1 Practice 17.1. Token authentication and middelware

> This practice is the next of the series of practices, consisting of a Library application. Each practice will be based on the previous one. Before starting it, finish the previous one, make the corrections you consider, if needed, and start coding.
>
> Before start, create a new git branch named `books_review-u17` and switch to it:
>
> ```
> git branch books_review-u17
> git checkout books_review-u17
> ```
>
> Do all the practice in the `books_review-u17` branch. Once finished, do the submission via GitHub Classroom as detailed at the end of the practice.

1.1 Overview

In this practice we're going to do an the token authentication for the app. The app will have 3 access levels:

- **Public access**: can access **all the GET** methods and to **login** and **register** url's.
- **Registered users**: in addition to the previous permissions, they can **add reviews to any book**.
- **Admin users**: can access all the url's and can edit, delete and add all the data.

1.2 Exercise 1. Users

The authentication must be based in a **user** document (table). The users will be stored using a model named `User` in a file named `user.js`.

The user fields to be stored are:

- Name: String, required, unique, minimum length 4 characters.
- Email: String, required, unique, email format.
- Password: String, required, minimum length 8 characters.
- Role: String, default value `'user'`.

You can validate the email format using regular expressions or using the validator library.

The passwords must be store encrypted. Store and check the password using the **Bcrypt.js** library.

The users must be registered using the `/users/register` route. Each new user needs to send the `name`, `email` and `password` fields. The `role` field must be added automatically with the `'user'` value.

A registered user can get an authentication token using the `/users/login` route, sending their email and password.

1.3 Exercise 2. Authorization

As seen in the Overview section, each user role has the next permissions:

- **Administrators**, with the role `admin`. Can do all the operations. Optionally, an admin can change the users data.
- **Registered users**, with the role `user`. Only can write a new review to an existing book and can view all the data of the `books` collection (optionally, a user can view their own `user` data and change their own password).
- **Unauthenticated users**. Can view all the `books` data, except for the `user` entity data.

> To make an admin user, you can change the role field of an existing user with the next command in the MongoDB console:
>
> ```
> db.users.updateOne({_id: ObjectId('your_user_id')}, {$set: {role:
> ↪  'admin'} })
> ```

Make the authentication with a **middleware function** in a file named `authentication.js`.

1.4 How to submit to GitHub Classroom

Once you finish the task, make a commit with the comment “PRACTICE 17.1 SUBMISSION COMMIT” and push it to GitHub. Make a **pull-request** so that i could know your code is submitted.

1. This task must be submitted to the same repository of the previous one. Remember to make a new branch before starting the practice.
2. Once you finish the task, make a commit with the comment “PRACTICE 17.1 SUBMISSION COMMIT”, merge the branch into the main branch, and push it to GitHub. For example:

```
git commit -m "PRACTICE 17.1 SUBMISSION COMMIT"
git checkout main
git merge books_review-u17
git push
```

3. It’s recommended to tag your commit with the tag “Practice_17.1”.

4. Before that, you can do the commits and push you want. If you change your code after your submission commit, make another commit and push with the same text in the message adding the corrections you've done.

If you have any doubt in your task, you can push your code and ask me by email what's your problem. It will make it easier for both the solutions of code issues.