

|  |  |
| --- | --- |
| DESIGN DOCUMENT | |
|  |  |
| CALVIN KWAN |  |

Contents

[1. Spring Boot 3](#_Toc146017323)

[1.1 What is Spring Boot? 3](#_Toc146017324)

[1.2 Why using Spring Boot? 3](#_Toc146017325)

[2. React 4](#_Toc146017331)

[2.1 What is React? 4](#_Toc146017332)

[2.2 Why using React? 4](#_Toc146017333)

[3. MySQL 5](#_Toc146017331)

[2.1 What is MySQL? 5](#_Toc146017332)

[2.2 Why using MySQL? 5](#_Toc146017333)

[3. SOLID Architecture 6](#_Toc146017334)

[3.1 Single Responsibility 6](#_Toc146017335)

[3.2 Open/Closed 6](#_Toc146017336)

[3.3 Liskov Subtitution 6](#_Toc146017337)

[3.4 Interface Segregation 6](#_Toc146017337)

[3.5 Dependency Inversion 6](#_Toc146017337)

[4. Diagrams 7](#_Toc146017338)

[4.1 C4 Diagram 7](#_Toc146017337)

# Spring Boot

## 

## What is Spring Boot?

Spring Boot is an open-source Java-based framework designed to simplify the development of production-ready, stand-alone, and web-based applications. It is part of the larger Spring Framework ecosystem, which provides comprehensive support for building enterprise-level Java applications. Spring Boot builds upon the Spring Framework, offering several advantages over other backend frameworks.

## Why using Spring Boot?

1. **Built-in Web Server**

Spring Boot includes an embedded web server ( such as Tomcat, Jetty, or Undertow ) by default. This can simplify deployment to a separate web server and can run as a standalone application.

1. **Rapid Development**

Spring Boot provides set of conventions and defaults so developers are able to set up the application quickly since configurations process can be reduced a lot so the development increases a lot.

1. **Spring Initializr**

Spring Boot offers a web tool called “Spring Initalizr” which allows software developers to generate a new Spring Boot Project quickly with the needed dependencies and configurations.

1. **Auto-Configuration**

Spring Boot use “auto-configuration” principle to automatically configure application component based on dependencies in the classpath which eliminates the need to manually configure the application in many cases

# React

## What is React?

React is an open-source JavaScript library for building user interfaces. It was developed and is maintained by Facebook and a community of individual developers and companies. React is often used to build interactive and dynamic web applications, and it offers several advantages over other frontend frameworks and libraries.

## Why using React?

1. **Content-Based Architecture**

React is built around a component-based architecture, thus allows developers to break down the user interface into several components which promote code reusability and maintainability.

1. **Virtual DOM**

React uses a virtual DOM ( Document Object Model ). That means React does not directly modify the actual DOM. React updates the virtual representation of the DOM and only re-render when necessary which results in faster rendering.

1. **Unidirectional Data Flow**

React enforces a one-way data binding that will make developers understand the data changes that affect the application’s state and UI easier.

1. **JSX**

React use JSX file which allows developers to write a HTML-like code within the javascript. This allows developers to express component hierarchy and provides a clear visualization of the UI structure.

1. **Compatibility to Other Libraries**

React can be easily be integrated with other frontend library such as Bootstrap and Tailwind, allowing developers to create a frontend application according to libraries that they know better.

# MySQL

## What is MySQL?

MySQL is an open-source relational database management system (RDBMS) that is widely used for managing and organizing data. It is part of the MySQL relational database software stack often referred to as LAMP (Linux, Apache, MySQL, PHP/Python/Perl). MySQL is developed, distributed, and supported by Oracle Corporation.

## Why using MySQL?

1. **Relational Database Management System (RDBMS)**

MySQL is a Relational Database Management System which allows storing data in the form of tables with rows and columns. It is simplified and more suitable form for the Huister Web Application since it doesn’t store data with a lot of types.

