LDN8 Structured Query Language (4)

2022-06-25

<CODE>YOUR FUTURE

Module Overview

1. Introduction to SQL (Jack)
   Basic data processing using PostgreSQL
2. NodeJS SQL Programming (Abdi)
   NodeJS PostgreSQL Integration
3. NodeJS CRUD with REST (Shakil)
   Connect your database to Internet by REST
4. Review
   From API to SQL a Heroku Walkthrough

You should have completed the followings before class

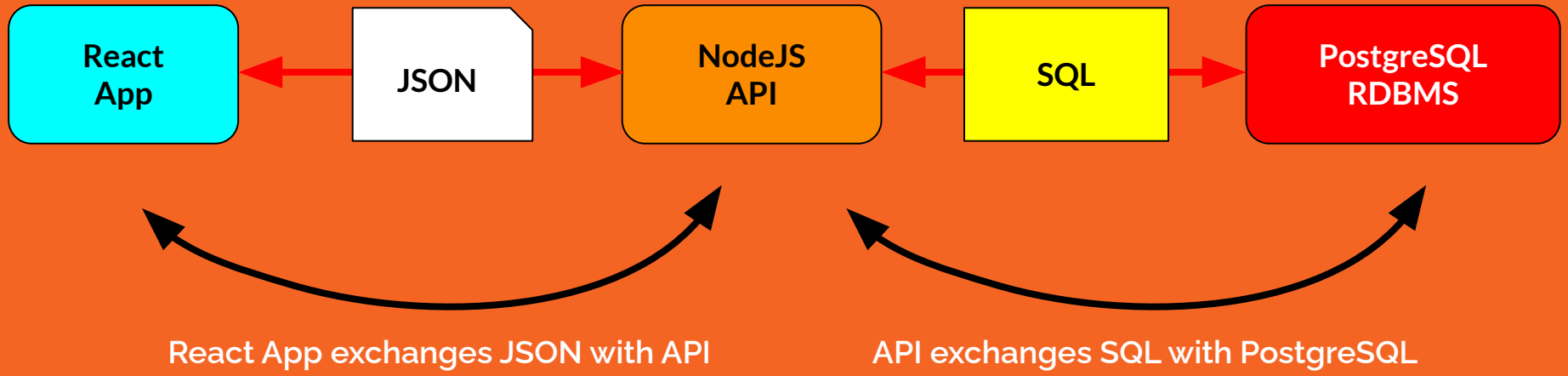
- Registered for Heroku with 2FA authentication
- Installed Postman in your machine
- Installed and authenticated Heroku CLI
- Created your own cyf-hotels-api project at Github and cloned locally
- Downloaded the cyf_hotels_exercise5.sql in your computer's Downloads folder
- $ means its a terminal command

Lesson Overview

Time	Activity
10:05 - 11:00	Recap and Heroku Setup
11:00 - 11:15	Break
11:15 - 12:30	Testing Heroku App
12:30 - 13:30	Lunch
13:30 - 14:30	Q & A

Recap 3-tier Full Stack Application

- **Separation of concern!**



Why Use Heroku?

- Its free :)
- Host your NodeJS API with PostgreSQL RDBMS
- Integration with your Github repository
- So that you can work your final project together!

Heroku Preparation (0)

- Modify NodeJS API as follows to work in Heroku platform
- 🚨 Bind to a port dynamically assigned by Heroku

```
let port = process.env.PORT || 3000;
```

- 🚨 Specify Pool's connectionString property and bind it to env var DATABASE_URL

```
const pool = new Pool({
 connectionString: process.env.DATABASE_URL,
 ssl: {
   rejectUnauthorized: false
 }
})
```

Heroku Preparation (0)

- connectionString should be encoded as follows:

```
postgres://<USERNAME>:<PASSWORD>@<HOST>:<PORT>/<DB>
```

For example

```
postgres://user1:secretPGSQL@some-server.eu-west-1.com
pute.amazonaws.com:5432/projectDB
```

- To set env. variable in Linux / macOS

```
$ export PORT=9999
$ export DATABASE_URL=<postgres:// ……>
```

