Design Document

**Teachers:** Kiavash Bahreini, Márcio Paixão Dantas

**Student Name:** Omar Abou Dehn

**Student Number:** 3560813

**Interface Git Repository:** https://git.fhict.nl/I407846/safar\_travelapp



Document Version History:

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.1 |  |  |

Contents

[Architecture: 4](#_Toc99491725)

[Three Tier Architecture: 4](#_Toc99491726)

[C4 Architecture Diagrams: 6](#_Toc99491727)

[Entity Relational Diagram: 10](#_Toc99491728)

[UML Class Diagram: 11](#_Toc99491729)

[Backend Framework: 12](#_Toc99491730)

[Frontend Framework: 13](#_Toc99491731)

[Database Framework: 14](#_Toc99491732)

# Architecture:

## Three Tier Architecture:

The three-tier architecture will be used to build the web-app, which consists of a presentation layer built using React framework, a logic layer built using SpringBoot, a persistence layer using MySQL.

Some of the advantages of using it are that it can be developed fast as multiple teams can work on different layers, and since the layers are separated, they can be scaled independently from one another, moreover the data coming from the frontend can be processed and validated in the backend before being inserted to the database which increases security.

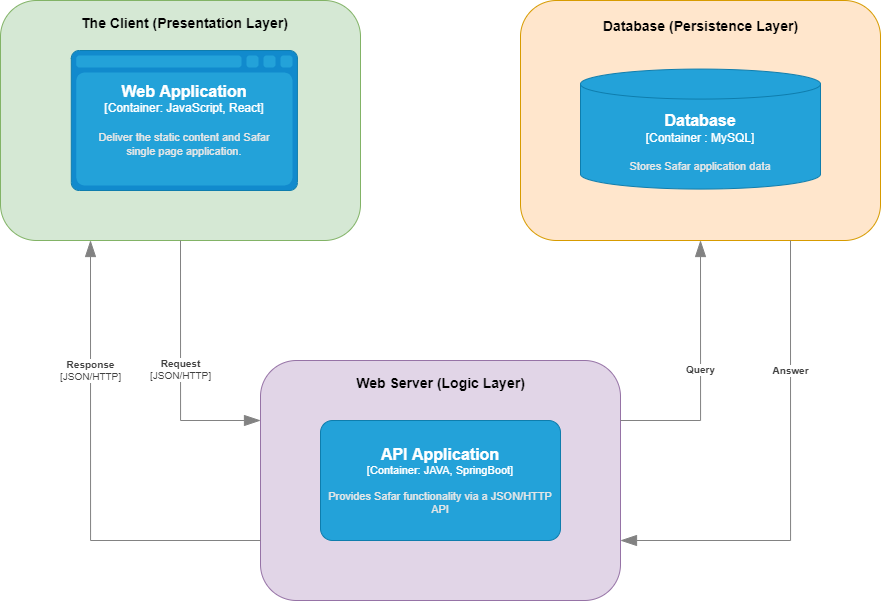


Figure Three Tier Architecture

## SOLID Design Principle:

SOLID principles are a set of five principles that ensure good Object-Oriented Design.  
Those principles are:

1. Single Responsibility Principle: A class should have one and only one reason to change, meaning that a class should have only one job.
2. Open Closed Principle: Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.  
   The idea behind this principle, is to ensure that the functions/classes have scope for extension in the future. New features can be added to it, without introducing new bugs. You’ll add new features by extending or writing new code without modifying old code.
3. Liskov Substitution Principle: functions that use pointers to base classes must be able to use objects of derived classes without knowing it.
4. Interface Segregation Principle: A client should never be forced to implement an interface that it doesn’t use, or clients shouldn’t be forced to depend on methods they do not use.
5. Dependency Inversion Principle: Entities must depend on abstractions not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.

During this project, the above-mentioned principles will be followed and applied when relevant.

## C4 Architecture Diagrams:

The C4 model consists of a hierarchical set of software architecture diagrams for context, containers, components, and code which provides different levels of abstraction for different audience.

