Software Architecture Document

Housing App

Semester 3

Rositsa Nikolova

Version Table

Version	Date	Author	Description
1.0	06.10.2022	Rositsa Nikolova	Document Created

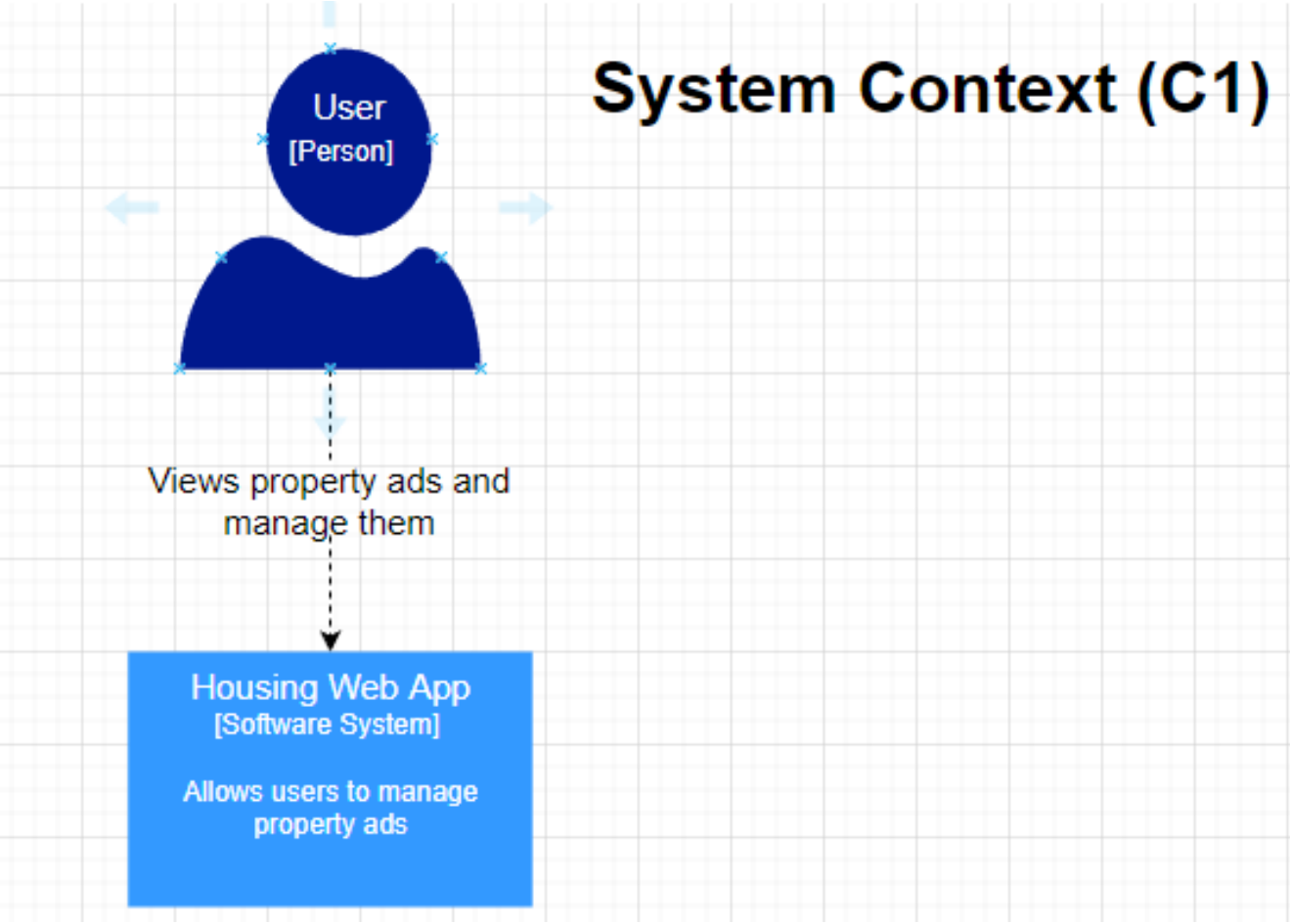
Table on contents

Document overview

This document aims to provide information about the software architecture and the technology choices for a software system for managing rental properties. The intention of this document is to determine the system will be structured at the highest level. The technical teachers can use this document to validate that the project is meeting the requirements and to understand how the system is going to work.

