README

Data Processing Project

Caterina Aresti  
Joey Krämer  
Nicanor Martinez  
Zsombor Hajzer

2024

# README

Table of content

[README 1](#_Toc156682034)

[Prerequisites 2](#_Toc156682035)

[.env 2](#_Toc156682036)

[Database 3](#_Toc156682037)

[Backend 3](#_Toc156682038)

[Frontend 3](#_Toc156682039)

[Documents 4](#_Toc156682040)

## Prerequisites

The following programs need to be installed or created before using our project.

* Node
* Docker
* .env file

## .env

The .env must contain all of these items:

POSTGRES\_USER=postgres

POSTGRES\_PASSWORD=password

POSTGRES\_HOST=localhost

POSTGRES\_DB=netflix

POSTGRES\_PORT=5432

JWT\_SECRET=secret

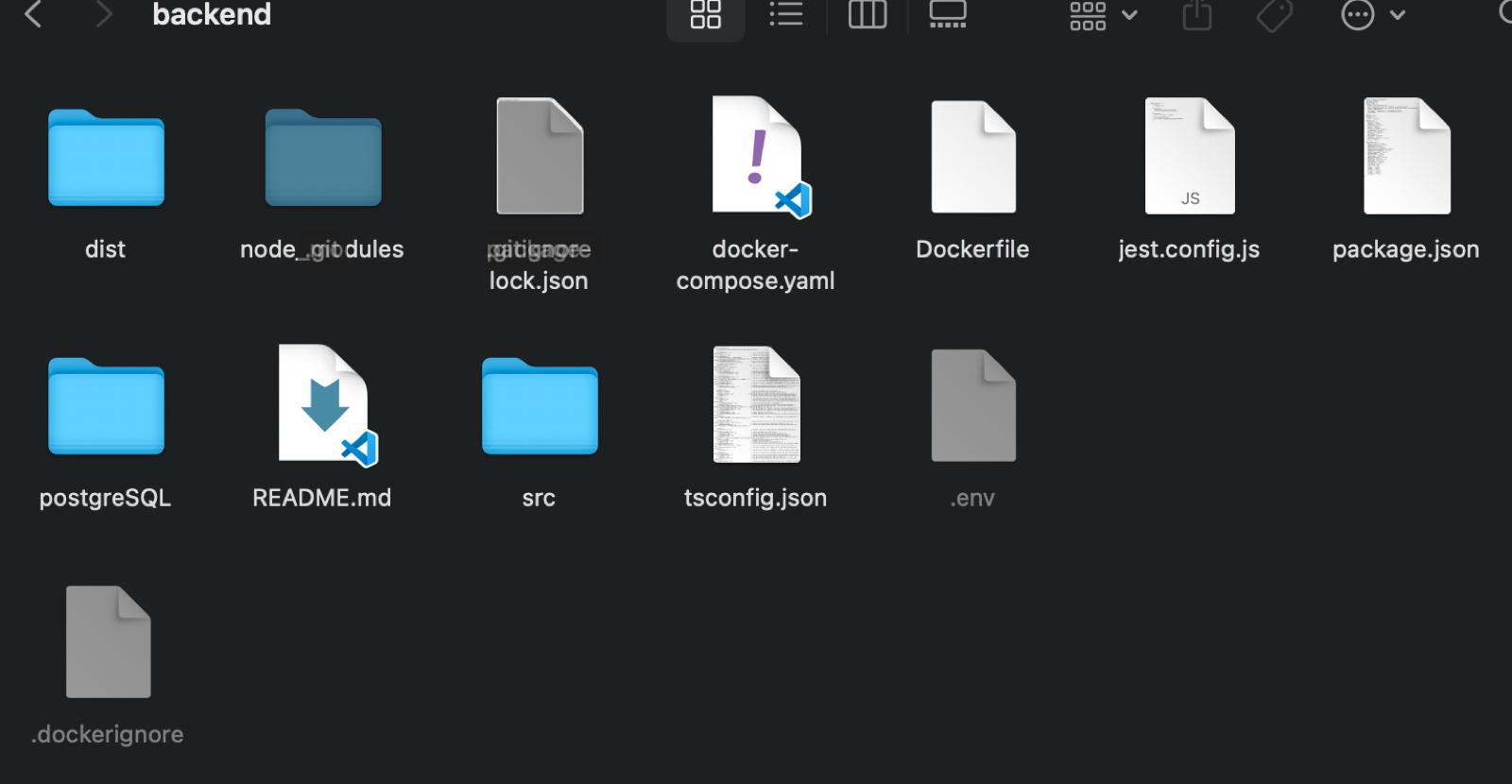
MAILING\_SERVICE\_ADDRESS = nhlstenden.work@gmail.com

MAILING\_SERVICE\_APP\_PASSWORD = ieykdgjbgcitkdpr

EMAIL\_API\_KEY = **580b14682491403ba15bc31a29301966**

PORT = 3000

Please place the .env file on the same level as the docker-compose.yaml file. It should look like the image below:



## Database

The database is hosted locally in a docker container. You can find the docker-compose.yaml with the Postgres database in the GitHub/Backend. The database can be found in script in the postgreSQL folder.

