ESTATE LAND WEBSITE  
DESIGN DOCUMENT  
  
 Ana NITU

October 5, 2021

Contents

1. Introduction to the project………….……………………….2
2. SOLID Principles and applicability…………………………2
3. React frontend………………….…………………………..3
4. Java Spring boot backend………………………………….3
5. C4 Diagrams…………..…………………………………..4

# Introduction TO THE PROJECT

The project consists of a complex real estate agency web application, where the user is able to view a wide range of properties, for rent or sale, filter them, view more details about a certain property, and, if wanted, contact the agent in order to plan a viewing. It is also possible to search the properties, to view them by map (location), see information about schools/hospitals/ etc. in a certain area, also see news and insights real estate related. As the user, you can also purchase memberships (one that provides you with a personal real estate agent able to help you with everything needed or the other option that also includes a moving company).

As the admin, you are able to view, add, delete and update properties, agents, other admins, view statistics and have access to other useful features.

The agent will have information related to the people subscribed to him or the houses that he manages and the planning calendar for the viewings.

The moving company needs to have a calendar with the future planning for moving.

# SOLID Principles and applicability

SOLID Principles that play a major role in Object-Oriented Programming, as they make software designs more maintainable, understandable, well structured, effective, and flexible. It is of great importance to consider them and to use the principles within this project.

There is five design that contribute to the concept of SOLID:

1. *The Single-responsibility principle:* This refers to the idea that every class should have only one responsibility. In this project, I try to divide every class, keeping them as one responsibility entities ( ex: User, Person, Admin, Agent, Property, Address)
2. *The Open-close principle:* "Software entities ... should be open for extension, but closed for modification." It doesn’t take much change in the existing code to add needed classes and functionalities.
3. *The Liskov-substitution principle:* “The principle defines that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. That requires the objects of your subclasses to behave in the same way as the objects of your superclass” All the subclasses that inherit from a superclass behave the same way, there is no class that has way different functionalities.
4. *The Interface-segregation principle:* "Many client-specific interfaces are better than one general-purpose interface." Interfaces are used for the controllers, each having their own methods that need to be implemented in the class that uses it. (IUserController, IAdminController etc.)
5. *The Dependency inversion principle:* "Depend upon abstractions, [not] concretions."

# React

For the project, I am using React, which is a framework for creating user interfaces for websites and applications, as it provides flexibility and the possibility to reuse parts of the code.

Creating components is an advantage because it keeps the code clean, organized, and not repetitive. React provides high performance, time-efficient development because it allows the developer to create a collection of reusable components.

It was easy to learn how to use React in a proper way, and, most importantly, I understood the concepts on which React is based. (less use of classes, and embracing the use of react hooks, use state)

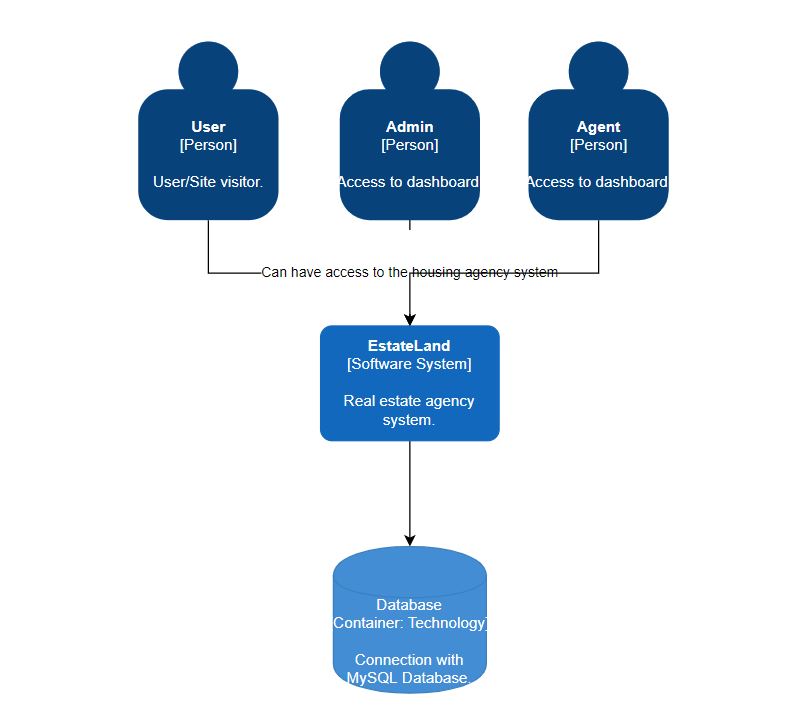
At this moment, React is the most popular web development tool, which means there is a large community, more than enough online resources, documentation, materials, discussions on different problems, that help me during the development of the project.