System Context diagram

The Housing App is not going to depend on any external systems. The system supports 3 user roles - – administrator, homeowner and renter, each one with a different purpose, consequently rights. For the goals of the project the software system is going to be self-sufficient, all functionalities could be created internally and there is no need for integration with external systems.



Container diagram

Explanation: Why two separate applications

The housing software system is going to consist of two containers. For the backend part - a Java Spring Boot application, which is going to handle business logic, retrieve data from the database, writes data to the database and serves as a REST API. The front end is going to be created with the JavaScript based library – React. The two applications will communicate via HTTP requests where the data will be delivered in JSON format. The main purpose of creating two separate applications – containers is that it gives flexibility of the project: the front-end part is independent from the back-end part, which means the back-end part can serve a second or a third front. Therefore, it gives a good foundation for extensibility.

Explanation: Choice of programming language for the back-end application

The Java programming language was the choice for this software system because firstly, it's very similar to C# which will make the learning process faster. Secondly, it's one of the most used programming languages nowadays which guarantees there will be detailed documentation and available materials on the Internet which will make solving problems way easier. Third, it's platform-independent which gives a lot of flexibility to switching Operating Systems because the code will run everywhere which reduces the project constraints.

Finally, there are a lot of available prebuild tools and libraries which are going to make the development of the application easier and faster like Gradle – build automation tool and Lombok – library for reducing boiler-plate code.

Explanation: Choice of framework for the back-end application

The Java Spring Boot framework is going to be used first because it takes care of many configurations and dependencies needed for the app to run that otherwise should be done manually so more time can be spent working on the project. Secondly, it has an embedded server – Tomcat which makes testing the application easier in the development process.