1. Create a database with the name ‘netflix’ (otherwise the API is not working)
2. Run the script in the following order:
   1. Netflix.sql
   2. Stored-procedures.sql
   3. triggers.sql
   4. Privileges.sql

Now you have a functional database without data.

Please execute dummy\_data.sql to insert necessary data to make frontend run for show case and give the database the necessary data to display some postman testing.

## Backend

The backend runs on node.js on port 3000. To start the project please open a terminal window and find yourself in the correct directory.

1. Clone the repository into your selected folder

2. Open VSCode in the project-setup branch

3. Create a .env file in the root folder

4. Open a termial in VSCode

5. Run the following code to download all the necessary packages

```npm run i```

6. Build the project (necessary for the creation of a dist folder from which our server runs from)

```npm run build```

7. From here on out, the server can be run with either

```npm start```

or

```npm run dev```

8. if there are any problems with dependencies, manually uninstall them and install them again with the following commands:

```npm uninstall <packagename>```

```npm install <packagename>```

Notes for the backend use case:

1. All of the necessary requests can be found in the exported postman file. In order to use this file, import it into your own postman environment and change the environment variables to your personal ones in order to make its work. Fake email addresses are not going to work, password can be an actual password with one capital, one lower case letter, one number and one special character, and at least 6 characters long OR it can be just left as "password" which was left in to make testing easier.

2. The API for email validation is using a free tier service which only allows 100 requests maximum. after this expires the email validation will throw errors. If you would like to continue testing on the backend, go to /src/utils/validators.ts and comment out everything between lines 11-24.

3. The backend is using a real emailing service in order to sign up, verify, reset password or to invite someone. For this real emails are necessary to use and links need to be clicked in order to verify or to invite someone. Gmail allows around 300 emails per day which should be plenty.

4. Some backend interactions needs to follow specific flows of action. For example:

- To create an account to use the restricted part of the backend. (routes that need authentication)

- put your actual email as a global variable in postman -> call Register JSON->verify email (click on link in inbox)->after this account can make requests in other parts of the postman environment.

- To invite users and get -2-euro discount in the database

- Have an already verified account with a **\_\_subscription\_id\_\_** of **\_\_1\_\_** OR **\_\_2\_\_** OR **\_\_3\_\_** -> invite a second email address by calling the Invite Sending request located in the profile folder -> open email address and click on the invitation link -> use the second email to call the Register JSON call in postman (make sure **\_\_subscription\_id\_\_** is **\_\_1\_\_** OR **\_\_2\_\_** OR **\_\_3\_\_**) -> access db and in table **\_\_account\_subscription\_\_** you should see column account\_id and **\_\_price\_\_** these should be 2 euros smaller than the prices available in table **\_\_subscription\_\_**

5. Some of the use case scenarios could not be done logically without a frontend for it such as password reset(it would require redirects). When **\_\_Forgot Password\_\_** is called it also sends an email with a link. extract the JWT token from this request and use it in the patch request **\_\_New Password submit\_\_** in order to reset a password.

6. All test can be found in the postman environment. Since setting up different scenarios is not possible here, the comment above the test should explain what it is for or the description string in the first line of the test.

7. Backend is taking advantage of headers in the use case. It is capable of responding in both JSON and XML. This can be done by changing the default Accept header to application/xml (it will default to json). Some separate requests were made in postman to showcase this, but we did not duplicate all of them.

Accepted languages header is also used to infer the language the user uses when creating a profile. This can also be changed manually, if nothing is sent it will default to "en" (English)

8. Image upload functionality also works, all images can be found uploaded into to /dist/images/. In order to differentiate images with the same name, a Unix timestamp is attached to the image name showing when an image was created. If an image is not submitted it will default to "default.jpeg"

## Frontend

The frontend runs on node.js on port 4000.

For **Linux/Mac users** please change line 18 from frontend/netflix/package.json

"start": "set PORT=4000 && react-scripts start",

to

"start": "PORT=4000 react-scripts start",

Commands:

1. npm i
2. npm start

All permissions are applied on the frontend. Seniors, Mediors and Juniors have different dashboards.

Admin example accounts to login:

Senior:

Email: [senior@example.com](mailto:senior@example.com)

Password: password

Medior:

Email: [medior@example.com](mailto:medior@example.com)

Password: password

Junior:

Email: [junior@example.com](mailto:junior@example.com)

Password: password

## Documents

All documents can be found on GitHub/Frontend.

Document item list:

* Diagrams folder
  + Versioning of ERD and class diagram
* Netflix\_openAPI.yaml (openapi 3.0 specification)
  + <https://editor.swagger.io/> has been checked with this tool
* SQL\_backup.docx