GLUB TRAINING MANAGER

**ARCHITECTURE DOCUMENT**

Cristin Rusnac | Fontys University of Applied Science

Contents

[C4 model 2](#_Toc75118418)

[C1 3](#_Toc75118419)

[C2 4](#_Toc75118420)

[C3 5](#_Toc75118421)

[C4 6](#_Toc75118422)

[Entity Relation Diagram 7](#_Toc75118423)

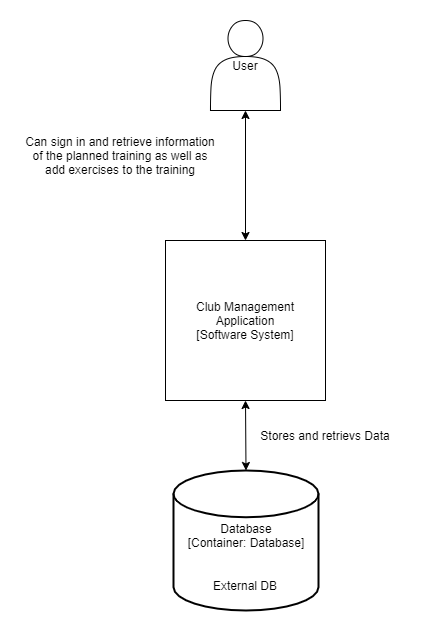
[CI/CD Diagram 8](#_Toc75118424)

# C4 model

C4 model is a lean graphical notation technique for modelling the architecture of software systems. It is based on a structural decomposition of a system into containers and components and relies on existing modelling techniques such as the Unified Modelling Language (UML) or Entity Relation Diagrams (ERD) for the more detailed decomposition of the architectural building blocks.

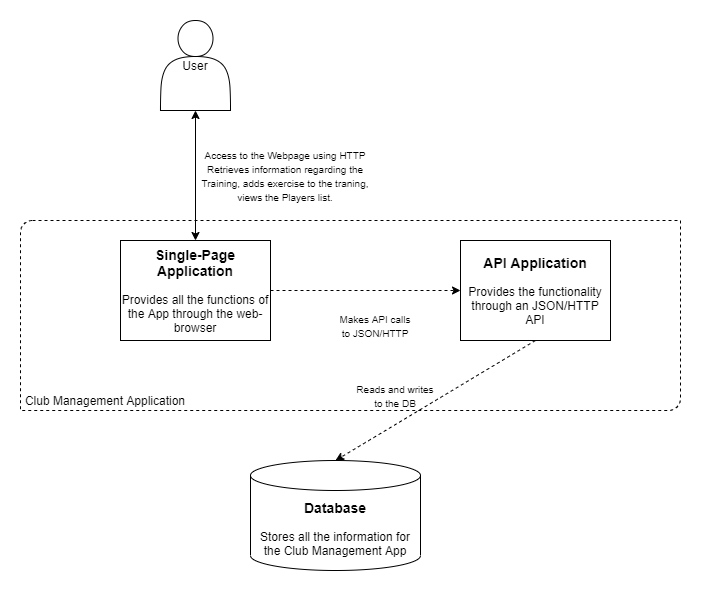
## C1

A System Context diagram provides a starting point, showing how the software system in scope fits into the world around it. In this diagram, you see a User that is meant to interact with all the features of the application, like authentication, retrieving, adding, editing and deleting data. External systems are seen as well. There is a Postgres DB used to store data. And there is the Software System that provides all the functionality.