# Springboot

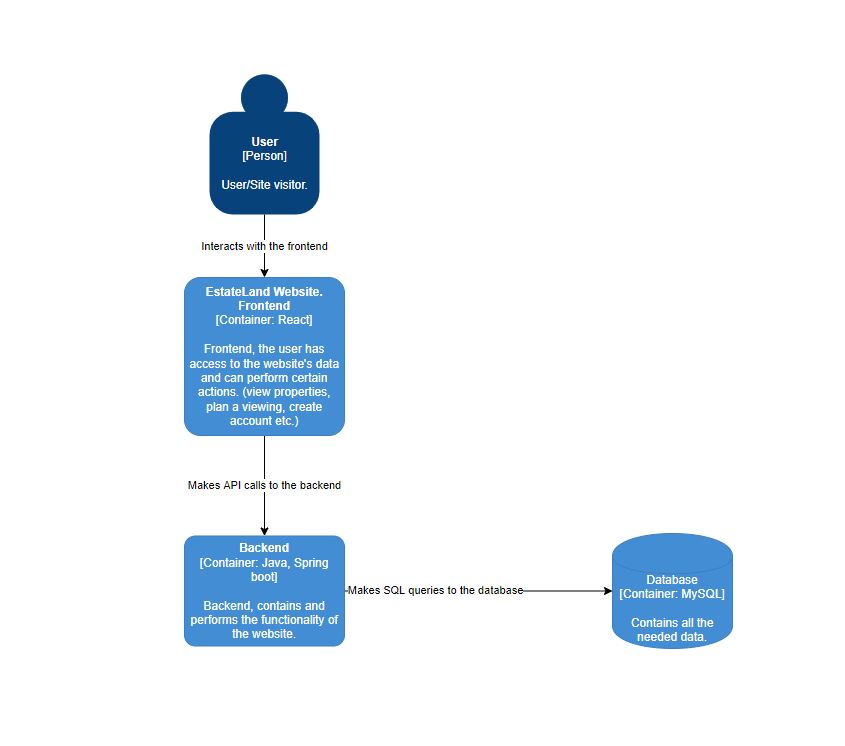
Spring boot is a java framework that aims to reduce the overall development time and increase efficiency by having a default setup for unit and integration tests.

Spring's dependency injection approach encourages writing testable code. Easy to use but powerful database transaction management capabilities.