1. **Scalability**

MySQL is scalable and has the ability to handle large amount of datasets and huge traffic. It also support data replications across servers, load balancing, etc.

1. **Ease of Use**

MySQL is easy to setup and use. MySQL provides developers a standard SQL interface so developers and database administrators can easily familiarize themselves with the database as long as they are experienced with relational database.

1. **Transaction Support**

MySQL has support for transaction. This ensured the consistency and integrity of the stored data. This is a great feature for Huister Application since it requires ACID ( Atomicity, Consistency, Isolation, Durability ) properties

# SOLID Architecture

1. **Single Responsibility**

This principle means that a class should have only a single job or responsibility. This principle is applied in the Spring Boot Application which use use cases at the business layer. A single use case only meant to do a single job only.

1. **Open/Closed**

Open/Closed principle mandate that an entity should be extendable but not modifieable. That means whenever a new feature is implemented, the previous codes cannot be modified. So, techniques like inheritance, abstractions, or interfaces to increase functionality. This principle is applied in the Spring Boot API Application where each user stories are only connected to a single use case, which doesn’t allow modifications to existing code.

1. **Liskov Subtitution**

This principle asserts that if a child class is a subtype of its parent class, it should be able to function like its parent class without causing errors in the program. Currently, this principle has only been applied to the error handling exception which will be sent as a response in case there is a bad request from the client that is happening in the server.

1. **Interface Segregation**

This principle tells us that instead of creating an interface that handle a lot of methods, we should divide those methods into several interfaces so classes do not need to override unused methods.

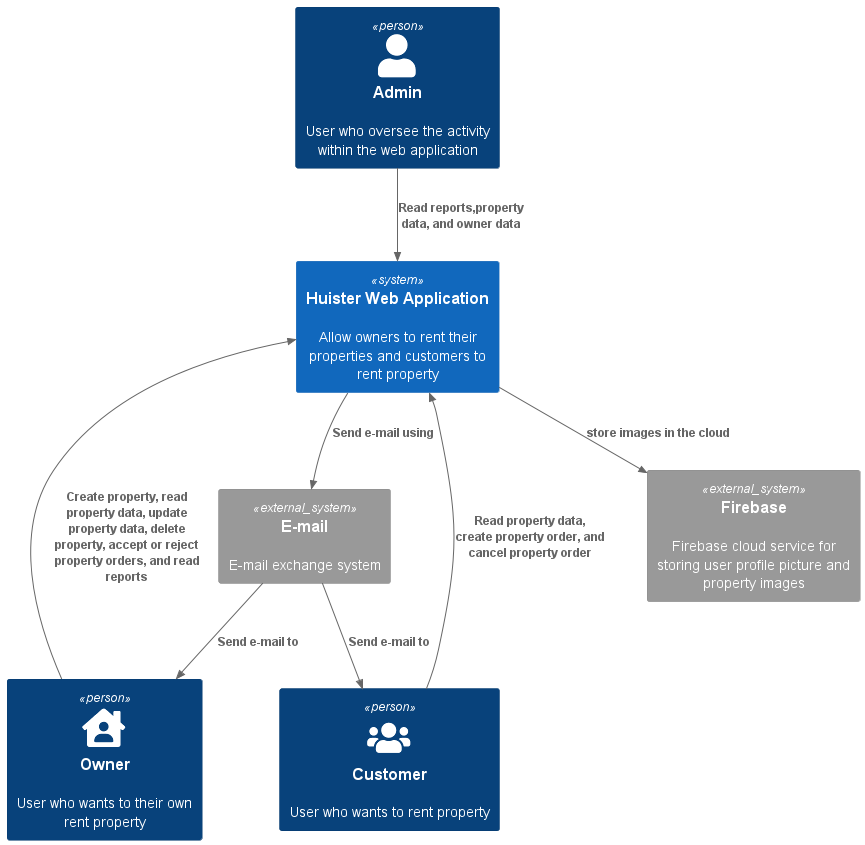
1. **Dependency Inversion**

Dependency inversion emphasizes the need of abstraction using abstract class or interfaces to decouple high-level modules to lower-level modules. This principle is implemented in the business layer which allows easier testing and greater flexibility.