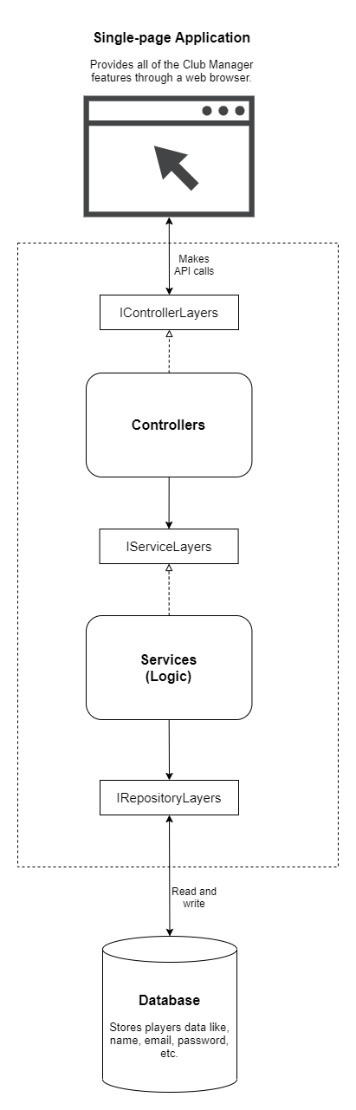
## C2

A Container diagram zooms into the software system in scope, showing the high-level technical building blocks. This diagram is a zoomed-in version of C1. The application consists of 2 containers. The website made using React framework and Javascript and the API application made using Java and Springboot. The User visits the website where he can manipulate the data. The website makes an API call to the API who would store/ retrieve the requested data via the Postgres DB.



