School Engine

Database LLD

Document Number 2

Table of Contents

[Introduction 3](#_Toc124700977)

[Database 3](#_Toc124700978)

[Diagram 3](#_Toc124700979)

[Keyspaces 3](#_Toc124700980)

[Tables 4](#_Toc124700981)

[environment.schools 4](#_Toc124700982)

[environment.countries 4](#_Toc124700983)

[environment. holidays\_by\_country\_or\_school 4](#_Toc124700984)

[schools.announcements 5](#_Toc124700985)

[schools.announcements\_by\_tag 5](#_Toc124700986)

[schools.answers 5](#_Toc124700987)

[schools.answers\_by\_announcement\_or\_question 6](#_Toc124700988)

[schools.courses 6](#_Toc124700989)

[schools.courses\_by\_user 7](#_Toc124700990)

[schools.files 7](#_Toc124700991)

[schools.grades 7](#_Toc124700992)

[schools.tags 8](#_Toc124700993)

[schools.tags\_by\_user 8](#_Toc124700994)

[schools. student\_reference 9](#_Toc124700995)

[schools.lectures 9](#_Toc124700996)

[schools.tokens 9](#_Toc124700997)

[schools.users 9](#_Toc124700998)

[schools.users\_by\_course 10](#_Toc124700999)

[schools.users\_by\_tag 10](#_Toc124701000)

[schools.questions 10](#_Toc124701001)

[schools.questions\_by\_course 11](#_Toc124701002)

[schools.todos 11](#_Toc124701003)

[schools.todos\_by\_user 11](#_Toc124701004)

[Database objects 12](#_Toc124701005)

# Introduction

The purpose of this document is to describe in detail the design of the database, what every field represents, but also the code structure.

# Database

The database chosen for this project is Apache Cassandra. The main reasons why I chose this database are:

1. Scalability and reliability. Cassandra clusters/nodes can be connected, and they will communicate between them so that the data is stored on every node. This will ensure that, if one node breaks down, the data will not be lost.
2. How the data is stored. Cassandra is a nonrelational database, and it is not stored in a json format, but as a table. Although we will not be able to use some of the features a relation database has (such as joins), Cassandra has an important feature: we can use maps, sets and lists.
3. Compatibility with C++. The [Cassandra cpp-driver](https://github.com/datastax/cpp-driver/tree/2.16.0) made by datastax is an easy to use, reliable and tested driver used by multiple companies for their software.

# Diagram

In the diagram displayed bellow we can see all the tables that we are going to need in order to have a functional system.

Diagram

Description automatically generated

Image 1: Database diagram

# Keyspaces

The table will be split into two keyspaces (think of them as some kind of groups). The environment keyspace will contain the tables schools, countries, and holidays\_by\_country\_or\_school. The rest of the tables will be stored in the schools keyspace.

# Tables

This subchapter will cover the functionality and purpose of each table. We will have two keyspaces, on that will contain the general information about schools, holidays and countries, and another one that will contain the actual information of a specific school. Considering that Cassandra cannot create keyspaces based on a formatted name from an index, we will add an additional primary key for each of the table in the schools keyspace named “school\_id”.

## environment.schools

This table will contain the most important information about a school.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | id | uuid | The id of the school |
|  | name | varchar | The name of the school |
|  | country\_id | int | The id of the country that the school is in |
|  | image\_path | varchar | The path to the logo that the school has |

## environment.countries

This table will contain a list of all the countries in the world.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | id | int | The id of the country |
|  | name | varchar | The name of the country |
|  | code | varchar | The code of the country |

## environment. holidays\_by\_country\_or\_school

This table will contain all of the holydays that a country has and also every custom holiday that a school has. This will be used when the system will not show some classes in some specific days.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | country\_or\_school\_id | varchar | The if of the country or of the school |
| PK | type | int | 0 – if this is a national holiday  1 – if this entry was added by a school |
| CK | date | timestamp | The date of the holiday |
|  | name | varchar | The name of the holiday |

## schools.announcements

This table will contain all the announcements that the users have added.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the announcement |
| CK | created\_at | timestamp | The time the announcement has been created |
|  | created\_by | uuid | The id of the user that added this entry |
|  | title | varchar | The title of the announcement |
|  | text | text | The text that the announcement has |
|  | allow\_answers | boolean | True if other users can add answers. False otherwise |
|  | files | list<uuid> | A list of uuids that point to the files used for this announcement. We want a list because we want to keep the order of these files |

## schools.announcements\_by\_tag

This table will contain all announcements that a specific tag will have.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | tag\_id | uuid | The id of the tag |
| CK | announcement\_id | uuid | The id of the announcement |

## schools.answers

This table will contain all the answers that an announcement has.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the answer |
|  | text | text | The text of the answer |
|  | created\_by | uuid | The id of the user that added this entry |
| CK | created\_at | timestamp | The timestamp at which the answer was created |

## schools.answers\_by\_announcement\_or\_question

This table will contain all the answers’ id that an announcement or a question has.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | announcement\_or\_question\_id | uuid | The id of the announcement or the answer |
| PK | type | int | 0 – for an announcement  1 – for a question |
| CK | answer\_id | uuid | The id of the answer |

## schools.courses

This table will contain all the courses that schools have.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the course |
|  | name | varchar | The name of the course |
|  | course\_thumbnail | varchar | The path to the logo of the course. If null, chose from the list of default images |
|  | created\_at | timestamp | The date that the course was created |
|  | start\_date | timestamp | The date the course starts |
|  | end\_date | timestamp | The date the course ends |
|  | files | list<uuid> | List of all the files that are used by the course |

## schools.courses\_by\_user

This table will contain the courses that a user is attending.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | user\_id | uuid | The id of the user |
| CK | course\_id | uuid | The id of the course |

