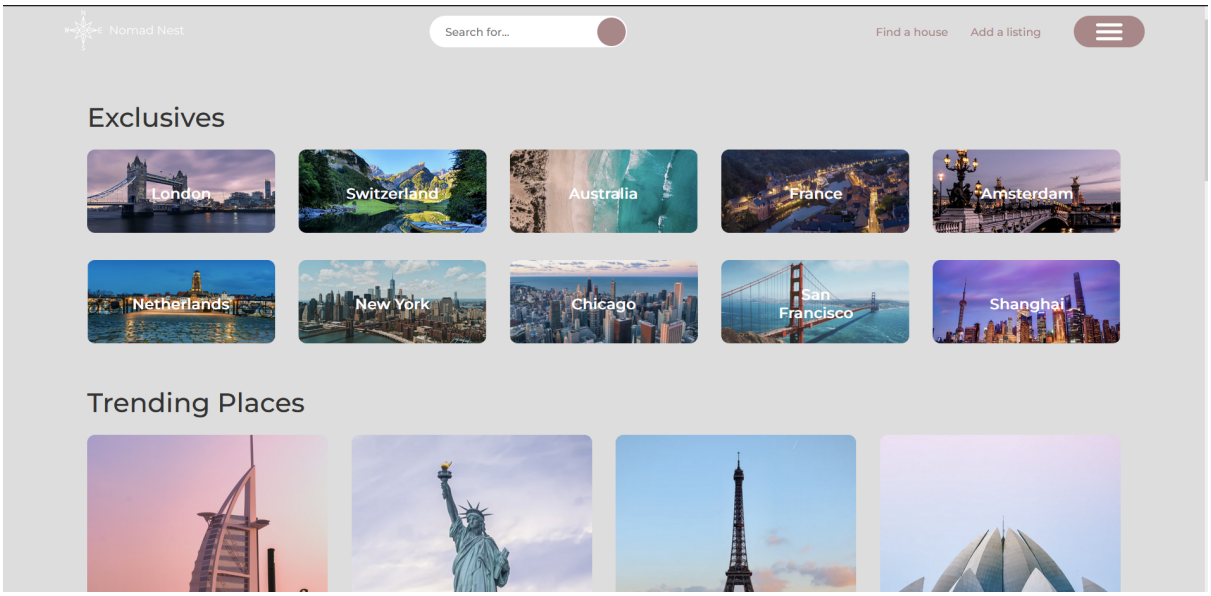
Listings Page:



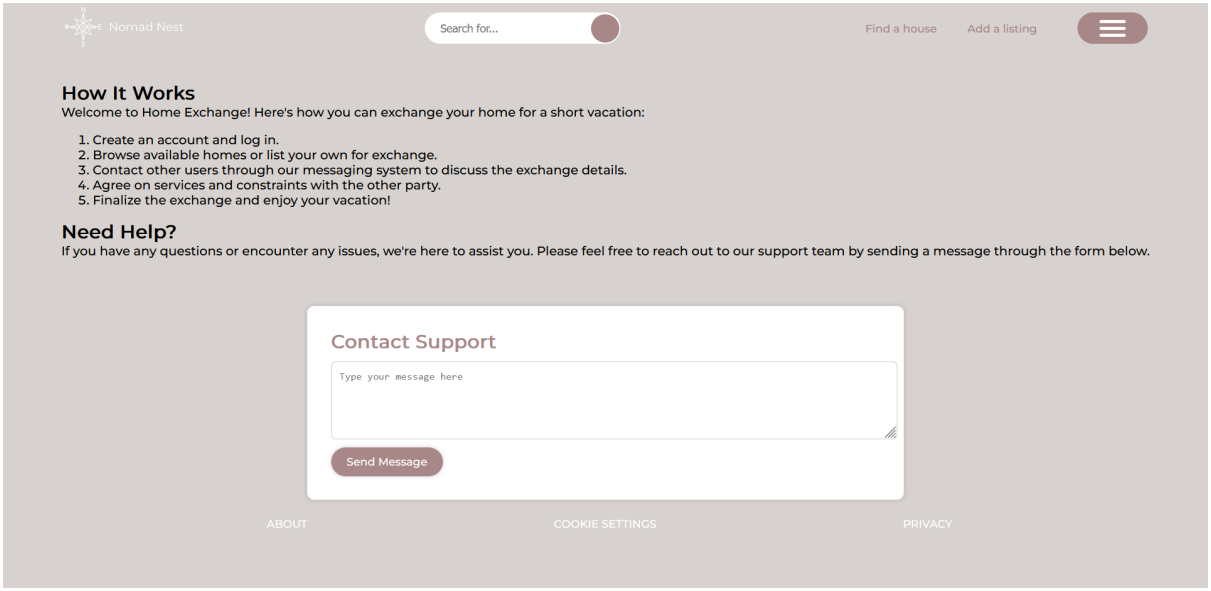
Chat Page:



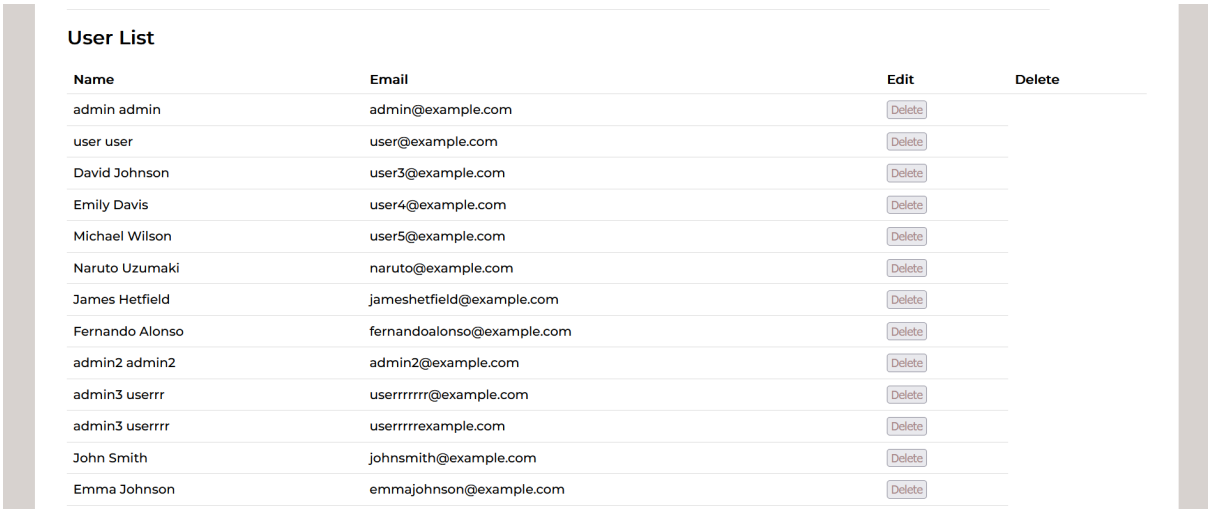
Favourites Page:



Help Page:



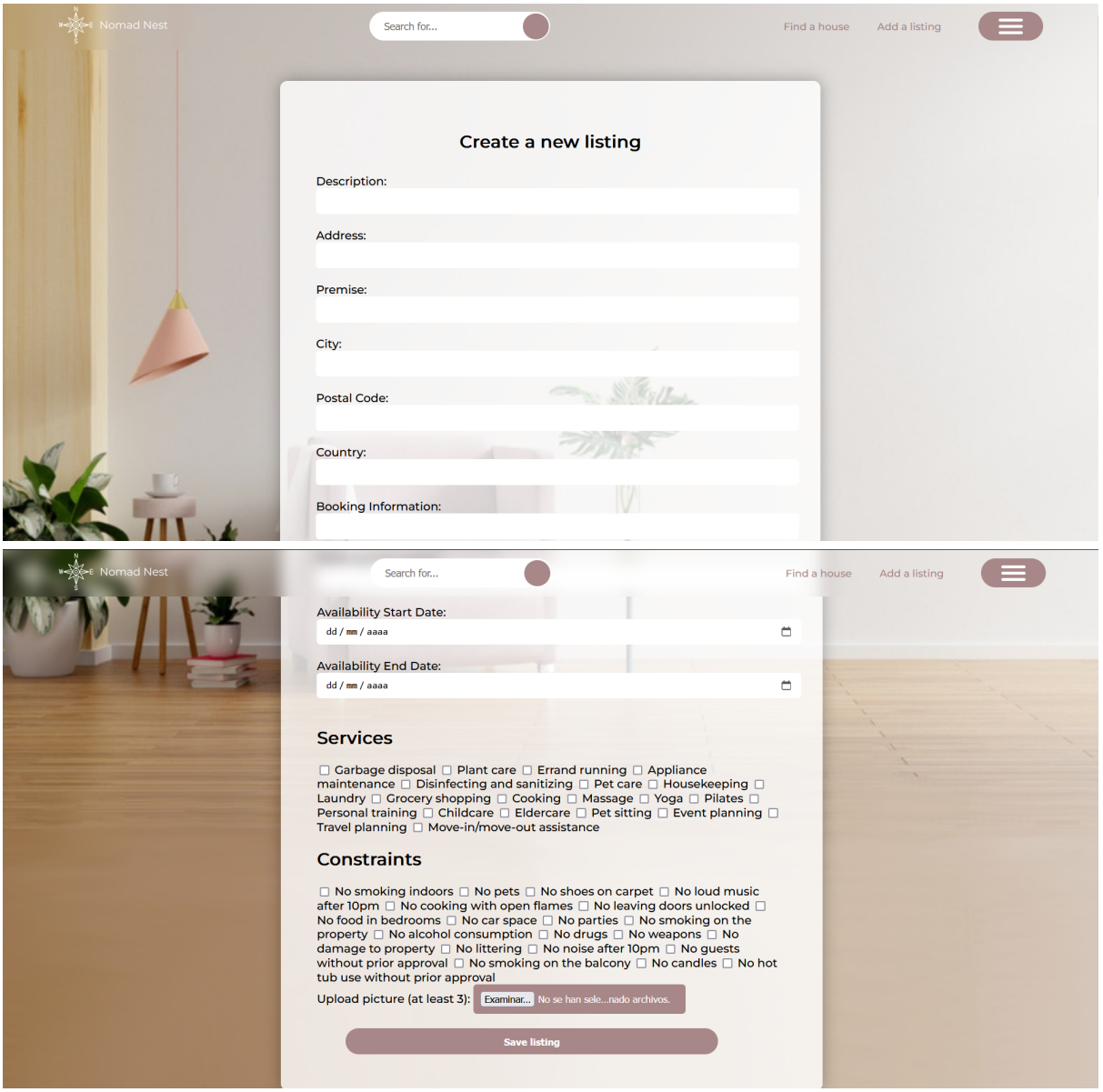
Admin Page:



User List

Name	Email	Edit	Delete
admin admin	admin@example.com	Delete	
user user	user@example.com	Delete	
David Johnson	user3@example.com	Delete	
Emily Davis	user4@example.com	Delete	
Michael Wilson	user5@example.com	Delete	
Naruto Uzumaki	naruto@example.com	Delete	
James Hetfield	jameshetfield@example.com	Delete	
Fernando Alonso	fernandoalonso@example.com	Delete	
admin2 admin2	admin2@example.com	Delete	
admin3 userrr	userrrrrrr@example.com	Delete	
admin3 userrrr	userrrrrexample.com	Delete	
John Smith	johnsmith@example.com	Delete	
Emma Johnson	emmajohnson@example.com	Delete	

New listing page:



4.4. Database design

The database design of the Home Exchange Manager application plays a crucial role in storing and managing the application's data. The chosen database management system is MySQL, which provides reliability, scalability, and excellent performance. The database design includes the following key entities:

1. **User:** Represents registered users of the application. The User entity stores information such as username, email, password, and profile details. It also maintains relationships with other entities, such as listings and messages.
2. **Listing:** Represents the homes available for exchange. The Listing entity contains attributes such as title, description, location, availability, services offered, and constraints imposed by homeowners. It is associated with the User entity to establish ownership.