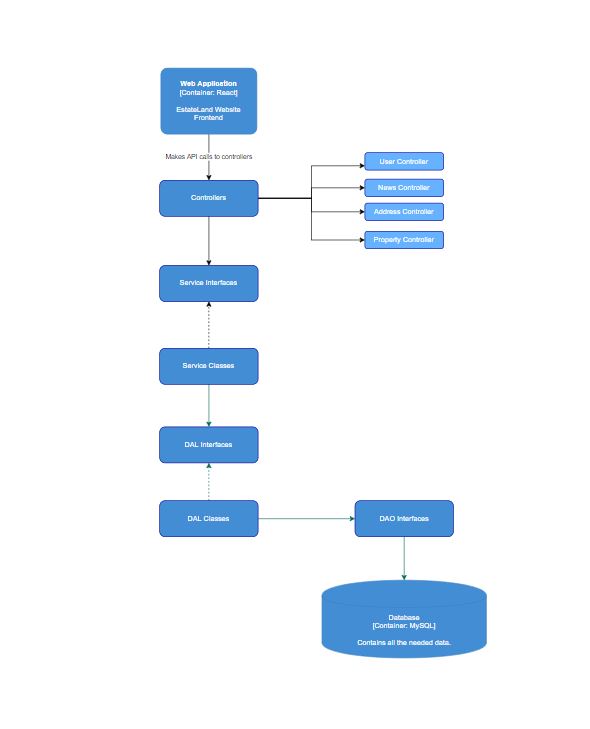
# C4 DIAGRAMS



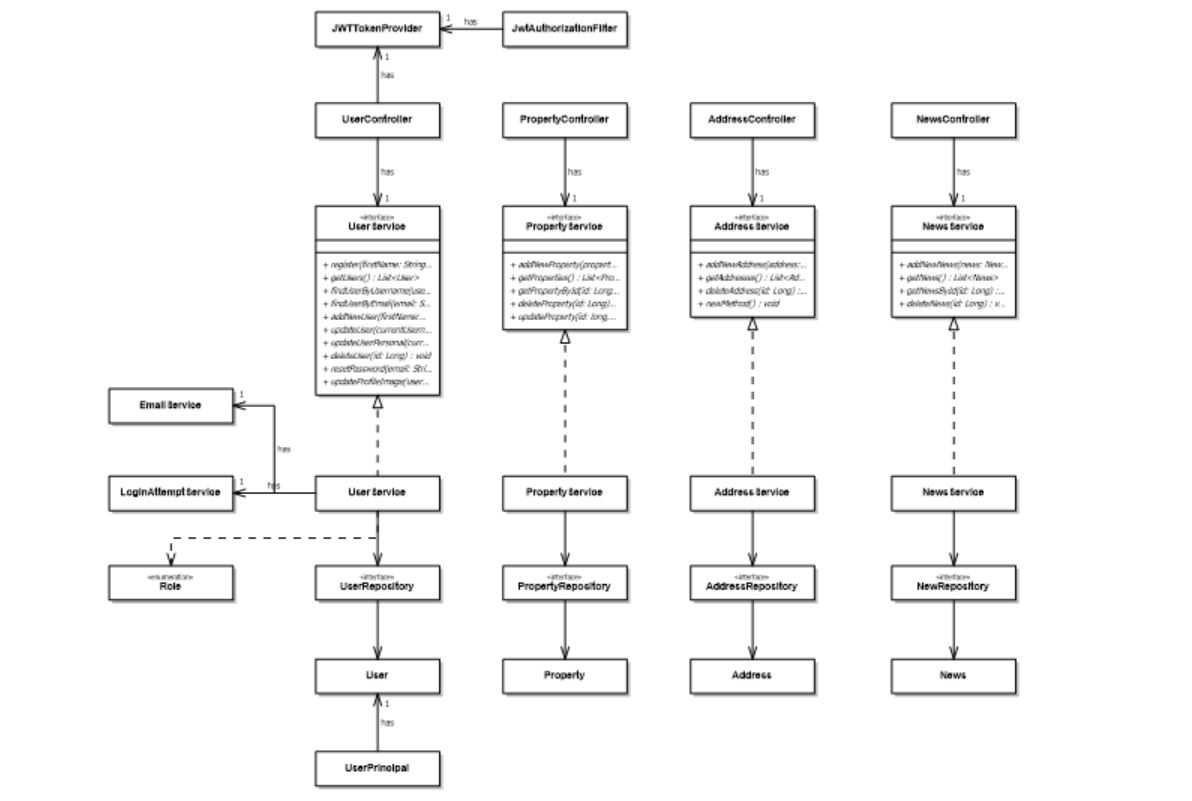
The web application has 3 main users, the site visitor that is able to view the readable data, request or plan viewings, create an account, etc., the admin that has access to the main managerial functionalities such as statistics, seeing an overview of the properties, users, agents, perform update, add or delete actions etc., and the agent that can manage the viewings and some properties.



The application has 3 essential containers: frontend, backend and database. For the frontend’s development I use React, the user only interacts with the frontend and has access to certain actions. The backend, Java Spring Boot, is responsible for the logic of the website and also for retrieving data from the MySQL database, which later will be displayed on the frontend. The database stores the needed data.



The following diagram represents the structure and organization of the backend.



This is the UML diagram, with the classes used for the backend.