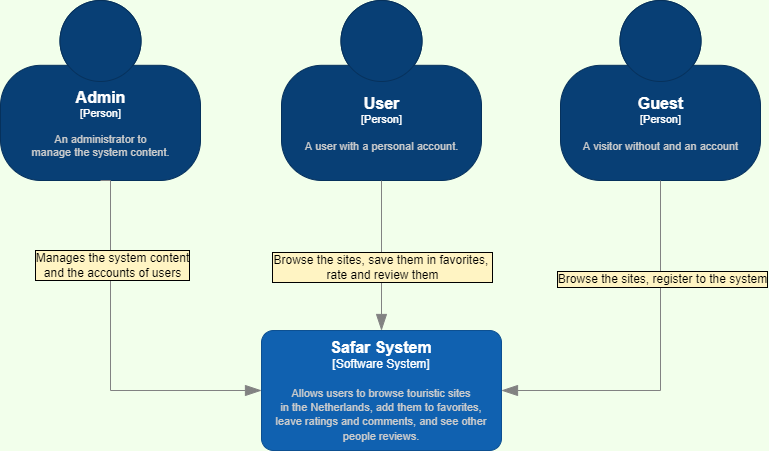


Figure C1

Diagram

Description automatically generated

Figure C2

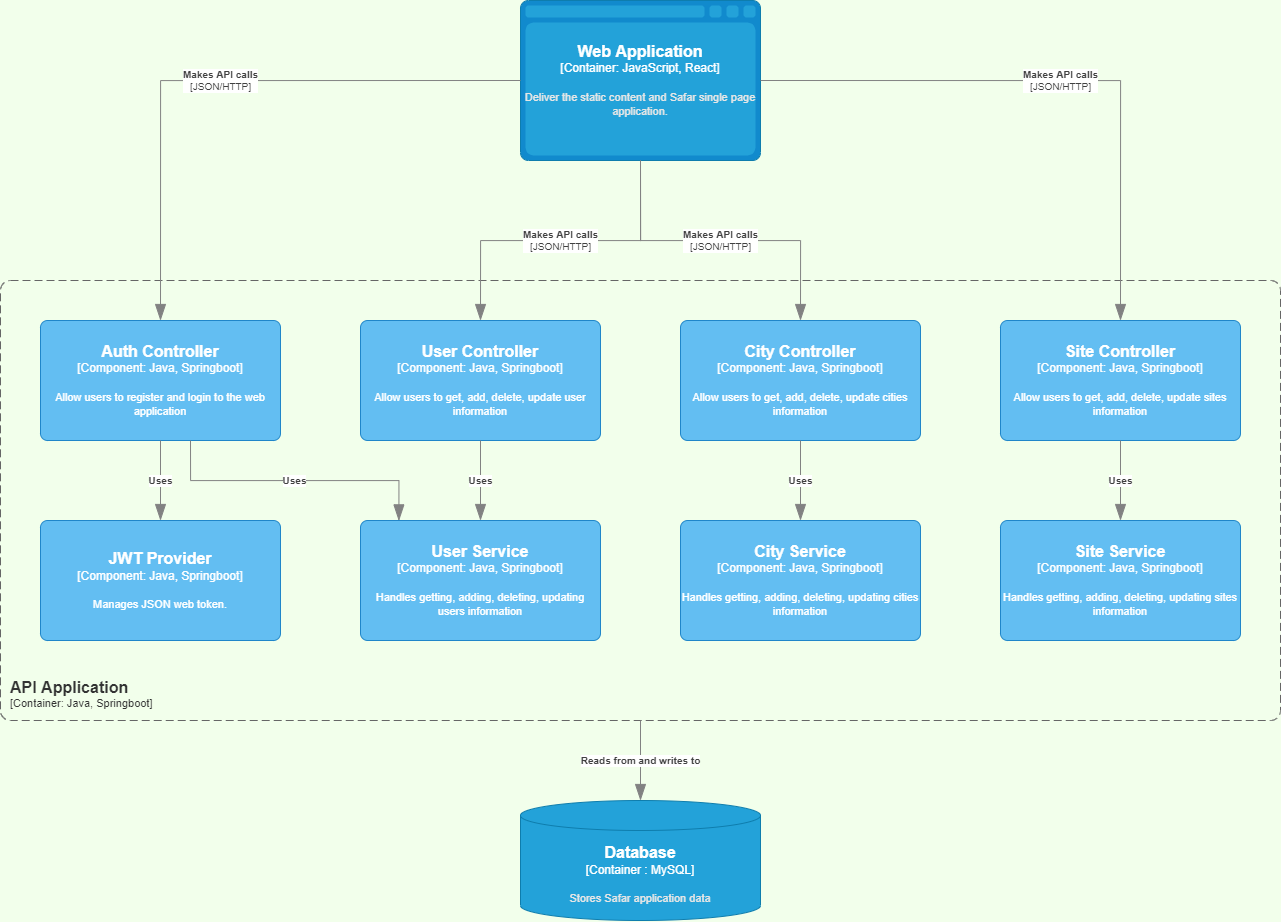
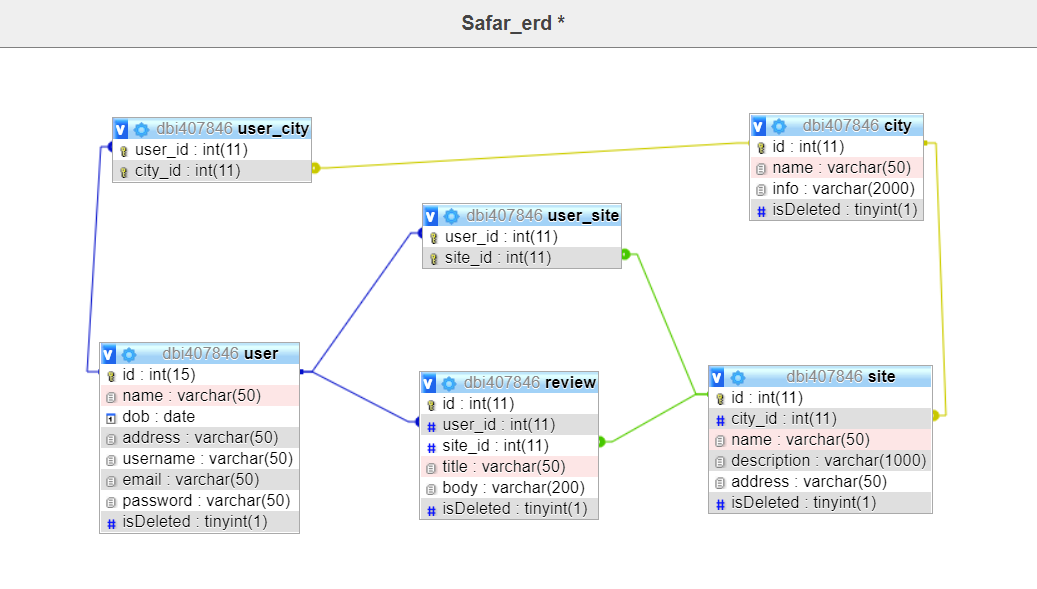
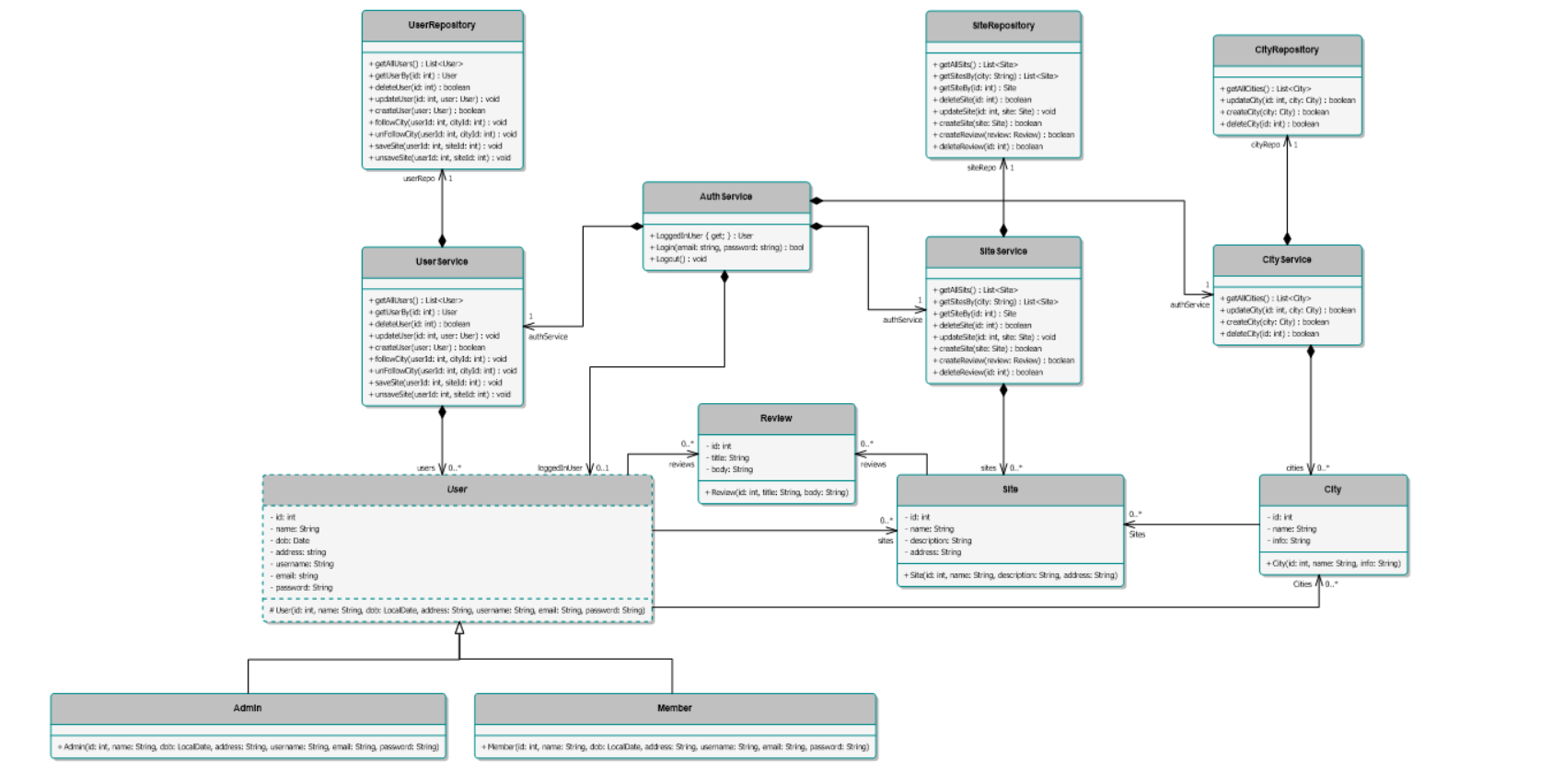