3. **Role**: Represents a role or authority assigned to a user, defining their permissions and access levels within the system. It can be either Admin or regular user
4. **Booking:** Represents the booking requests made by users for home exchanges. The Booking entity stores information such as the guest user, host user, requested dates, and booking status. It enables users to request and confirm exchanges, while homeowners have the flexibility to accept or reject requests.
5. **Message:** Represents the communication between users. The Message entity stores attributes such as sender, receiver, timestamp, and message content. It allows users to discuss exchange details, clarify queries, and finalize arrangements securely within the application.
6. **Service**: Represents a service that can be associated with a listing, providing additional features or options.
7. **Constraint:** Represents a constraint or requirement associated with a listing, defining specific conditions or limitations.
8. **ListingRating:** Represents a rating or review given to a listing by a user, including the rating score and review comments.
9. **Message:** Represents a message sent between users, containing the sender, receiver, content, and timestamp.
10. **Image:** Represents an image associated with a listing, providing visual content for the listing.

The database design follows relational database principles, utilizing tables, primary keys, and foreign keys to establish relationships between entities. It ensures data consistency, integrity, and efficient querying.

5. Project management

5.1. Planning

The project followed a one-week sprint approach, where tasks were assigned to team members based on their skills and availability. Trello, a project management tool, was used to organize and track tasks. The Trello board had lists for different stages of development, and tasks were represented as cards with relevant details.

During sprint planning meetings, tasks were selected from the project backlog, estimated, and assigned to team members. The Trello board was updated with these tasks, providing visibility into the sprint's planned work.

Team members updated task statuses on Trello. This allowed real-time visibility into the project's state and facilitated collaboration.

The use of Trello and the one-week sprint approach provided an efficient and transparent way to plan, manage, and track project tasks, ensuring incremental value delivery. Here it is available the team´s trello board

5.2. Releases

First release was for the First Milestone day 15th of May in which the team focused on presenting and justifying the technologies used in the project. This involved explaining the rationale behind the choices made for the project's tech stack. The team also discussed the advantages of the selected technologies and how they contributed to the project's goals, scalability, security, and performance.

Second release was for the second milestone day 30th of May at this stage, the team presented a functional backend for the application consisting of 70% of the project. Where users could already be

registered and logged in and it was possible to search for listings both using the simple search only by city or the advanced search by city and availability date.

The third and final release will be presented on june 13th and will have all the functionalities described in this report, it will be the final version of the application.

5.3. Team members roles

- **Nizar Bikti:** Nizar has been the team leader, overseeing the project's progress and managing the team. He has worked on all backend operations involving the creation of the project, the database settings, handling requests, processing data, accessing the database, implementing business logic, integrating with external services, and securing the application. He was also responsible for a great part of the integration between front-end and backend, injecting the Thymeleaf tags into the HTML pages. As well as for testing the application and general troubleshooting.
- **Magdalena Pakula:** Magdalena has played a crucial role in frontend operations and design. She has been responsible for creating and implementing visually appealing user interfaces and ensuring a seamless user experience. In addition to frontend work, Magdalena has also contributed to backend tasks, supporting the development of certain functionalities such as dynamic and static mapping, security etc. She was also responsible for creating and providing weekly reports to track the team's progress and management of assigning tasks for team members.
- **Aleksandra Banasiak:** Aleksandra has primarily focused on frontend operations and design, working closely with Magdalena. She has been instrumental in crafting captivating and user-friendly interfaces, translating design concepts into functional components. Aleksandra has also contributed to some backend tasks, assisting with the development and integration of certain features.
- **Mario Aroca Páez**: Mario is a newcomer to web development. He has primarily focused on basic backend operations, involving tasks such as managing data, or implementing basic functionalities. Mario has been actively involved in providing weekly reports to track the team's progress and has also taken on the responsibility of preparing the final report.

