**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**ПРОЕКТ**

**По дисциплине: «Databases»**

**Пояснительная записка**

**Исполнители:**  
Студенты группы БПИ181

Плющ E.Д., Кара Д.А., Никифоров М.С., Степанов Е.В., Дубинский Р.Д.

**Москва 2020**

# Введение

## Описание проблемы

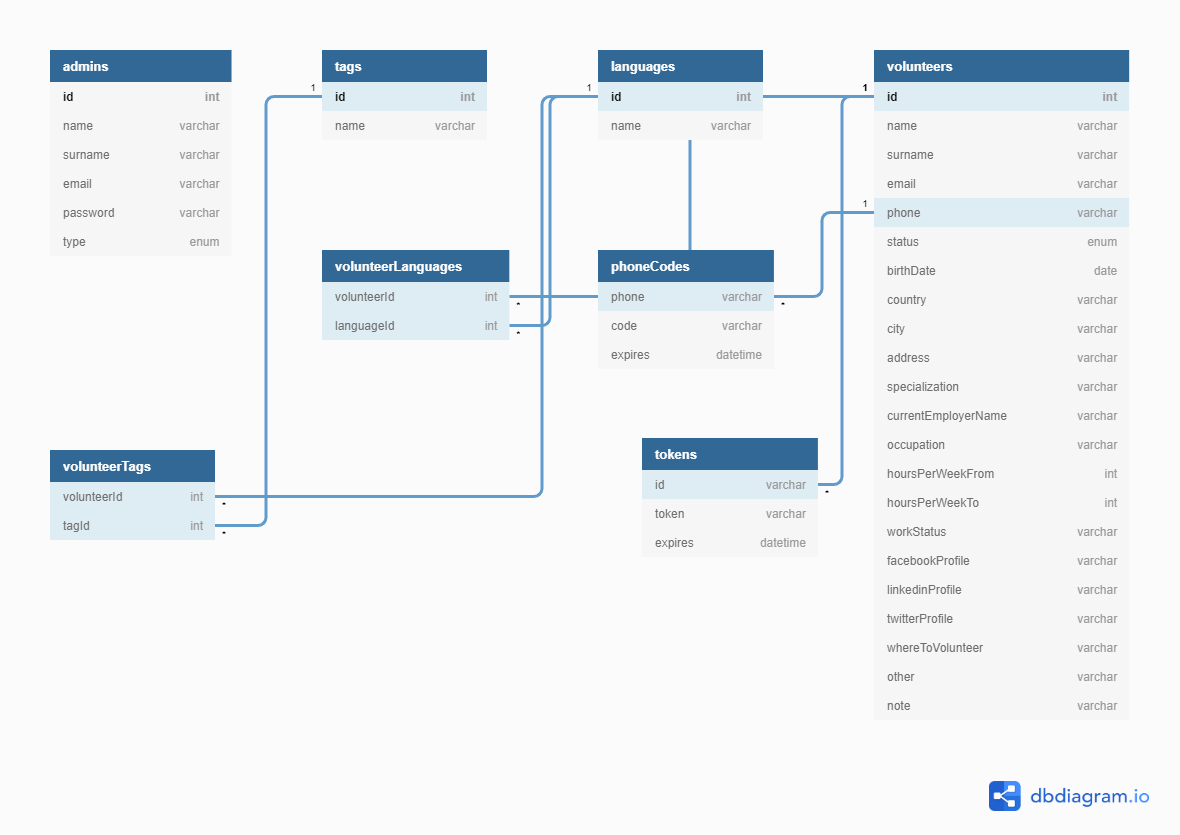
## Наше решение

# Требования

## Нефункциональные требования:

## Функциональные требования:

# Структура базы данных



As mentioned earlier the database structure was created by converting an existing MongoDB database to MySQL one. It can be seen that there is no relational table in MongoDB models for connecting languages to volunteers. But there is one for tags. The non normalized version could be the addition of a field with string type which will contain languages separated by comma. But it will create some issues when we need to delete, edit, or insert data. So we need to normalize it. That's why the new relational table volunteerLanguages was created. As for mongoDB model, there is no need for a table, as in this project the count of the available languages is fixed (10) and it can store them as an array (noSQL benefit).

Some of the tables were also added like tables for tokens and phone codes. Originally the project uses JWT tokens for authentication, which enables to create secure tokens with an expiration date, without needing to store it in a database. Also some third party services were used to identify secure codes which were sent to phone numbers. As for this project I had decided to add those tables, as it can suggest some alternative mechanism for email/phone validation and authentication.