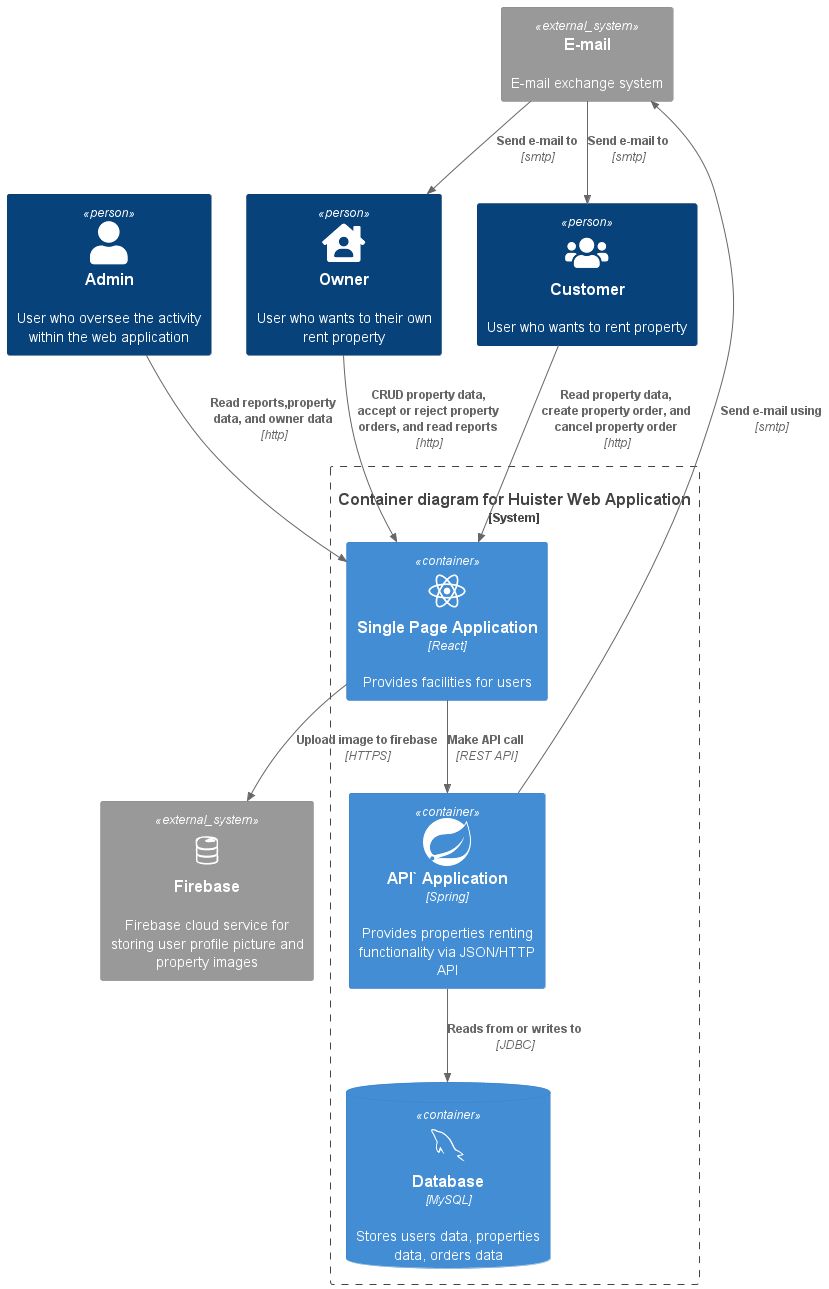
# Diagrams

## C4 Diagrams