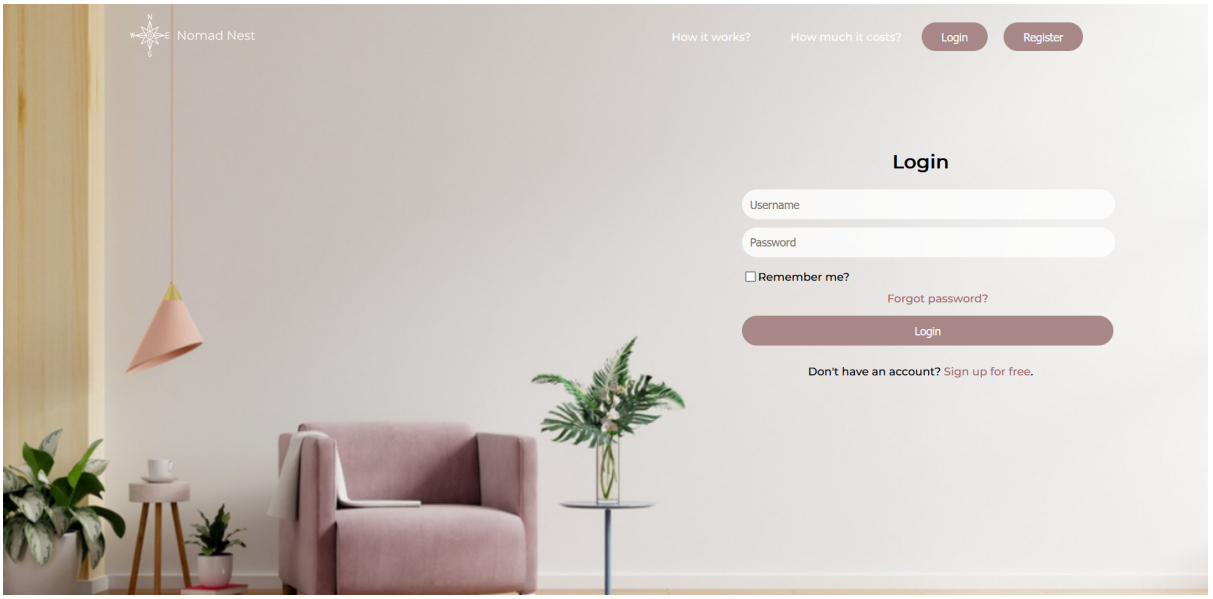
6. How to install

In order to install the Home Exchange application it is necessary to follow the next steps:

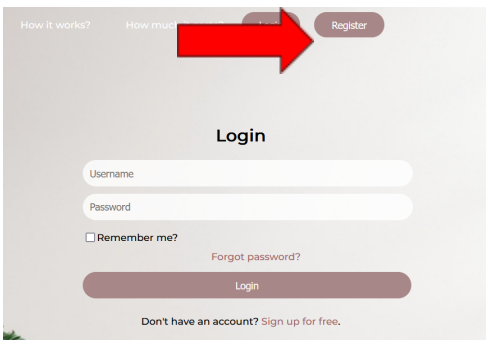
1. **Prerequisites:**
    a. Ensure that your system meets the requirements for running the web application, such as having Java Development Kit (JDK) version 17 or compatible installed.
    b. Install a MySQL database server, ensure it is running and create a database named: homexchangemanager
    c. Install Spring Boot by following the official documentation
2. **Obtain the Application Files:**
    a. Download or clone the project files from the designated source. https://github.com/nbebop/homeXchangeManager
    b. Place the project files in a suitable directory on your local machine.
3. **Build the Application:**
    a. Open your preferred Integrated Development Environment (IDE), such as IntelliJ IDEA or Eclipse.

b. Import the project into the IDE by selecting the project's root directory.
c. Allow the IDE to resolve dependencies and configure the project.
d. Build the application using the IDE's build or compile feature. This will compile the source code and download the required dependencies.