# DDL:

**CREATE** **DATABASE** **IF** **NOT** **EXISTS** volunteersDB;

**USE** volunteersDB;

**DROP** **TABLE** **IF** **EXISTS** `admins`;

**CREATE** **TABLE** `admins`

(

`id` **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

`name` **VARCHAR**(40),

`surname` **VARCHAR**(40),

`email` **VARCHAR**(80),

`password` **VARCHAR**(20),

`type` enum ('general', 'super') **DEFAULT** 'general'

);

**DROP** **TABLE** **IF** **EXISTS** `tags`;

**CREATE** **TABLE** `tags`

(

`id` **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

`name` **VARCHAR**(20) **UNIQUE**

);

**DROP** **TABLE** **IF** **EXISTS** `languages`;

**CREATE** **TABLE** `languages`

(

`id` **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

`name` **VARCHAR**(20) **UNIQUE**

);

**DROP** **TABLE** **IF** **EXISTS** `volunteers`;

**CREATE** **TABLE** `volunteers`

(

`id` **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

`name` **VARCHAR**(40),

`surname` **VARCHAR**(40),

`email` **VARCHAR**(80),

`phone` **VARCHAR**(20) **DEFAULT** **NULL**,

`status` enum ('initialized', 'email verified', 'phone verified', 'completed') **DEFAULT** 'initialized',

`birthDate` **DATE** **DEFAULT** **NULL**,

`country` **VARCHAR**(40) **DEFAULT** **NULL**,

`city` **VARCHAR**(40) **DEFAULT** **NULL**,

`address` **VARCHAR**(255) **DEFAULT** **NULL**,

`specialization` **VARCHAR**(255) **DEFAULT** **NULL**,

`currentEmployerName` **VARCHAR**(255) **DEFAULT** **NULL**,

`occupation` **VARCHAR**(255) **DEFAULT** **NULL**,

`hoursPerWeekFrom` **INT** **DEFAULT** **NULL**,

`hoursPerWeekTo` **INT** **DEFAULT** **NULL**,

`workStatus` **VARCHAR**(255) **DEFAULT** **NULL**,

`facebookProfile` **VARCHAR**(255) **DEFAULT** **NULL**,

`linkedinProfile` **VARCHAR**(255) **DEFAULT** **NULL**,

`twitterProfile` **VARCHAR**(255) **DEFAULT** **NULL**,

`whereToVolunteer` **VARCHAR**(255) **DEFAULT** **NULL**,

`other` **VARCHAR**(255) **DEFAULT** **NULL**,

`note` **VARCHAR**(255)

);

**DROP** **TABLE** **IF** **EXISTS** `volunteerTags`;

**CREATE** **TABLE** `volunteerTags`

(

`volunteerId` **INT**,

`tagId` **INT**

);

**DROP** **TABLE** **IF** **EXISTS** `volunteerLanguages`;

**CREATE** **TABLE** `volunteerLanguages`

(

`volunteerId` **INT**,

`languageId` **INT**

);

**DROP** **TABLE** **IF** **EXISTS** `phoneCodes`;

**CREATE** **TABLE** `phoneCodes`

(

`phone` **VARCHAR**(20),

`code` **VARCHAR**(8),

`expires` datetime

);

**DROP** **TABLE** **IF** **EXISTS** `tokens`;

**CREATE** **TABLE** `tokens`

(

`id` **INT**,

`token` **VARCHAR**(40),

`expires` datetime

);

