sword

API Developer Practical Exercise

Implementation description and comments:

- Created a Database called "tasks" for development purposes and stored it's name on environment variable called DB_NAME. Later on this can be changed for production;
- Cloned the database called "tasks" and renamed it as "tasks-test" for unit testing purposes;
- Decided on using just 2 tables. One called "users" other called "tasks";
- Unique key, besides each table's ids, are the username and email;
- The table "tasks" has a Foreign Key named userId that stores the id of the user that created a task;
- On register, each user has to fill in "manager" or "technician" for the "position" column. I sat this as VARCHAR, instead of TINYINT(boolean) since further down the line we can have other different positions;
- Not everyone can register as manger. To register as manager, you have to provide the correct "Admin Password" which I sat to be the DB_PW on the environment variables;
- On register, each Technician must provide his manager email;
- All the created managers will still have access to the complete users and tasks tables and also on the endpoint '/users/team' will have access to all his technicians which have the "manager_email" column filled and matching the manager sending the request to that endpoint;
- All passwords are Hashed by Bcryptjs before being stored in the Database;
- The app is using jsonwebtoken to deal with authentication;
- On the development I'm setting the created jsonwebtoken after login on the cookies, I find it easier to implement afterwards on the frontend and of course this token could be stored in the database also. For the testing I'm adding the Bearer token to the database on user creation and then getting it from the headers to authenticate the user.
- For simplicity and readability I created, as middleware, authManager for managers and authTechnician for technicians. Then I can use whichever is fit it for each of the routes;
- For unit testing I used Jest and Supertest and just included testing for each of the endpoints;
- For Manager notification when a task was completed I used a free emailing api called Sendgrid (https://sendgrid.com/). For this exercise I used my personal email with my sendgrid api key. On the real world an admin email could be used;
- For unit tests purposes I created a mock without api key so no emails are send on unit tests.
- Since the task description can be up to 2500characters long, I chose to not mention the description on the email, instead I'm just sending the task Id (only for simplicity);
- For this exercise I'm sending the .env files along to GitHub;

X I wasn't able to solve the ECONNREFUSED error on the Docker Container creating. Seems like I can't connect to the database image when I start the application due to database connection port conflict (127.0.0.1:3306) I could not solve.

X No validation of user input fields was implemented for this exercise. Could do it manually or with validate.js

Api endpoints:

On src/routes/userRouters.js:

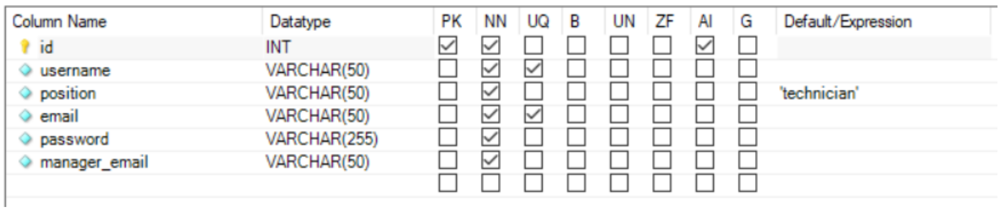
- '/users': method GET – for the manager to get the whole users table on the database;
- '/users/team': method GET – for the manager to get all his team members;
- '/login': method POST – for login;
- '/logout': method POST – for logout;
- '/register': method POST – for register new user;

On src/routes/taskRouters.js:

- '/tasks': method GET – for the manager to get all the tasks;
- '/tasks/user/:id': method GET – for the logged technician get his own tasks;
- '/tasks/add: method POST – for the technician add a task;
- '/tasks/completion/:id': method PATCH – for the technician to mark a task as complete;
- '/tasks/description/:id': method PATCH – for the technician to change the task summary;
- '/tasks/delete/:id': method DELETE – for the manager to delete a task;

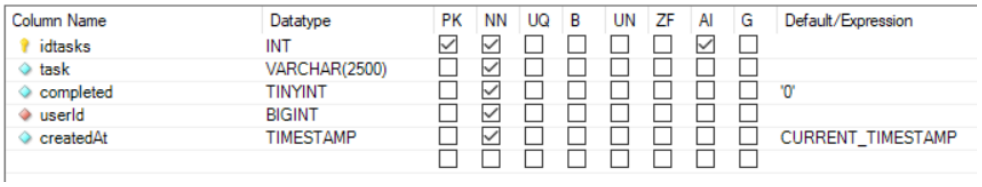
Tables:

Users table:



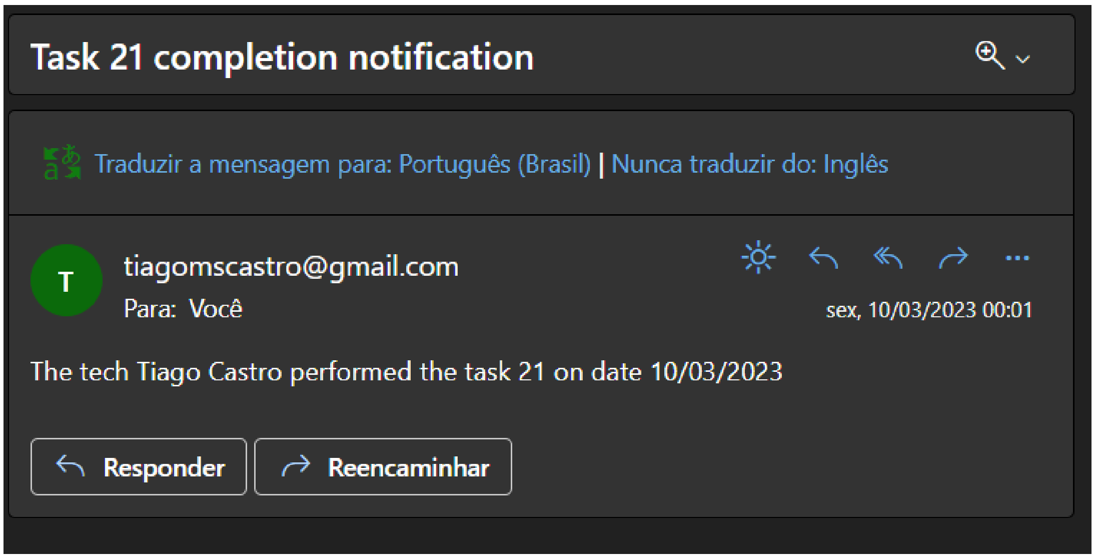
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓	☐	☐	☐	☐	✓	☐	
username	VARCHAR(50)	☐	✓	✓	☐	☐	☐	☐	☐	
position	VARCHAR(50)	☐	✓	☐	☐	☐	☐	☐	☐	'technician'
email	VARCHAR(50)	☐	✓	✓	☐	☐	☐	☐	☐	
password	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	☐	
manager_email	VARCHAR(50)	☐	✓	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Tasks table:



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idtasks	INT	✓	✓	☐	☐	☐	☐	✓	☐	
task	VARCHAR(2500)	☐	✓	☐	☐	☐	☐	☐	☐	
completed	TINYINT	☐	✓	☐	☐	☐	☐	☐	☐	'0'
userId	BIGINT	☐	✓	☐	☐	☐	☐	☐	☐	
createdAt	TIMESTAMP	☐	✓	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
		☐	☐	☐	☐	☐	☐	☐	☐	

Manager Notification:

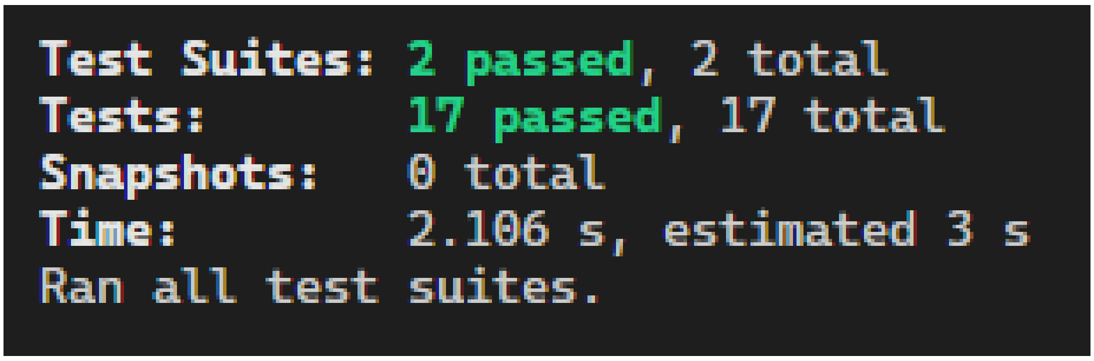

Task 21 completion notification

Traduzir a mensagem para: Português (Brasil) | Nunca traduzir do: Inglês

tiagomscastro@gmail.com
Para: Você
sex, 10/03/2023 00:01

The tech Tiago Castro performed the task 21 on date 10/03/2023

Jest and Supertest results (on the console, inside src folder -> npm run test):


```
Test Suites: 2 passed, 2 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        2.106 s, estimated 3 s
Ran all test suites.
```

Tests runned at userRouters.test.js:

- Should login existing user
- Should not login non-existent user
- Should not login with wrong password
- Should register a new Technician
- Should register a new Manager
- Should not register a manager without Admin Password
- Manager should see all technicians/users

Tests runned at taskRouters.test.js:

- Technician should see only his own tasks
- Technician should create a new task
- Manager should not create a new task
- Technician should complete a task
- Manager should not complete a task
- Technician should update task description
- Manager should not update task description
- Technician should not delete a task
- Manager should delete task