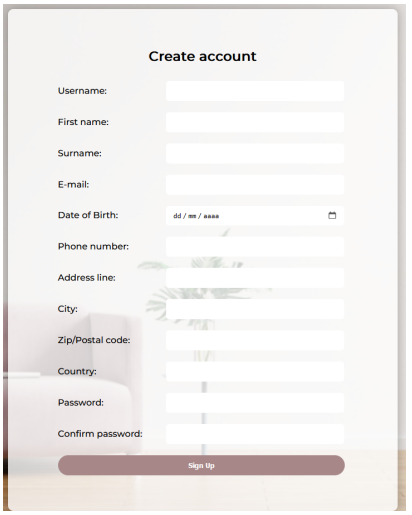
4. **Configure the Application:**
   a. Open the application.properties file located in the project's root directory.
   b. Verify or modify the following properties according to your setup:
      i. spring.datasource.url: Specify the URL for your MySQL database server, including the necessary configurations such as the host, port, and database name.
      ii. spring.datasource.username: Set the username for accessing the MySQL database.
      iii. spring.datasource.password: Set the password for the specified MySQL user.
5. **Start the Application:**
   a. In your IDE, build the application using gradle and run it
   b. Open a web browser and enter the following URL: http://localhost:8090 (If you did not change the port in the **application.properties** file)
   c. If the application is running properly, you should see the login page



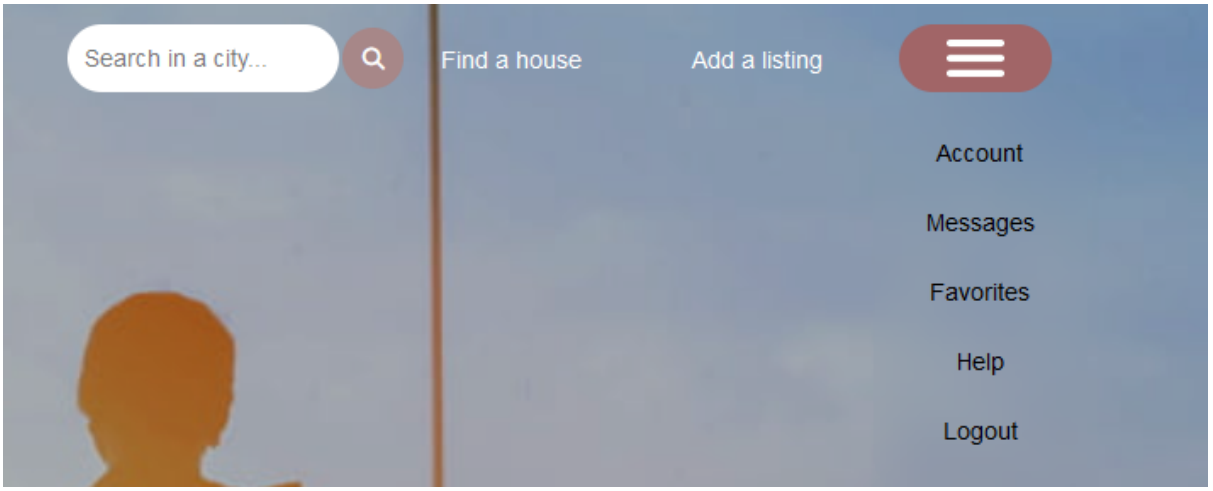
7. How to use



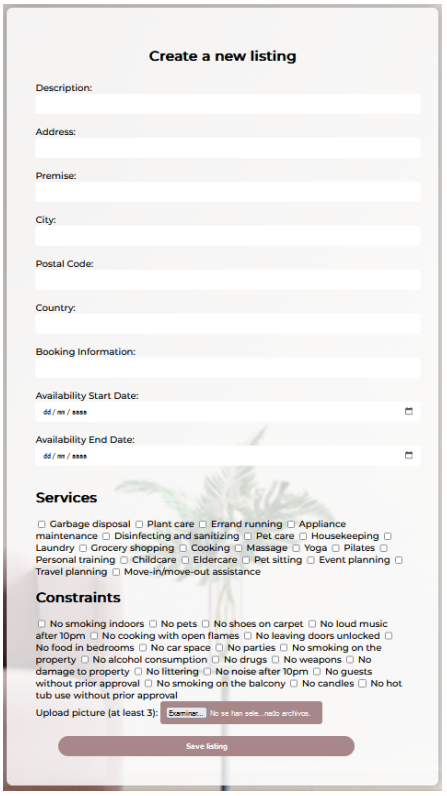
Upon opening the HomeXChangeManager application, users are greeted with a login screen. New users can easily create an account by providing their personal information being their Username, First name, Surname, e-mail, date of birth, phone number, address and setting up a secure password. Once logged in, users have access to a range of features and tools.