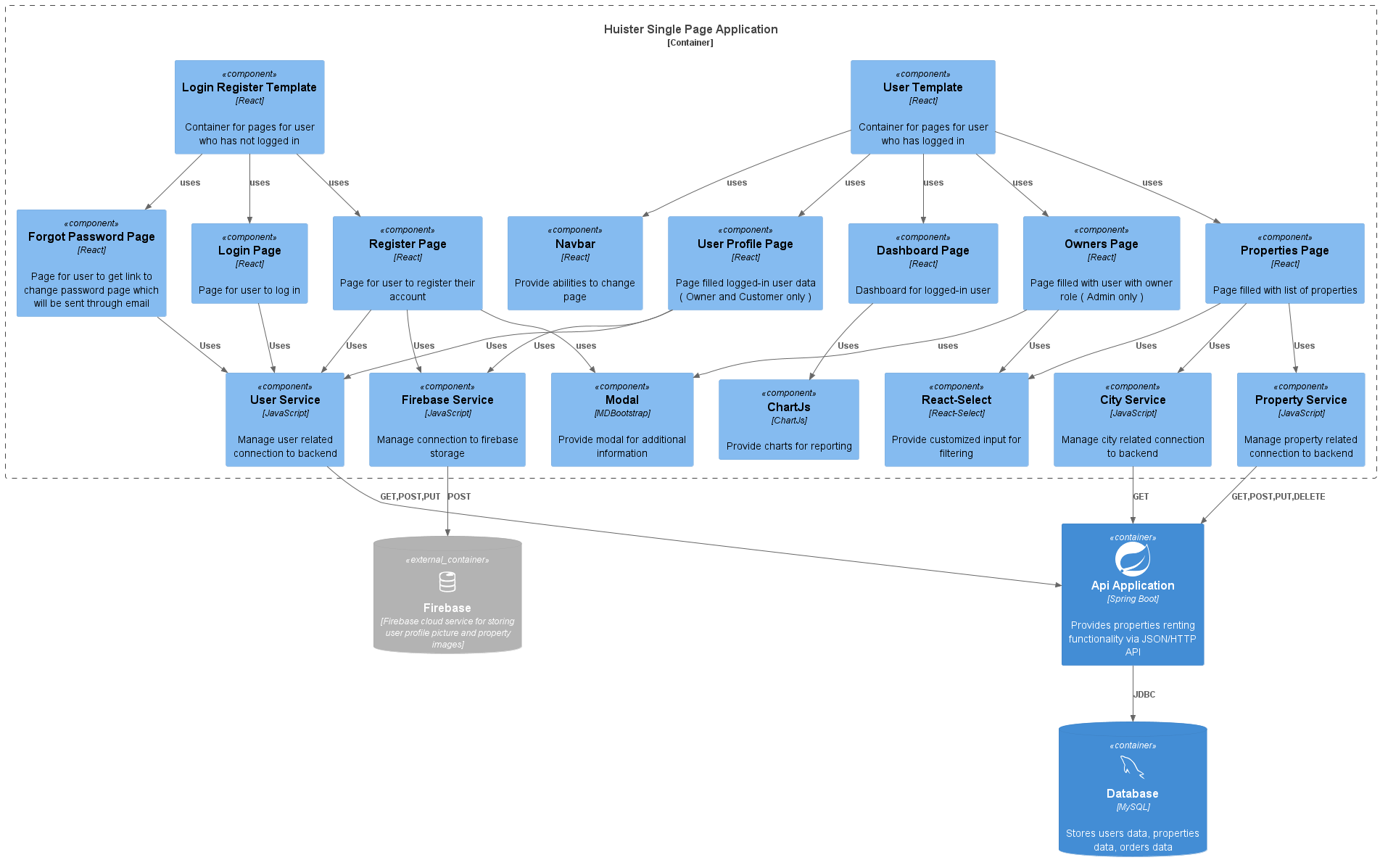
1. **Context**



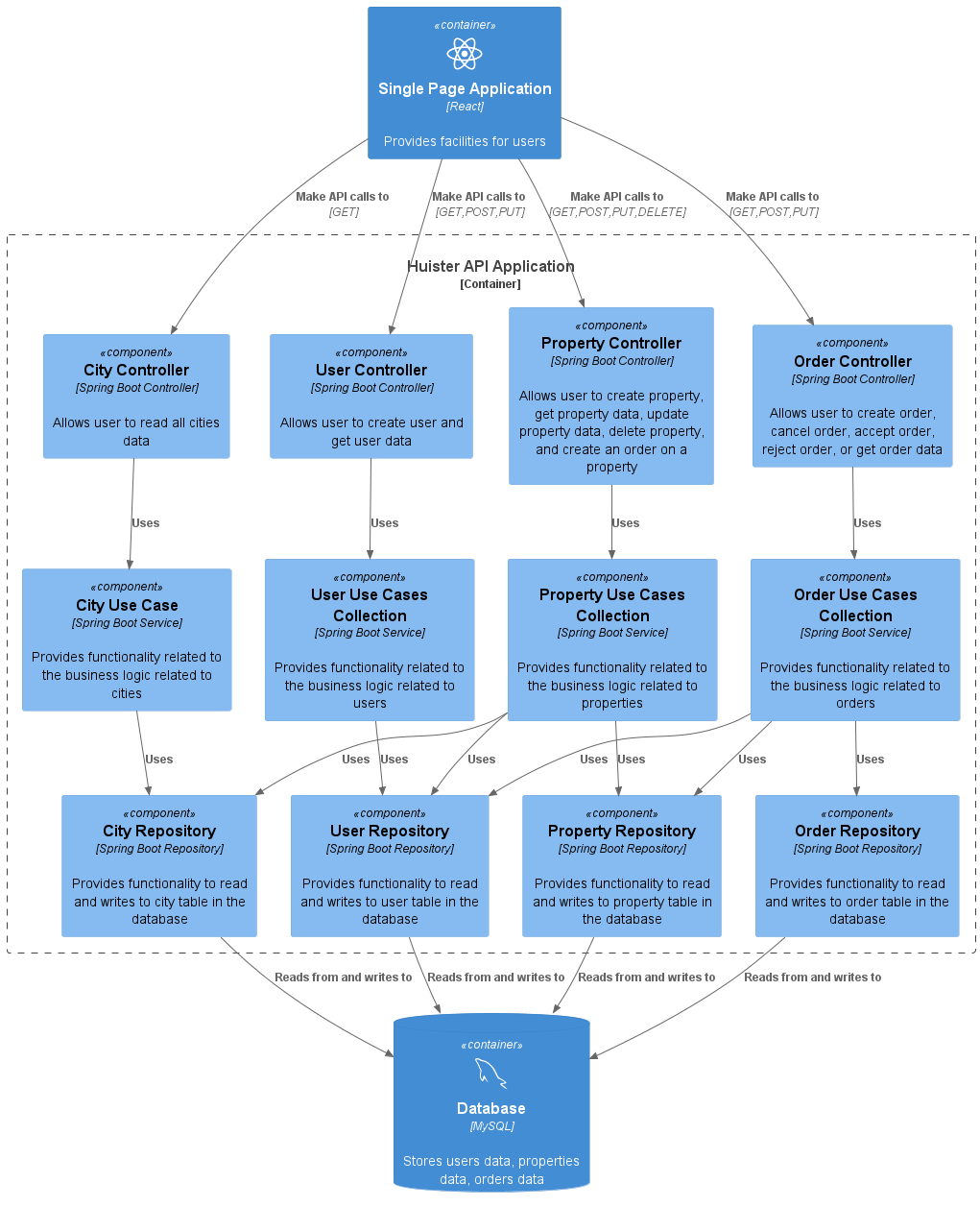
