DESIGN DOCUMENT

S3_IPS - MUZIEKZAALS

Noelia Rodriguez Morales

V2.0 - 14.04.2022

Table of Contents

Version Control:

1.0	21.03.2022
2.0	14.04.2022

1. How is SOLID guaranteed

Using the diagram model C4 it is possible to show multiple points of view that explains the decomposition of the system in containers and components and their relations between them and with the users.

Single Responsibility Principle: Each class has only one purpose to cover.

How do I apply it: Defining multiple classes but assigning only one purpose to each one.

Open/Closed Principle: Every entity as classes, modules, functions.. have to be open for extension of new features but closed for modification of its intrinsic function.

How do I apply it: Keeping in mind new functionalities for the application that could be added later.

Liskov Substitution principle: Any subclass should be substitutable for its base class in a way that each member of an inheritance hierarchy should fulfil the same behavioural contract.

How do I apply it: applying inheritance between classes with similarities when it is possible.

Interface Segregation Principle: Any class should be forced to depend on methods that it is not using.

How do I apply it: isolating methods in small interfaces that are used by classes that need them.

Dependency Inversion Principle: Organize the class structure in different levels in a way that high level modules do not have any dependency on the low level modules. In this way high level modules include an instance of low level modules so details depend on the abstractions and not the other way around.

How do I apply it: Organizing classes in levels and make low level modules to depend on high level modules.

2. Design decisions justification

Why are you using Spring boot?

Because Spring boot uses libraries from third parts that makes easier to create applications based in Spring that works in an autonomous way and that are easy to run.

How do you separate concerns?

Establishing a well organized system where each part fulfils a role at the same time that it is open to adapt to changes. For that I will use modules to divide the software in addressable components that works together to cover functionalities.

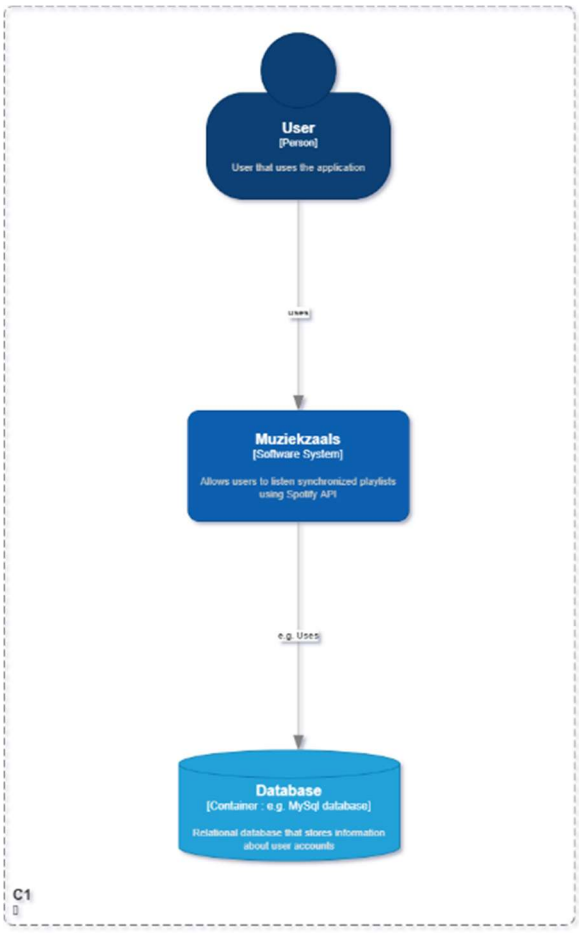
Which framework are you using and why?

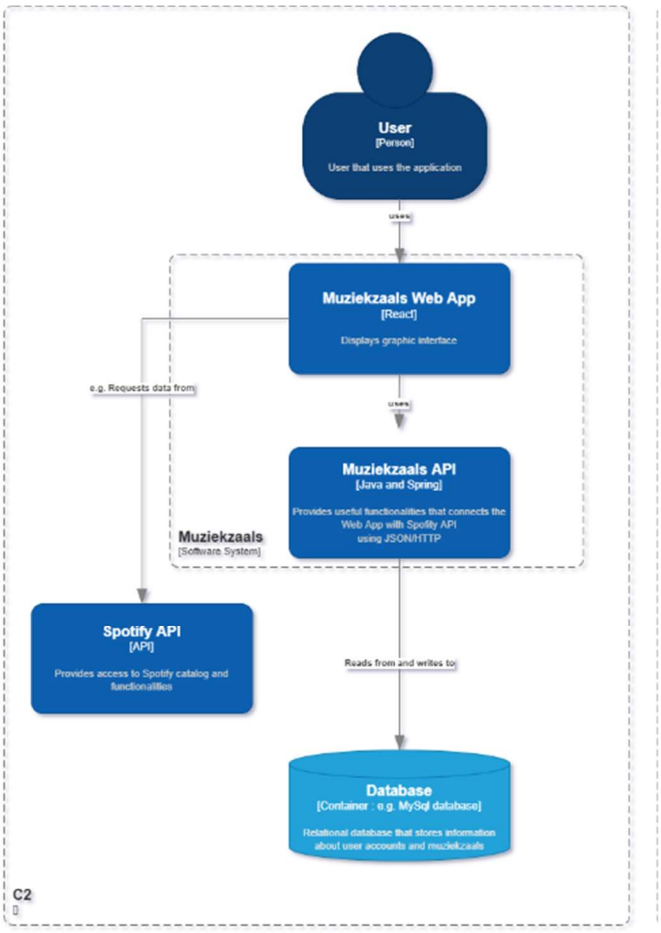
I am using Spring framework because is an adjustable framework that allows me to program faster on Java.