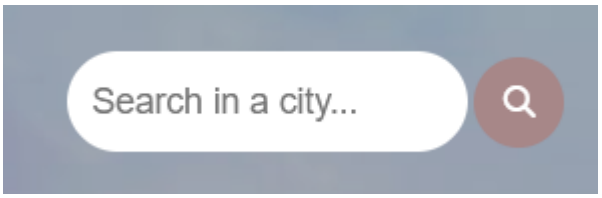
The application's home page serves as the central hub for managing home exchanges. From it, users can view and manage their profiles, browse available home exchange listings, and communicate with other users.



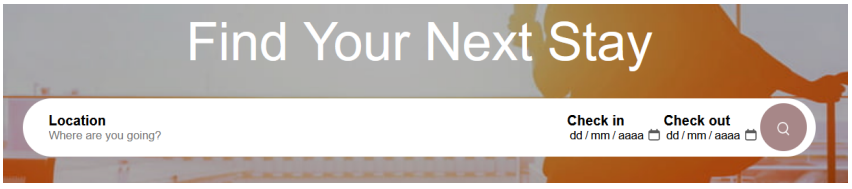
To get started, it is important for users to add their own listing so they have their home uploaded in the webpage in order to perform the home exchange with other users, the information they will have to provide about their home is a description, the address, some booking information, the date where the availability starts and the date when it ends, also services to be done in the house and constraints can be included and Finally pictures of the house must be included, at least 3 of them.



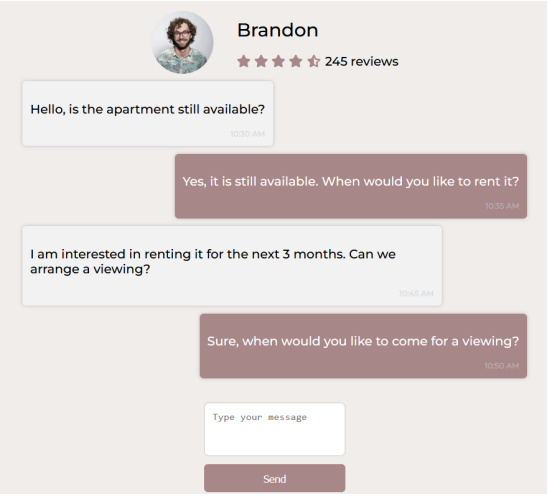
When searching for a home exchange, users can navigate to the listings section. Here, they can browse through a database of available properties. First there is a simple search option where users will search just based on location. They will type the name of the city where they want to go and will get all the available listings in that city.



Also an advanced search option is available where users will search listings based on the location where they want to go and the availability of the listings. They will insert the name of the city where they want to go and their arrival and departure date for this listing



Once a potential listing is found, users can initiate communication with the homeowner through the messaging system provided within the application. This allows users to discuss details, negotiate terms, and address any questions or concerns they may have. The messaging system ensures that all communication remains centralized and secure.



After finalizing the details of a home exchange, users can proceed to the agreement and confirmation stage. They will send a booking request to the homeowner and he will be able to either accept it or refuse it.

Once the home exchange is complete, users are encouraged to leave a review and rating for the other party. This helps build trust within the HomeXChangeManager community and provides valuable feedback for future users.

8. Conclusions

Undertaking the Home Exchange Manager project has been an exhilarating journey that has presented various challenges. The limited timeframe and unforeseen difficulties along the way have tested the team's skills, resourcefulness, and collaboration. However, despite these challenges, the team persisted and emerged stronger, having developed a remarkable solution.

