School Engine

Backend HLD

Document Number 1

Table of Contents

[Introduction 3](#_Toc121830473)

[Overview 4](#_Toc121830474)

[Libraries and software 5](#_Toc121830475)

[Libraries 5](#_Toc121830476)

[Code structure 6](#_Toc121830477)

# Introduction

This file is responsible for describing the backend for the final year project. After reading this document, you should have a basic understanding of how the system works.

# Overview

Graphical user interface, diagram

Description automatically generated

# Image 1: Code structure

The main components of the backend will be split into 3 categories: the http controller, the object managers and the cql managers.

The basic flow of the software is going to be the following:

1. An http requests comes to the backend
2. The backend processes and parses the request
3. Calls the object related object manager
4. The object manager calls the related cql managers
5. The cql managers communicate with the Cassandra database for the related information

# Libraries and software

## Libraries

For the backend code the software is going to be written in C++.

In order to implement this software, the project is going to use one of the most popular backed libraries: [**Drogon**](https://drogon.docsforge.com/). This library will contain features such as:

* **Http controllers** for communicating with other machines
* **SSL security**. For the scope of this project the backend will use a self-signed certificate but, considering that the software might be deployed on the internet, a normal security SSL certificate can be used.
* **Json formatting** for formatting the information in a proper standard formatting system.
* **Logging** for logging events of the project such as error, important events etc.

Drogon has an integrated library that can communicate with MySql and PostgreSQL, but I have decided to use another non-sql database that is able to achieve some important features from the requirements, mainly scalability and fault tolerance. For these reasons I have decided to choose Cassandra, which has the features mentioned prior.

For using Cassandra with C++, the library that is going to be used, is going to be [**casssandra-cpp**](https://github.com/datastax/cpp-driver)**,** an opened soured library that is also used in production software.

Furthermore, for the testing part, the [**Google Test C++**](https://github.com/google/googletest) library will be used considering that it is well know and used into industry.

In order to keep track of the code version control this software is going to use Git and GitLab and for building the project, **CMake** has been the chosen software.

Considering that the software should also be deployable on the internet, we have to make it a container, therefore **Docker** is going to also be used to containerise the backend.

For code formatting, **Clang** is going to be the way to go if we want to be sure that the code is properly formatted.

## Code structure

The structure of the code is going to respect a simple to understand logic.

├── CMakeLists.txt -> The building configuration

├── Dockerfile -> The file that will be able to make this software deployable

├── certificates -> The folder that contains the SSL certificate

├── conf.json -> The file that will contain the project’s settings

├── documents -> The folder that will contain the LLDs and HLDs

├── format.sh -> The formatting code

├── include -> The folder that will contain the header files

├── modules -> The folder that will contain Drogon, google-test & cass-cpp

├── src -> The folder that will contain the source code

└── test -> The folder that will contain the testing code code

# Class structure

In order to construct the software displayed in Image 1 multiple classes will need to be implemented.

## Cql manager

The cql\_manager implementation is going to be the group of classes that is going to communicate with the database. Considering that we are going to have 23 tables in our database, the code is going to have one class for each table. They will be similar to each other and will contain functionality such as initialising the table, creating a new entry in the database, editing it, or deleting it.

Furthermore, another class is going to be created that will take care of smaller steps such as managing memory and taking care of the connection with the database. More details can be found in the CQL Managers LLD.

## Object managers

After building the cql managers, in order to combine of the information, some new classes will be implemented that will take care and make sense of all of the information.