HOMEWORK 2

IN MEMORY CRUD REST SERVICE WITH VALIDATION

PREREQUISITES:

1. Install the latest **LTS** (Long Term Support) version of **Node.js** (https://nodejs.org/en/), by any available means (.exe, nvm, brew, etc.).
2. Check in the console (terminal) that the Node.js installation was done properly by running the following commands `node -v` or `node -version`.
3. Create a repo for your homework tasks on **Github** (https://github.com/) or **git.epam.com**.
4. Provide your mentor with the link to the repo and add read access permissions.
5. Create package.json by running the following commands `npm init` or `npm init -y`.
6. Install globally or locally npm package **nodemon** (https://github.com/remy/nodemon), configure **babel** (https://babeljs.io/) and **eslint** (https://eslint.org/).

   Use the following **eslint** config file: https://epa.ms/nodejs19-hw2-ex1.

   As an alternative you can use TypeScript, this will be a big plus. Please inform your mentor if you decide to move on with TypeScript.
7. Get ready to watch the lectures and do the homework tasks to study the basic principles and approaches of development server-side applications with **Node.js**.

TASK 2.1

Write a simple **REST** service with **CRUD** operations for User entity.

- To create **REST** service, use **ExpressJS** (https://expressjs.com/).

  The User should have the following properties (you can use **UUID** as a user identifier (**id**)):

```
type User = {
  id: string;
  login: string;
  password: string;
  age: number;
  isDeleted: boolean;
};
```

- Service should have the following **CRUD** operations for **User**:
  - get user by **id**;
  - create and update user;
  - get auto-suggest list from `limit` users, sorted by login property and filtered by `loginSubstring` in the login property:

    `getAutoSuggestUsers(loginSubstring, limit)`
  - remove user (**soft delete** - user gets marked with `isDeleted` flag, but not removed from the collection).
- Store user's collection in the service memory (while the service is running).

To test the service **CRUD** methods, you can use **Postman** (https://www.getpostman.com/).

TASK 2.2

Add server-side validation for create/update operations of **User** entity:

- all fields are required;
- login validation is required;
- password must contain letters and numbers;
- user's age must be between 4 and 130.

In case of any property does not meet the validation requirements or the field is absent, return **400 (Bad Request)** and detailed error message.

For requests validation use special packages like **joi** (https://github.com/hapijs/joi, https://www.npmjs.com/package/express-joi-validation).

EVALUATION CRITERIA

2. Task 2.1 is partially implemented (w/o getAutoSuggestUsers or other methods).
3. Task 2.1 is fulfilled to the full extent.
4. **Task 2.1** eslint rules are applied.
5. **Task 2.2** is fulfilled to the full extent; validation package is used.

5*. *Consider to use Typescript.*