From the outset, time constraints posed a significant hurdle. The team faced an ambitious project scope with a tight deadline, necessitating adaptation and streamlining of the development process. Strategic decisions were made to prioritize essential features and maximize efficiency. Despite the pressure, the team maintained a positive mindset and remained focused on delivering the product.

Technical challenges were encountered along the way, demanding creative problem-solving. The team had to navigate through complex situations, especially because of the difference in background and knowledge between team members. They brainstormed innovative solutions, conducted research, and leveraged their collective expertise to overcome these hurdles. Each obstacle served as an opportunity for growth, enabling the team to expand their knowledge and refine their skills.

Coordination and collaboration within the team were not without difficulties. With multiple stakeholders and diverse responsibilities, effective communication and coordination became crucial. Instances of misalignment and conflicting viewpoints arose, but the team approached these situations with open minds and a genuine willingness to understand one another. Through active listening, respectful discussions, and compromise, an environment of collaboration and mutual support was fostered, ultimately strengthening the team's bond

8.1. Learned lessons

1. **Importance of User-Centric Design:** Throughout the development of the Home Exchange Manager application, we learned the significance of prioritizing user experience and designing the application with the users' needs in mind. By focusing on creating a

user-friendly platform, we were able to enhance usability, attract more users, and improve overall satisfaction.

2. **Effective Project Planning and Management:** The project management aspect of the Home Exchange Manager application taught us the importance of thorough planning, setting realistic goals and assigning clear roles and responsibilities to team members. Adhering to a well-defined project plan helped us stay organized, meet deadlines, and ensure smooth collaboration within the team.
    a. There have been problems with organization concerning meeting schedules. Despite the effort, the team was not able to find a common time in order to discuss the project. Therefore, we resorted to meetings during class hours and to communicate through MS Teams. Looking back, we should have met more often, considering that the sprint schedule was weekly.
    b. Separation of work and skill assessment was a big concern during the project development. After having separated the frontend and backend tasks and let the teams work independently from each other, we did not manage to clearly define early who was responsible for the integration, which resulted in a delay of the delivery of the presentation during the semester. It is important to note that some members encountered difficulties and were not able to fulfill their share of the project.
3. **Collaboration and Version Control**: The use of collaboration tools, particularly GitHub, proved invaluable in facilitating seamless teamwork and effective version control. It allowed team members to work simultaneously, track changes, and resolve conflicts efficiently, ultimately enhancing productivity and ensuring code integrity.
4. **Testing and Quality Assurance**: Implementing a comprehensive testing strategy and conducting regular quality assurance checks were instrumental in identifying and resolving bugs, improving application performance, and ensuring a smooth user experience. We learned the significance of thorough testing at each stage of development to deliver a reliable and stable application.

8.2. Perspectives

1. **Continuous Improvement and Updates**: The Home Exchange Manager application can benefit from continuous improvement and regular updates based on user feedback and emerging technologies. By actively listening to user suggestions and incorporating new features and enhancements, we can ensure that the platform remains competitive and meets the evolving needs of its users.
2. **Expansion and Scaling:** As the Home Exchange Manager application gains popularity and attracts a larger user base, there may be opportunities for expansion and scaling. This could involve exploring partnerships with other home exchange platforms, expanding to new geographical regions, or introducing premium features for monetization.
3. **Community Building and Engagement:** Fostering a strong community of users is crucial for the success of the Home Exchange Manager application. Encouraging user engagement, facilitating communication and collaboration among users, and organizing events or meetups can help build a vibrant and active user community.
4. **Integration with External Services**: To enhance the functionality and user experience of the application, integrating with external services such as online mapping APIs, payment gateways, or social media platforms can be considered. These integrations can provide additional value to users and make the platform more versatile.
5. **Mobile Application Development:** As mobile devices become increasingly popular for accessing web applications, developing a dedicated mobile application for the Home

Exchange Manager platform can be a promising avenue for growth. A mobile app would provide users with greater convenience and accessibility, expanding the reach of the platform.