1. **Containers**

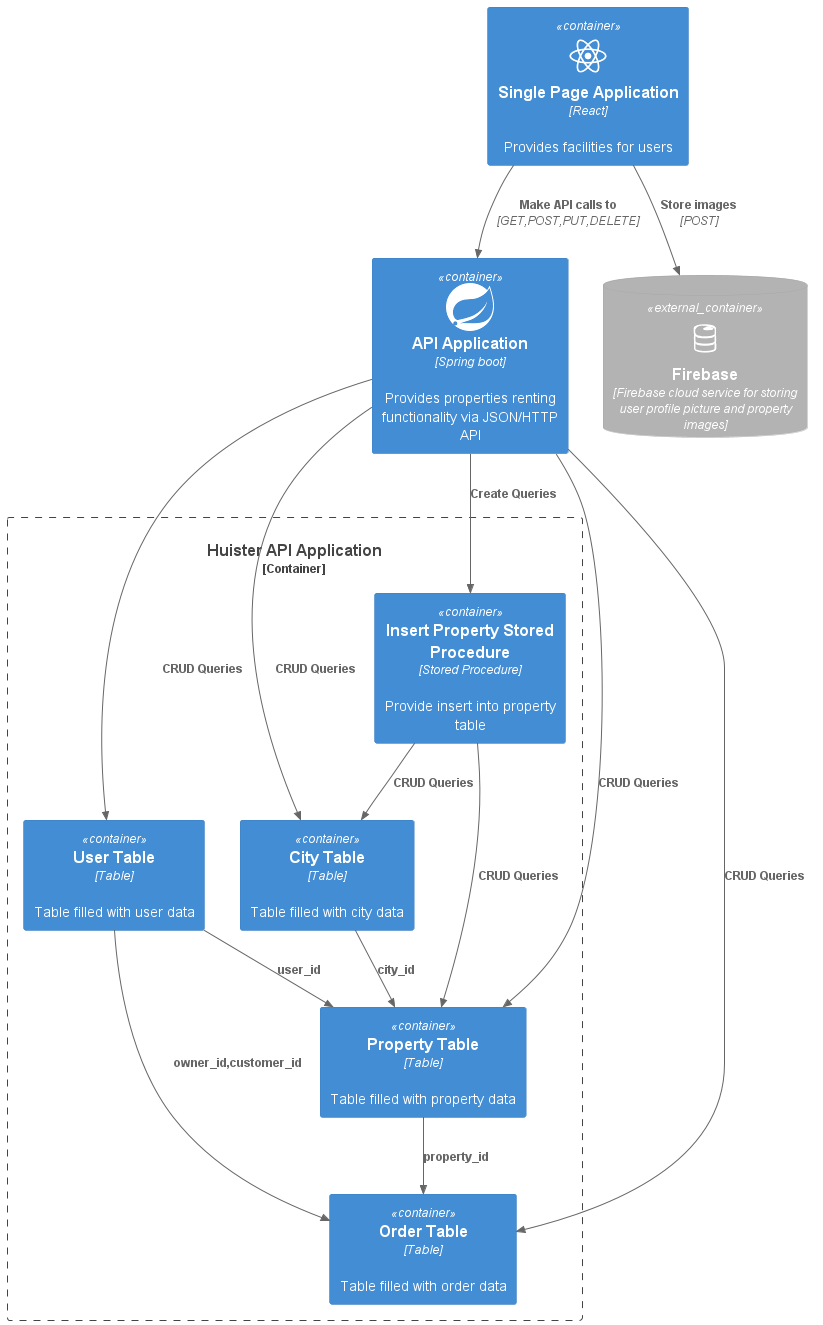


1. **Components – Frontend**



1. **Components – Backend**



1. **Components – Database**
2. **Code**