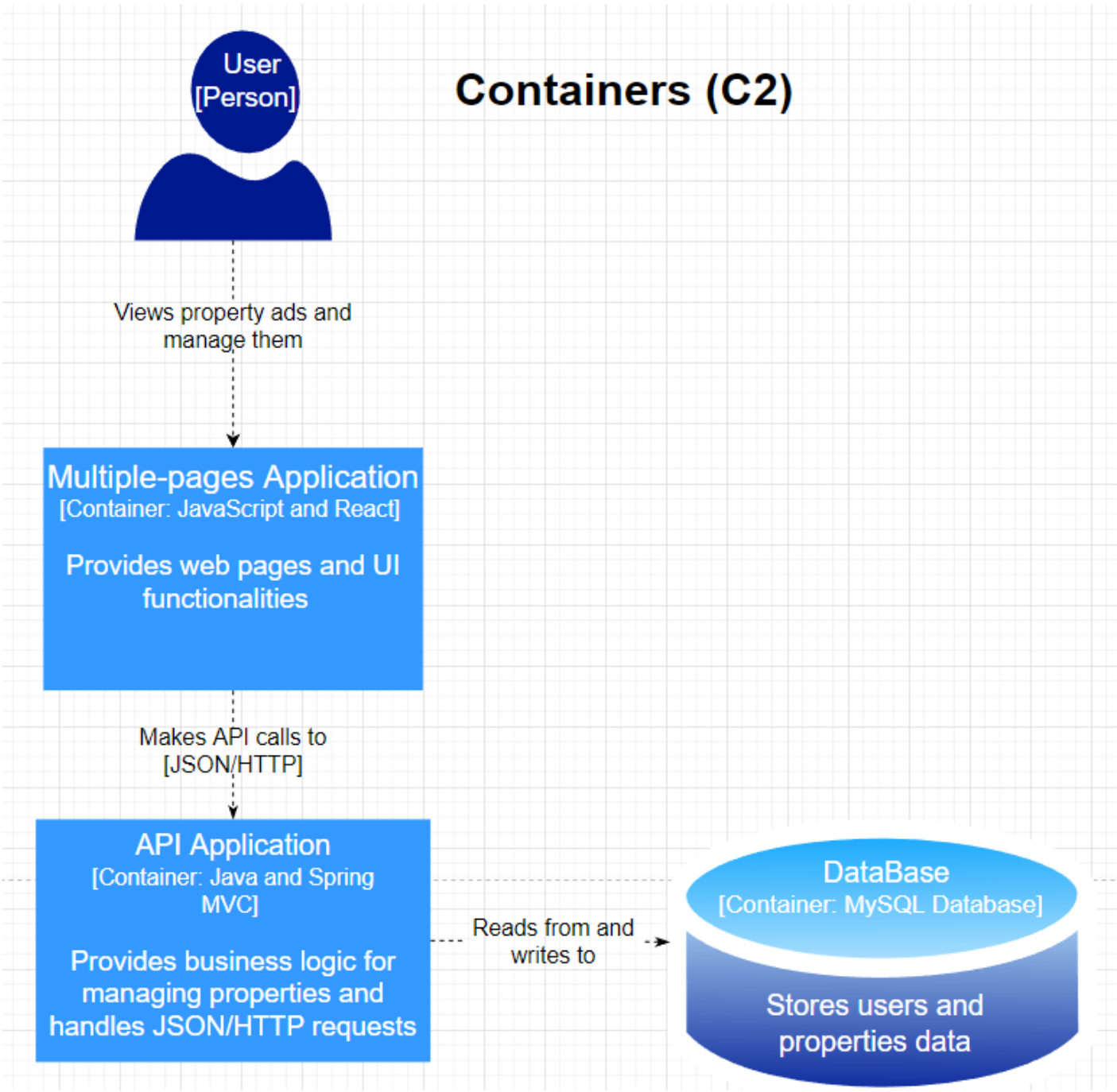
Explanation: Choice of technologies for the front-end application

The React library allows us to create more complex dynamic user-friendly user interfaces fast. First, its component–based character allows us to break up the application into small independent reusable pieces of code – components. The separation of concerns is by functionality not by technology (HTML, JavaScript, CSS). Therefore, it reduces the amount of code and speeds up the process of development. Secondly, it renders only that should be rendered and not the whole page every time something changes which makes it efficient.

