

## **POPA EDUARD STEFAN**

### **Datavid Cake Tracker – internship task**

#### **Functionalities description**

The implemented functionalities are described below:

- 1) the insertion of a new member into the application's database;
- 2) the presentation of a list with all the members and their associated data;
- 3) the presentation of a list with the upcoming member's birthdays;
- 4) the possibility to update member's data if necessary (changing their location for example);
- 5) the deleting of members (if they are leaving the company);
- 6) the release of warning alerts if the user doesn't provide consistent information according to the provided rules;

#### **The technology stack**

I have developed a full stack solution in order to solve this problem.

The graphical interface was implemented by using the React library and a few additional npm packages. The web client runs on port 3000 and sends HTTP requests to the server and renders the received response data in a organized manner.

For the backend side I made use of the Node.js platform based on JavaScript programming language. The server runs on port 8080 and receives HTTP requests from the client in order to perform a certain task involving the access to the database.

All the member's data is stored in a SQLite relational database which is stored local.

#### **The project's installing requirements**

In order to be able to run the solution, the hosting system must have installed Node.js and npm (Node Package Manager). Both of them can be installed simultaneously from the Node official website (<https://nodejs.org/en/download/package-manager>) by downloading and executing the kit.

To run the backend server, use the next command in the "backend" folder: **node index.js**

To run the client, use the next command in the "frontend" folder: **npm start**

## The solution's architecture

### The backend project (Node.js)

The backend project consists of the next structure:

- node\_modules folder – has all the required packages in order to run the application;
- connectDB.js file – the script responsible with the connection with the database;
- controller.js file – the script with the database manipulation functions;
- database.db file – the database file with the stored info;
- index.js file – the starting point of the application;
- package.json / package-lock.js files – configuration files;
- router.js file – the file with the mapping of the endpoints and their actions;
- validator.js file – file with middleware functions that ensure the data consistency before accessing the endpoint;

The backend server's implemented endpoints are:

- **POST** /member – to insert a member in the database;
- **GET** /member – returns a list of all the members;
- **GET** /member/:id – returns the data of the member which has the provided ID;
- **GET** /upcomingBirthdays – returns the members ordered by their birthdays according to the systems current date;
- **PATCH** /member/:id – updates the member's data which has the provided ID;
- **DELETE** /member/:id – removes the member which has the provided ID;

The validation file contains middleware functions which performs the next checks:

- the member is at least 18 years old;
- all the provided fields are not null;
- the first name, last name and location doesn't already exist in a database row;

In the case of which at least one of these conditions are not met, a warning message is sent to the client.

### The frontend project (React)

After opening the web application the user will be able to access the following pages:

- localhost:3000/home – the page has a form in order to insert a new member;
- localhost:3000/members – all the member's data can be visualized in a table (the user has the option to delete or to update rows);

- localhost:3000/upcomingBirthdays – all the member’s data ordered by the period of time between the current date and the member’s next birthday;
- localhost:3000/memberInfo/:id – a completed form for the member with the ID specified in the URL in order to modify the fields;

## The database (SQLite)

The SQL script for the creation of table with members info is presented below:


```
CREATE TABLE IF NOT EXISTS MEMBERS (  
    MEMBER_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    MEMBER_FIRST_NAME NVARCHAR(30),  
    MEMBER_LAST_NAME NVARCHAR(30),  
    MEMBER_BIRTHDATE DATE,  
    MEMBER_COUNTRY NVARCHAR(30),  
    MEMBER_CITY NVARCHAR(50))
```

## Brief guide to application

The first page that is displayed is the one responsible for inserting a new member into the system. In order to ensure the format correctness of the birthdate, a date picker element was used instead of a simple text input. All the pages have the same navbar that allow moving from one page to another.

Home Members Upcoming birthdays

**Add a new member**

Member's first name Eduard-Stefan	Member's last name Popa
Member's country Romania	Member's city Bucharest
Member birthdate 03/25/2000 	

CREATE MEMBER

For this instance, the user already exists in the database and an alert is displayed below the “Create Member” button.

**Add a new member**

Member's first name  
Eduard-Stefan


Member's last name  
Popa

Member's country  
Romania

Member's city  
Bucharest

Member birthdate  
03/25/2000

CREATE MEMBER

 The member already exists in the database

Next up is the page where we can see all the member’s info.

Home Members Upcoming birthdays					
First Name	Last Name	Birthdate	Country	City	
Sebastian	Dumitru	1998-07-21	Romania	Galati	DELETE EDIT
Flavia	Spataru	1996-01-06	Romania	Alexandria	DELETE EDIT
David	Georgescu	2000-07-10	Romania	Pitesti	DELETE EDIT
Vlad	Constantinescu	2003-08-02	Romania	Ploiesti	DELETE EDIT
Eduard	Popa	1997-11-12	Romania	Bucharest	DELETE EDIT

By pressing the “Edit” button in the right part of the table, the user will be redirected to a new page where they can edit all the fields regarding the selected member.

**Update a member**

Member's first name  
Sebastian

Member's last name  
Dumitru

Member's country  
Romania

Member's city  
Galati

Member birthdate  
07/21/1998

UPDATE MEMBER

Also, the user can see the member's info ordered by the their upcoming birthdays as it is shown in the next image.

First Name	Last Name	Birthdate	Country	City
Paula	Radulescu	1995-06-18	Romania	Deva
David	Georgescu	2000-07-10	Romania	Pitesti
Sebastian	Dumitru	1998-07-21	Romania	Galati
Kate	Jordan	1992-07-31	England	Liverpool
Vlad	Constantinescu	2003-08-02	Romania	Ploiesti
Michael	Richards	1990-10-01	England	London
Edward	Rees	1997-11-10	Romania	Bucharest