Exercise 1

Modify cyf-hotels-api project server.js and test locally:

- change the listening port and connectionString by using environmental variables

```
$ PORT=9999 DATABASE_URL=<postgres://…> npm run start
```

- Windows

```
c:\> set PORT=9999
c:\> set DATABASE_URL="<postgres://…>"
c:\>npm run start
```

- Append ?sslmode=disable to the connection string if_ connection fails

Heroku Preparation (1)

- Registration and 2FA
- Install Heroku CLI
- Create a new application at Dashboard
- Select “Deploy” > “Deployment Method” > “Github”
- Select “Settings” > “Build Packs” > Select “NodeJS”
- Select “Resources” > “Add-ons” > “Heroku Postgres”

Heroku Preparation (2)

- Click on the Heroku Postgres link
- “Settings” > “Database Credentials”
- “URI” this is the PostgreSQL connection string i.e.

  connectionString in your NodeJS API

Note for PSQL users: Heroku periodically changes DB credentials so always check from dashboard if external connection fails

Accessing Heroku PostgreSQL (1)

- Click on the Heroku Postgres link
- “Settings” > “Database Credentials”
- “URI” is the PostgreSQL connection string

```
postgres://<USERNAME>:<PASSWORD>@<HOST>:<PORT>/<DB>
```

- You can access the PostgreSQL via PSQL or

```
$ psql -h <HOST> -p <PORT> -U <USERNAME> -W <DB>
// provide password when prompted
```

Accessing Heroku PostgreSQL (2)

- Use the Heroku CLI to access the PostgreSQL

```
$ heroku pg:psql -a <HEROKU_APP>
```

Exercise 2

- Connect Heroku's PostgreSQL by using
  - PSQL
  - Heroku CLI
- Load the cyf_hotels_exercise5.sql to the Heroku PostgreSQL DB

```
$ heroku pg:psql -a <HEROKU_APP>
-f ~/Downloads/cyf_hotels_exercise5.sql
```

- Count the number of records in each table

Deploying Heroku Application

- Automatic Deploys - deploy whenever change is committed in the main branch
- Manual Deploy - select a branch and deploy manually

Accessing and Examining Heroku Application

- Heroku application URL

  https://<HEROKU_APP>.herokuapp.com

- Use Heroku CLI to display all env variables

```
$ heroku config -a <HEROKU_APP>
```

Developing Heroku Application Locally

- Assume your are developing API “ABC”
- Open project ABC in VS Code
- Open Terminal within VS Code. Set DATABASE_URL env. var. and run node

```
$ DATABASE_URL=postgres://<USERNAME>:<PASSWORD>@<HOST>:<PORT>/<DB>
npm run start
```

- Make sure the above command is entered in one line

Testing Your Heroku Application by Postman

- Make sure you test all CRUD endpoints
- Make good use of Postman's Collection variable feature to simplify your life
- Instead of giving full URL in each request. Define scheme, host and endpoint variables in collection

  {{scheme}}://{{host}}/{{endpoint}}

  https://<HEROKU_APP>.herokuapp.com/customers

Exercise 3

- Test your cyf-hotels-api by using Postman
  - Create API collections and use variable to store the scheme, host and endpoint names and reuse in different API endpoints
- Test CRUD for customers endpoint to
  - Create a new customer record
  - Retrieve all customers records
  - Update the newly created customer record
  - Delete the newly created customer record

Exercise 4

- Edit customers GET endpoint. Add `console.log` statement to log the total number of records returned from DB. Push and re-deploy to test
- Use `$ heroku logs -t -a <HEROKU_APP>` command to check application logs and observe logged statements

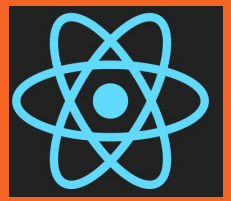
Development Summary

netlify

HEROKU

https://<web>.netlify.com

https://<api>.herokuapp.com



CORS !!

POSTMAN

PSQL

Questions and Answers