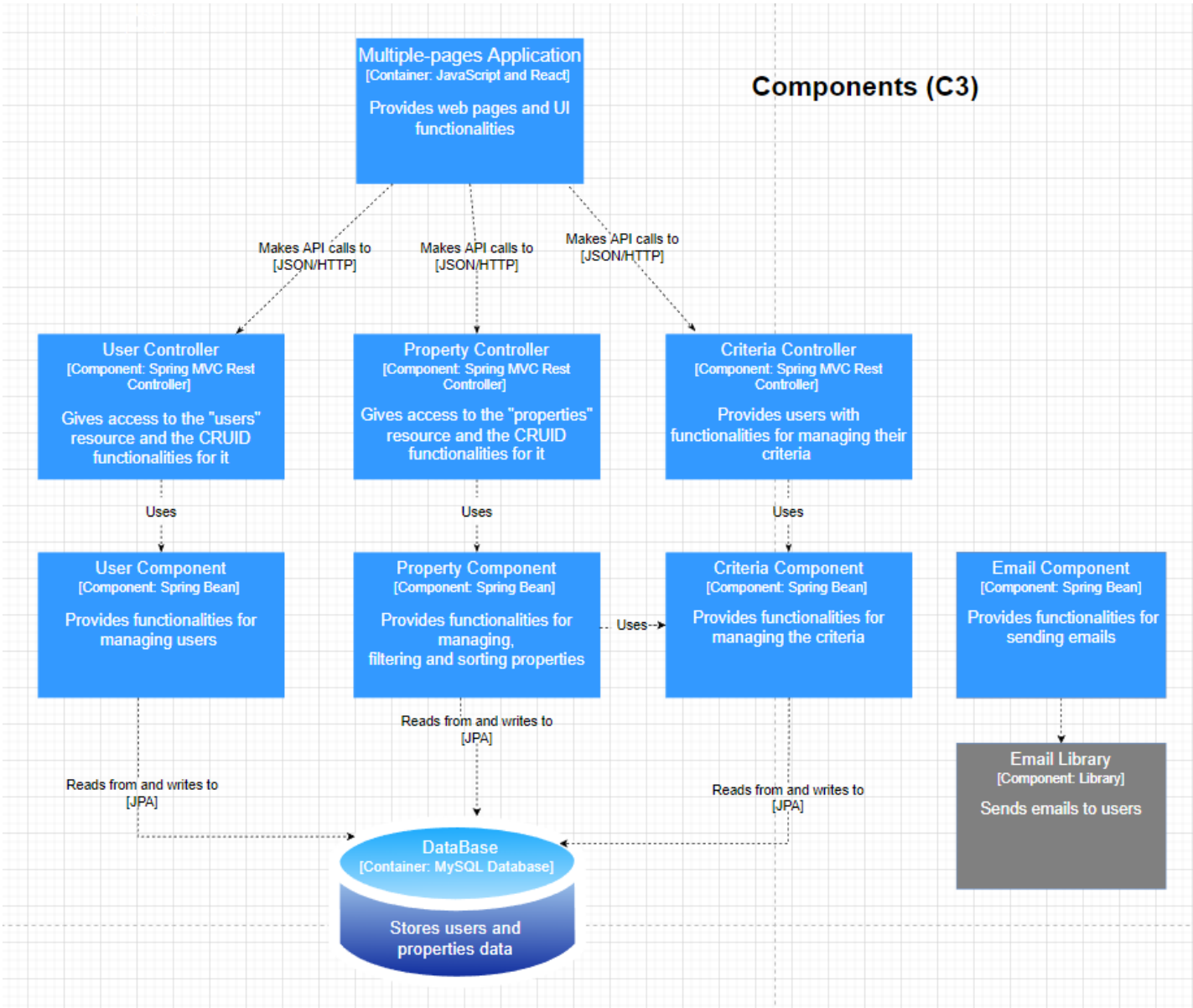
Explanation: Choice of database

The main benefit of a relational database is that it provides an intuitive way to structure the data and it's very easy to retrieve it by simple SQL queries. MySQL is the choice for RDMS because I am already familiar with it.

Containers (C2)



Component diagram



The back-end application is going to consist of several rest controllers – User Controller, Property Controller and Criteria Controller. They are going to define the endpoints of the REST API. The main resources in the application are going to be users, properties and the criteria of the

users. The application should be able to send email to users so that the Email Component will make use of an external library to handle this task.

Architecture decisions

The client-server architecture

The front end React application (the client) is going to make calls to the REST API (the server). In that way we accomplish separation of concerns and modularity. The system is also not going to be heavily coupled and directly dependent.

Key benefits:

Modularity

Maintainability

Extensibility

DTO design pattern

Data Transfer Objects are going to be used to not expose sensitive data directly to the client but instead only the required information is going to be sent to the client. The conversion is going to be done with the Model Mapper library to remove the amount of boilerplate code.

Key benefits:

Security

Dependency inversion

The dependency inversion is going to be accomplished by making use of interfaces which are going to be injected into the classes that use them. In that way we force the classes to rely on abstractions rather than concrete implementations.

Key benefits:

SOLID – Dependency inversion principle

Flexibility

Extensibility

Layered architecture

The layered architecture is going to be used in the back-end application. The layers are going to be the Controllers Layer, Business Layer and Data Access Layer. In that way we accomplish separation of concerns because each layer takes care only of what is responsible for. There is reduced dependency as well because the function of each layer is separated from the other layers. Testing is easier because of the separated components and each component can be tested individually.

Key benefits:

Separation of concerns

Readability

SOLID – single responsibility principle