How To Set Up and Run the Backend Project

System Requirements

This application has been tested on a 2014 MacBook (https://support.apple.com/en-us/111942, Accessed on: 27 Nov 2024), confirming it as the minimum system requirement. Testing on an M1 MacBook also confirmed compatibility. While Node.js and PostgreSQL work on Linux and Windows, this guide focuses on macOS. Unix-based users can follow along by using the official Node.js and PostgreSQL documentation for the installation steps.

Minimum System Requirements

- **OS**: macOS Ventura, version 13
- **CPU**: 2.6GHz dual-core Intel Core i5 processor
- **Memory**: 8GB
- **Storage**: 128GB

Troubleshooting

Node.js may introduce breaking changes, particularly in major version updates. If the backend application fails to run, install Node.js version 22.4.1.

For macOS, detailed instructions can be found at the following URL: https://katopz.medium.com/how-to-install-specific-nodejs-version-c6e1cec8aa11 (Accessed: 2 Dec 2024).

Prerequisites (macOS)

Ensure Homebrew is installed. If it is not, it can be installed using the following command:

```
/bin/bash -c "$(curl -fsSL
<https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"
```

Openssl Latest Version (macOS)

macOS comes with openssl pre-installed but sometimes, the pre-installed openssl is outdated. Please refer to https://formulae.brew.sh/formula/openssl@3 (Accessed: 6 Dec 2024) to check the latest openssl version available.

Please, run this command to check the version installed on your macOS:

```
openssl -v
```

If the version is outdated, please run this command to install openssl from brew:

```
brew install openssl
```

Please, check the version again:

```
openssl -v
```

If it's still not updated, please use this command:

```
brew upgrade openssl
```

Setting up the Environment Variables (macOS)

Use openssl to generate a RSA private key:

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:4096
```

Use the private key from the previous step to generate a RSA public key:

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Copy the contents of the keys and set them as environment variables in either `.zprofile`, `zshrc`, or `.bash_profile` depending on the OS version:

```
export PHOTO_ALBUM_PRIVATE_KEY={the content of the private key file}
export PHOTO_ALBUM_PUBLIC_KEY={the content of the public key file}
```

For more information on setting environment variables, please refer to https://medium.com/@craakash/setting-permanent-environment-variables-in-macos-catalina-and-later-using-zsh-37b0273d980e (Accessed: 6 Dec 2024).

Install PostgreSQL (macOS)

Install PostgreSQL using Homebrew:

```
brew install postgresql
```

Setting Up the Database Locally (Any Unix-based System)

1. Start PostgreSQL Service

Start the PostgreSQL service to ensure it runs in the background:

```
brew services start postgresql
```

2. Create the Database

Create a database named **photo_album** using the PostgreSQL command line:

```
createdb photo_album
```

3. Verify Database Creation

Check that the **photo_album** database has been created successfully:

```
psql -l
```

4. Access the Database

Access the **photo_album** database:

```
psql photo_album
```

5. Create a Database User

Create a new user with a username and password:

```
CREATE USER [user_name] WITH PASSWORD '[password]';
```

It's recommended to use "hello" for username and "password" for password to avoid reconfiguring **config/db.config.js**:

```
CREATE USER hello WITH PASSWORD 'password';
```

6. Grant Privileges

Grant the new user privileges to the **photo_album** database:

```
GRANT ALL PRIVILEGES ON DATABASE photo_album TO [user_name];
```

If you follow the recommendation:

```
GRANT ALL PRIVILEGES ON DATABASE photo_album TO hello;
```

7. Quit PostgreSQL Shell

Exit the PostgreSQL shell:

```
\q
```

8. Verify User and Privileges

Connect to the **photo_album** database as the new user to verify:

```
PGPASSWORD=[your_password] psql -U [user_name] -d photo_album
```

If you follow the recommendation:

```
PGPASSWORD=password psql -U hello -d photo_album
```

If you see this on the left of the cursor in the command line, the newly created user is all set:

```
photo_album=>
```

9. Quit PostgreSQL Shell

Exit the PostgreSQL shell:

```
\q
```

Install Node.js (macOS)

Install Node.js using Homebrew:

```
brew install node
```

Verify Node.js and npm Installation

Check the installed versions of Node.js and npm to ensure they were installed correctly:

```
node -v
npm -v
```

Setting Up Node.js Dependencies (Any Unix-based System)

1. Navigate to the Project Directory

Navigate to the directory where your project is located:

```
cd /path/to/your/project
```

2. Install Project Dependencies

Install all the dependencies specified in the **package.json** file:

```
npm install
```

Creating the Database Using Node.js (Any Unix-based System)

Run the following command to finalize the database creation:

```
node create_db.js
```

This process may take a few minutes. Wait until the command completes before proceeding.

Start the Application (Any Unix-based System)

Run the application:

```
node app.js
```

This command will display the port number for accessing the application.

Confirm the App Is Working

Open a web browser and navigate to "**http://localhost:[port]**", where **[port]** is the number displayed in the previous step.

Database Configuration in Node.js (Any Unix-based System)

The configuration for the local environment is in **config/db.config.js**. If you created the database user with a different username and password than recommended, please adjust that file.

config/db.config.js is used by **models/index.js**.

If the app is run with the argument **release**, it doesn't use **config/db.config.js** as it's a setup for Heroku.

Locally, the app is run with:

```
node app.js
```

But on Heroku, it's run with:

```
node app.js release
```

The database setup written in **models/index.js** is provided by Heroku: https://help.heroku.com/QD1AIH8R/how-do-i-use-sequelize-to-connect-to-heroku-postgres (Accessed on: 27 Nov 2024).

Heroku login may be required to access this page.