## schools.files

This table will contain the files that are in the system.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the file |
|  | type | int | 0 – for normal file  1 – for folder |
|  | files | list<uuid> | A list with all files that this entry points to. This field will not be taken into consideration if the type is not a folder |
|  | name | varchar | The name of the file |
|  | path\_to\_file | varchar | The path to the file in the backend |
|  | size | int | Size of the file in bytes |
|  | added\_by\_user\_id | uuid | The id of the user that added this file |
|  | visible\_to\_students | boolean | True if users can view this file; false otherwise |
|  | students\_can\_add | boolean | True if students can add files in response; false otherwise |

## schools.grades

This table will contain all the grades of a school.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the grade |
|  | evaluated\_id | uuid | The id of the evaluated user |
|  | course\_id | uuid | The id of the course that this |
|  | evaluator\_id | uuid | The if of the teacher that evaluated. |
|  | value | float | Float value from 0 to 100 for a percentage. Else the value received out of the next field |
|  | out\_of | float | The maximum value the value can take. Null if we use percentages |
|  | created\_at | timestamp | The date this grade was created |
|  | weight | float | Percentage of how much the grade weights in comparison to other grades at the same course. Null if this feature is not used. |

The rules of using the weight are as follows when we want to calculate the final grade:

* If we have grades that do not have a weight, we will multiply the received value by . If the top value is negative, we do not add them, thus resulting in unweighted grades not being used.
* If the sum of weights is , we calculate the final grade as usual and then apply the next formula .

## schools.tags

This table will contain all of the tags that a school has. All of the tags can be created by the admins and can be assigned to users by both teaches and admins.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the grade |
|  | value | varchar | The name of the tag |
|  | colour | varchar | The colour that the tag has |

## schools.tags\_by\_user

This table will return all of the tags that a user has.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | user\_id | uuid | The if of the user |
| CK | tag\_id | uuid | The if of the tag |

## schools. student\_reference

This table will contain the mail and phone numbers of a student’s parents or guardians.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | student\_id | uuid | The id of the student |
| CK | reference | varchar | Email of phone number of the parent/guardian |
|  | type | int | 0 – for email  1 – for phone number |

## schools.lectures

This table will contain all of the lecture times that a course has.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | course\_id | uuid | The id of the course |
| CK | starting\_time | timestamp | The time at which the lecture starts |
|  | duration | int | The duration in minutes |
|  | location | text | The location of the lecture. Will be null if not added |

## schools.tokens

This table will contain all the tokens of a user. Will be used when wanting to log in.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | value | varchar | The value of the token |
| CK | user\_id | uuid | The id of the user |

## schools.users

This table will contain all the data of a user.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the user |
|  | email | varchar | The email of the user |
|  | password | varchar | The password for the account |
|  | type | int | 0 – for admin  1 – for teacher  2 – student  3 – other |
|  | changed\_password | boolean | True if user has changed password; false otherwise. |
|  | first\_name | varchar | The first name of the user |
|  | last\_name | varchar | The last name of the user |
|  | phone\_nr | varchar | The phone number of the user |
|  | last\_time\_online | timestamp | The last time the user was online. It is used for highlighting new announcements |

## schools.users\_by\_course

This table is used for selecting all users that are in a course.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | course\_id | uuid | The id of the course |
| CK | user\_id | uuid | The id of the user |

## schools.users\_by\_tag

This table is used for selecting all users by a specific tag.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | tag\_id | uuid | The id of the tag |
| CK | user\_id | uuid | The id of the user |

## schools.questions

This table is used for storing the information about a question from a course.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the question |
|  | text | text | The text of the question |
|  | time\_added | timestamp | The time at which this question was posted |
|  | added\_by\_user\_id | uuid | The id of the user that added this question |

## schools.questions\_by\_course

This table is used for getting all questions of a course.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | course\_id | uuid | The id of the course |
| CK | question\_id | uuid | The id of the question |

## schools.todos

This table is used for storing all of the todos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | id | uuid | The id of the todo |
|  | text | text | The text of the todo |
|  | type | int | 0 – for not started  1 – for started  2 – for finished |

## schools.todos\_by\_user

This table is used for getting all of the todos of a user.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column Name | Type | Description |
| PK | school\_id | int | The id of the school |
| PK | user\_id | uuid | The id of the user |
| CK | todo\_id | uuid | The id of the todo |

# Database objects

For most of the tables there will be a C++ class that will store the correlated information from the database. This is done in order to keep the information easy to access.

The tables that will have a class will be:

|  |  |
| --- | --- |
| Database table | Correlated C++ class |
| environment.schools | SchoolObject |
| environment.countries | CountyObject |
| environment.holidays\_by\_country\_or\_school | HolidayObject |
| schools.announcements | AnnouncementObject |
| schools.answers | AnswerObject |
| schools.course | CourseObject |
| schools.tags | TagObject |

The classes will be simple and will have three main parts: the constructor and destructor, a function that will return the data in a Json format and all the variables that the class will have, these being all the fields that can be found in the correlated table.

The constructor will take as parameters a variable for each of the class’s variables. Note that, for every uuid variable, the constructor will take a string as parameter and change it to a uuid later.

As mentioned, each of these classes will have a function named Json::Value to\_json(bool secure) that will be responsible for returning the class’s data in a json format so that it can be transferred from the backend to the frontend.

The secure parameter represents if we want to add information such as the school’s id of the object. Considering that the end user will not need to receive such information, the system will not give it because of security reasons.

Furthermore, all the variables will be stored as public variables of the classes in order to keep the code clean. Usually, in production, in order to get or set a variable, you would have to add two functions for each variable of a class: a getter and a setter, but, as mentioned above, we will prioritise keeping things simple and clear and not using this approach.