Figure C3

# Entity Relational Diagram:



# UML Class Diagram:



# Backend Framework:

Spring Boot will be used to build the restful API as it is one of the most powerful and famous Java frameworks, and due to its popularity, it has a good community and support and a lot of online resources.

# Frontend Framework:

In this chapter the three most used JS frontend frameworks will be compared based on different criteria in order to determine which one will be used to build Safar interface.

Angular: an application design framework and development platform for creating efficient and sophisticated single-page apps [(1)](https://angular.io/docs).

React: a JavaScript library for building user interfaces.

Vue: a progressive JS framework for building user interfaces.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Angular** | **React** | **Vue** |
| **User Friendly** | + | + | + |
| **Performance** | + | + | + |
| **Reusable UI Components** | + | + | + |
| **MVC Support** | + | - | - |
| **Documentation** | + | - | + |
| **Strong/Large Communities** | + | + | - |
| **Testability** | + | + | + |

[(2)](https://www.codeinwp.com/blog/angular-vs-vue-vs-react/), [(3)](https://athemes.com/guides/angular-vs-react-vs-vue/), [(4)](https://academind.com/tutorials/angular-vs-react-vs-vue-my-thoughts/), [(5)](https://medium.com/techmagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d), [(6)](https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/)

# Database Framework:

|  |  |  |
| --- | --- | --- |
|  | **JDBC** | **JPA** |
| **Abstraction Level** | Low level standard for interaction with databases | High level standard for interaction with databases |
| **Usage** | JDBC allows to do more things with the Database directly which gives more flexibility | JPA allows to use an object model in the application which requires less code |
| **Migrating Database** | Due to its low abstraction, a lot of changes are required when changing the database | Due to its high abstraction, the database can be switched by changing some configurations |