**CREATE** **UNIQUE** **INDEX** `admins\_index\_0` **ON** `admins` (`email`);

**CREATE** **UNIQUE** **INDEX** `admins\_index\_1` **ON** `admins` (`id`);

**CREATE** **UNIQUE** **INDEX** `tags\_index\_2` **ON** `tags` (`name`);

**CREATE** **UNIQUE** **INDEX** `tags\_index\_3` **ON** `tags` (`id`);

**CREATE** **UNIQUE** **INDEX** `languages\_index\_4` **ON** `languages` (`name`);

**CREATE** **UNIQUE** **INDEX** `languages\_index\_5` **ON** `languages` (`id`);

**CREATE** **UNIQUE** **INDEX** `volunteers\_index\_6` **ON** `volunteers` (`phone`);

**CREATE** **UNIQUE** **INDEX** `volunteers\_index\_7` **ON** `volunteers` (`email`);

**CREATE** **UNIQUE** **INDEX** `volunteers\_index\_8` **ON** `volunteers` (`id`);

**CREATE** **INDEX** `phoneCodes\_index\_9` **ON** `phoneCodes` (`phone`);

**CREATE** **UNIQUE** **INDEX** `tokens\_index\_10` **ON** `tokens` (`token`);

**ALTER** **TABLE** `volunteerTags`

**ADD** **FOREIGN** **KEY** (`volunteerId`) **REFERENCES** `volunteers` (`id`);

**ALTER** **TABLE** `volunteerTags`

**ADD** **FOREIGN** **KEY** (`tagId`) **REFERENCES** `tags` (`id`);

**ALTER** **TABLE** `volunteerLanguages`

**ADD** **FOREIGN** **KEY** (`volunteerId`) **REFERENCES** `volunteers` (`id`);

**ALTER** **TABLE** `volunteerLanguages`

**ADD** **FOREIGN** **KEY** (`languageId`) **REFERENCES** `languages` (`id`);

**ALTER** **TABLE** `phoneCodes`

**ADD** **FOREIGN** **KEY** (`phone`) **REFERENCES** `volunteers` (`phone`);

**ALTER** **TABLE** `tokens`

**ADD** **FOREIGN** **KEY** (`id`) **REFERENCES** `volunteers` (`id`);

# Queries:

Some of the queries may assume that the returned result is single. There could be queries where there is no LIMIT construct used for performance gain, as the database structure is well constructed. For example if we add where clause for field which is unique, we ensure that only one field could fit it and inner optimizations were included by default. Some of the queries could be organized otherwise. Often developers do not use primary SQL queries but use some Abstraction Model Layers, which were created by the shape of the table. And such layers include already made methods, which include some structured queries on it and can have hidden queries, which are not visible. The queries below show how that queries could be implemented, they can be alternatively written. All values which were used in queries were validated on the backend side, so there are no explicit checking and validation mechanisms on the DB side.

1. Get the status of the volunteer with the provided email. If it is null, then there is no volunteer. If it is completed then no volunteer could be registered. Else validation link will be generated and sent to the email. If the status is initialized new volunteers will be added.

**SELECT** **STATUS** **FROM** volunteers

**WHERE** email="mat98@sql.com";

1. Inserting new volunteer

**INSERT** **INTO** volunteers (name, surname, email)

**VALUES** ("Armen", "Matevosyan", "mat98@sql.com");

1. Updating status of the volunteer to email verified.

**UPDATE** volunteers

**SET** **STATUS** = 'email verified'

**WHERE** id=103;

1. Adding token with expiration date and id

**INSERT** **INTO** tokens

**VALUES** (103, "5cb138284d431abd6a053a56625ec088bfb88912", DATE\_ADD(NOW(), **INTERVAL** 1 **DAY**));

1. Get the id of the volunteer connected with the provided token. If the result is null, then there is no valid token.

**SELECT** volunteers.id **FROM** volunteers

**JOIN** tokens t **ON** volunteers.id = t.id

**WHERE** t.token = "5cb138284d431abd6a053a56625ec088bfb88912" **AND** t.expires <= NOW();

1. Get the status of the volunteer with the provided id.

**SELECT** **STATUS** **FROM** volunteers

**WHERE** id = 103;

1. Checking if a volunteer with a provided phone exists (1 - exists, 0 - not).

**SELECT** **COUNT**(**STATUS**) **FROM** volunteers

**WHERE** phone="+37498989898";

1. Adding code with expiration date and phone

**INSERT** **INTO** phonecodes

**VALUES** ("+37498989898", "8068", DATE\_ADD(NOW(), **INTERVAL** 1 **DAY**));

1. Checking if the number and code are valid

**SELECT** **COUNT**(code) **FROM** phonecodes

**WHERE** phone="+37498989898" **AND** code="8068" **AND** expires >= NOW()

**LIMIT** 1;

1. Add phone to the volunteer and update it’s status to phone verified

**UPDATE** volunteers

**SET** **STATUS** = 'phone verified', phone = "+37498989898"

**WHERE** id=103;

1. Check if the admin provided password and email exists.

**SELECT** id **FROM** admins

**WHERE** email="ktremblay@example.com" **AND** password="Lala2020";

1. Do wildcard search(\*.com\*) with the field (in this case email) on volunteers

**SELECT** \* **FROM** volunteers

**WHERE** email **LIKE** "%.com%";

1. Add note to volunteer

**UPDATE** volunteers

**SET** note = 'Some note.'

**WHERE** id=103;

1. Get all volunteers

**SELECT** \* **FROM** volunteers;

1. Get volunteer with provided id

**SELECT** \* **FROM** volunteers

**WHERE** id = 103;

1. Add new tag

**INSERT** **INTO** tags

**VALUES** (**DEFAULT**, "sport");

1. Add new languages

**INSERT** **INTO** languages

**VALUES** (**DEFAULT**, "english");

1. Get all tags of the volunteer with provided id

**SELECT** t.id, name **FROM** volunteertags

**JOIN** tags t **ON** t.id = volunteertags.tagId

**WHERE** volunteerId = 103;

1. Get all languages of the volunteer with provided id

**SELECT** l.id, name **FROM** volunteerlanguages

**JOIN** languages l **ON** volunteerlanguages.languageId = l.id

**WHERE** volunteerId = 103;

1. Get all volunteers with given tags

**SELECT** \* **FROM** volunteertags

**JOIN** volunteers v **ON** volunteertags.volunteerId = v.id

**JOIN** tags t **ON** t.id = volunteertags.tagId

**WHERE** t.name **IN** ("officiis", "doloribus");

1. Get all volunteers with given languages

**SELECT** \* **FROM** volunteerlanguages

**JOIN** volunteers v **ON** volunteerlanguages.volunteerId = v.id

**JOIN** languages l **ON** volunteerlanguages.languageId = l.id

**WHERE** l.name **IN** ("english", "russian");