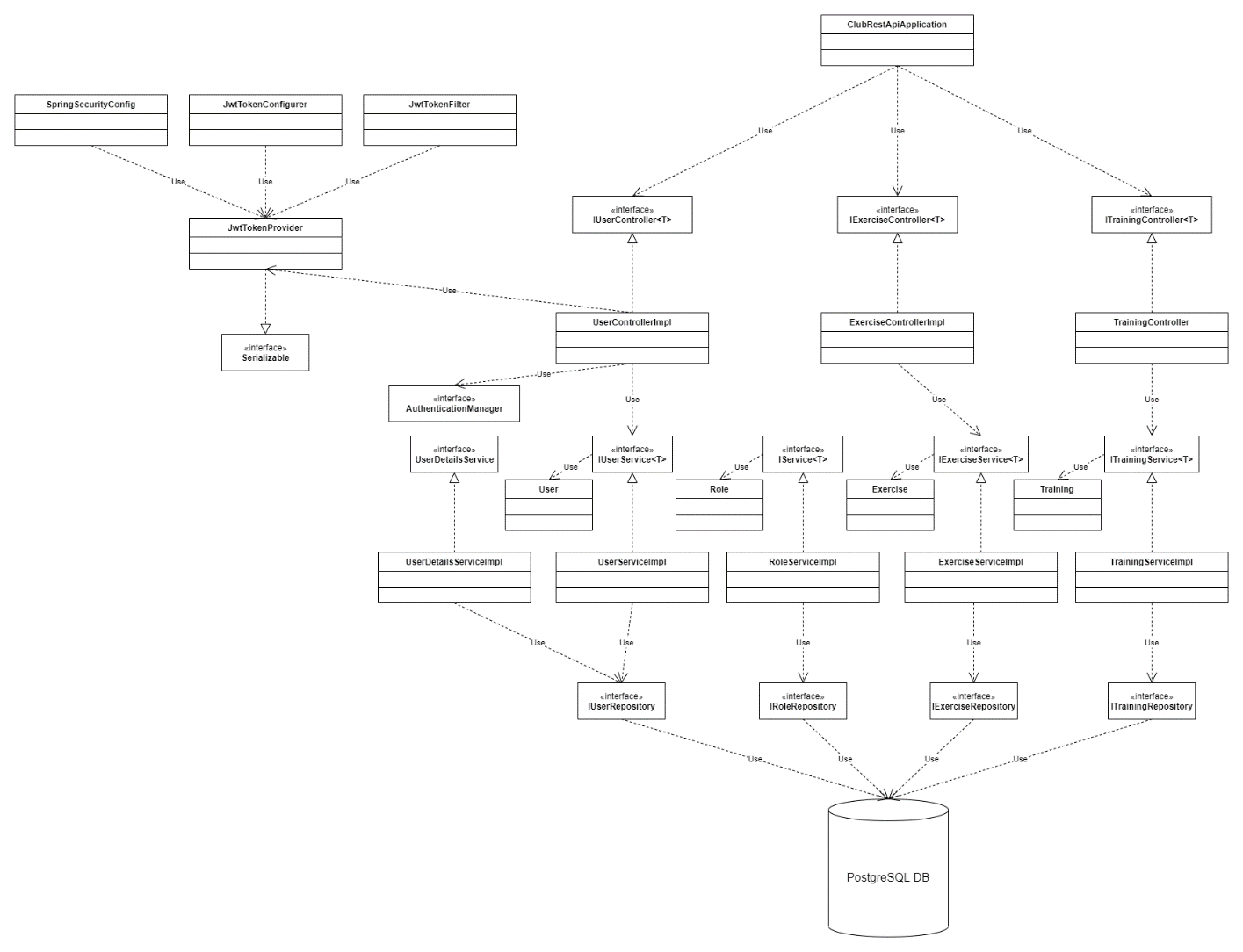
## C3

A Component diagram zooms into an individual container, showing the components inside it. This Diagram is a closer look to the Application. Our classes are divided into certain layers for a better-optimized flexible application wich follows the SOLID principles. Controllers are the ones that will receive and return the request call from the front-end. Service classes are the ones that will do the logic and the algorithm stuff. Repository classes are the ones that will retrieve/store data to the external database and are the only layer that has access to the database. The logic classes and Repository classes inherit from an interface layer. This is done for Dependency Inversion (Solid) to make it more flexible.



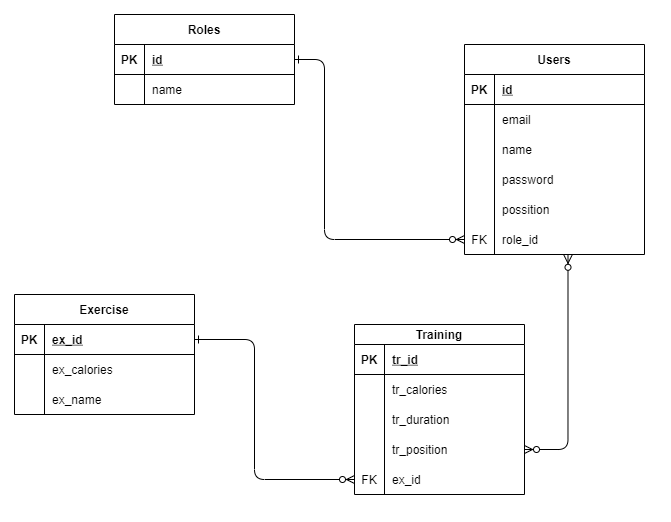
## C4

A code (e.g. UML class) diagram can be used to zoom into an individual component, showing how that component is implemented. In this layer the full Architecture is viewed in a UML diagram. The Repository Interfaces have the purpose of communication with the Database and are used by the Service layer. The services are responsible for all the logic of the application. The controllers use the services through appropriate service interfaces and are responsible for the CRUD operations. There is also a JWT Token Provider class which is responsible for the generation of a token during authentication.



# Entity Relation Diagram

The ERD represents the architecture of the Postgres Database that is used by the SpringBoot Backend which retrieves, adds and deletes data from it.



# CI/CD Diagram

The pipelines set are meant to ensure that the application has no errors in its functionality after changes are being implemented. At first the backend is built then the tests are run. The sonarqube evaluates the code after which a image of the backend is built, deployed to docker and launched in a container that will be used for the next step. The cypress phase deploys the react frontend and uses the backend container to run the cypress test which tests all the functionalities of the application. After the cypress phase is successful the front end is also built into a docker image and then the docker container are terminated and the CI/CD pipeline is over.