3. C4 architecture diagrams

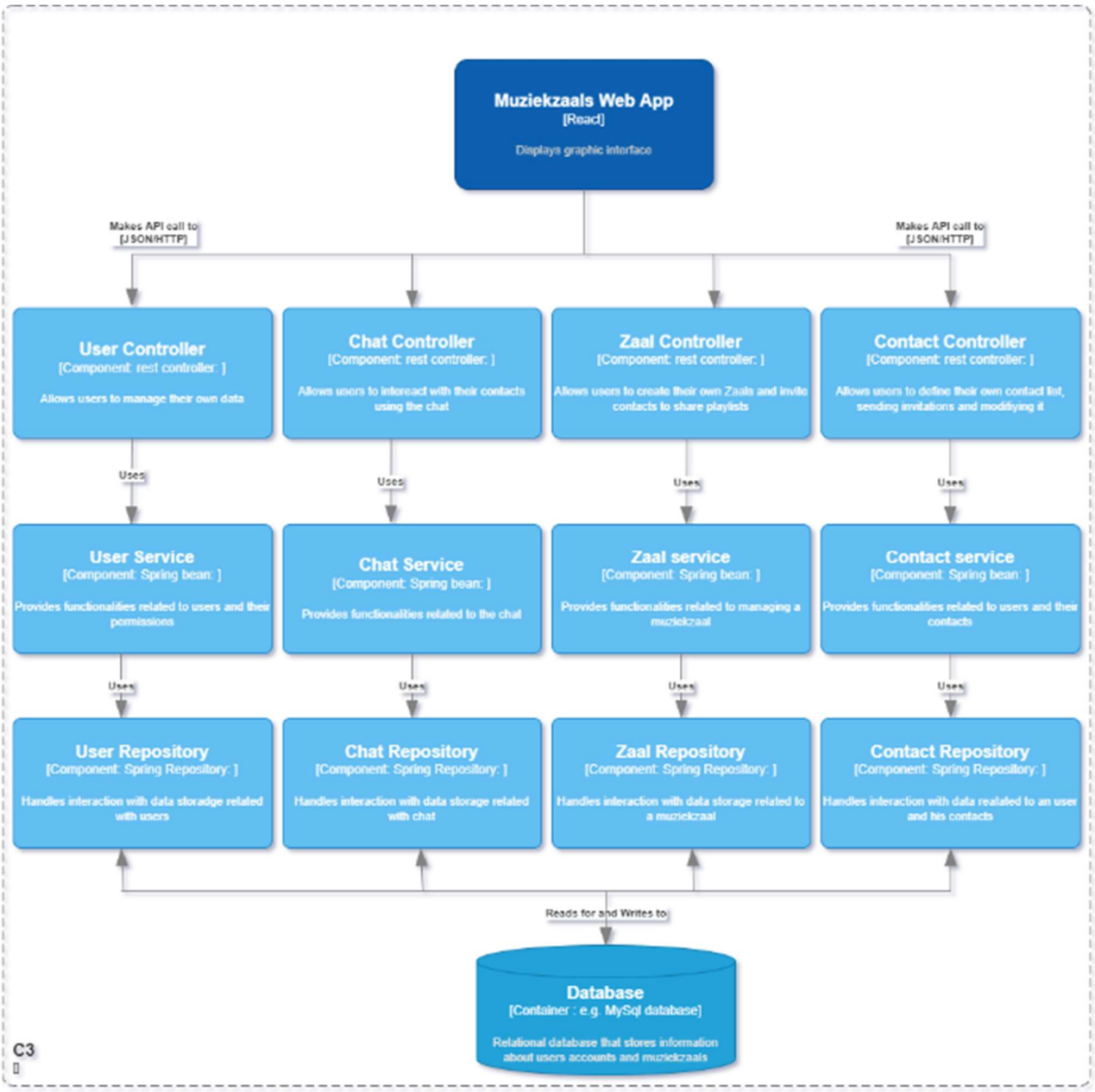
In the C1 (context) diagram it is shown how the Muziekzaals system is related to the user and the database.

In the C2 (containers) diagram we zoom on the system and we can see the components of the system: A Web application based in React, an API that uses Java and spring and an external API provided by Spotify that is connected to the Web App.





In the C3 (Components) diagram, we can see with more detail the main components of the API.


Muziekzaals Web App
[React]
Displays graphic interface
Makes API call to
[JSON/HTTP]
Makes API call to
[JSON/HTTP]
User Controller
[Component: rest controller:]
Allows users to manage their own data
Chat Controller
[Component: rest controller:]
Allows users to interact with their contacts using the chat
Zaal Controller
[Component: rest controller:]
Allows users to create their own Zaals and invite contacts to share playlists
Contact Controller
[Component: rest controller:]
Allows users to define their own contact list, sending invitations and modifying it
Uses
Uses
Uses
Uses
User Service
[Component: Spring bean:]
Provides functionalities related to users and their permissions
Chat Service
[Component: Spring bean:]
Provides functionalities related to the chat
Zaal service
[Component: Spring bean:]
Provides functionalities related to managing a muziekzaal
Contact service
[Component: Spring bean:]
Provides functionalities related to users and their contacts
Uses
Uses
Uses
Uses
User Repository
[Component: Spring Repository:]
Handles interaction with data storadge related with users
Chat Repository
[Component: Spring Repository:]
Handles interaction with data storage related with chat
Zaal Repository
[Component: Spring Repository:]
Handles interaction with data storage related to a muziekzaal
Contact Repository
[Component: Spring Repository:]
Handles interaction with data realated to an user and his contacts
Reads for and Writes to
Database
[Container : e.g. MySql database]
Relational database that stores information about users accounts and muziekzaals
C3

In the C4 (Code) we can see how are designed the relations between the components, including classes, interfaces and its implementations.