Thneyti - Tunisian Public Transportation REST API

IT325 Web Services Final Project

by

Sarra Koussaier

January 2022

IT/BA Junior Student

Information Technology Major

Tunis Business School

Ben Arous,TUNISIA.

2021-2022

Declaration of Academic Ethics

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: January 24th, 2022 Sarra Koussaier

Abstract

Adoption of digital technologies has accelerated largely in the last decade and has reached a critical stage today. [1]Tunisia alone is a driving force of the digital economy in North Africa and across the African continent. The digital economy accounts for 11 per cent of Tunisia's gross domestic product, making it one of the strongest and fastest growing sectors in the country. [2] However, even with this growth, we are still behind in many aspects of the digitisation.

An undeniable example of that is public transportation. One of the reasons for this is that on many buses, the information (destination, point of departure, and the route number) is only displayed in Arabic. [3] No guidance, manuals, or apps that could help tourists or anyone who's new to the city are available. This presents an indisputable drawback to the Tunisian transportation sector.

Keywords - Digitisation, Public transportation, Tunisia.

Contents

Chapter 1

Introduction

My final Project for the IT325 Web Services course this semester consists of a RESTful API developed using Node.js, Express, JWT, HTML, EJS, and CSS. The resources were stored thanks to the implementation of PostgreSQL and ElephantSQL.
I have also used technologies such as Heroku, NGINX, Postman, VSCode, and Git-Version Control to maximize the project's quality.
This project contains all the CRUD Operations, secured with JWT token authentication, SSL Certificates (by Heroku), and SSH Keys. Multiple Postman Snippets were also used in order to test this project's efficiency.
This projects aim is to facilitate the information sharing of the Tunisian public bus transportation data in a secure manner.
Its main function is returning a desired bus station name to someone who wishes to travel from one station to the other and is clueless about the Tunisian public bus transportation system.

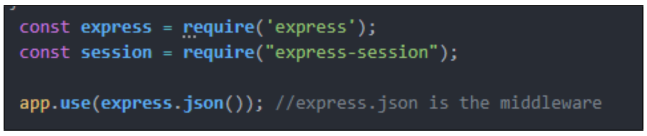
Chapter 2

Explanation of the work carried out

2.1 Node.js/EXPRESS Contribution

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. [4]
Express is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. [5]

I have used the Express framework for Node.js to develop my app.
Implementation code :

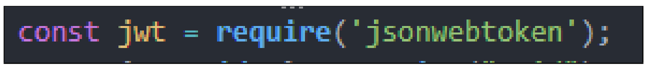


```
const express = require('express');
const session = require("express-session");

app.use(express.json()); //express.json is the middleware
```

2.2 JWT Contribution

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. [6]

I have used JSON Web Token authentication to secure the API access. Implementation code :



```
const jwt = require('jsonwebtoken');
```

2.3 EJS/HTML/CSS Contribution

EJS is a simple templating language which is used to generate HTML markup with plain JavaScript. It also helps to embed JavaScript to HTML pages. [7]

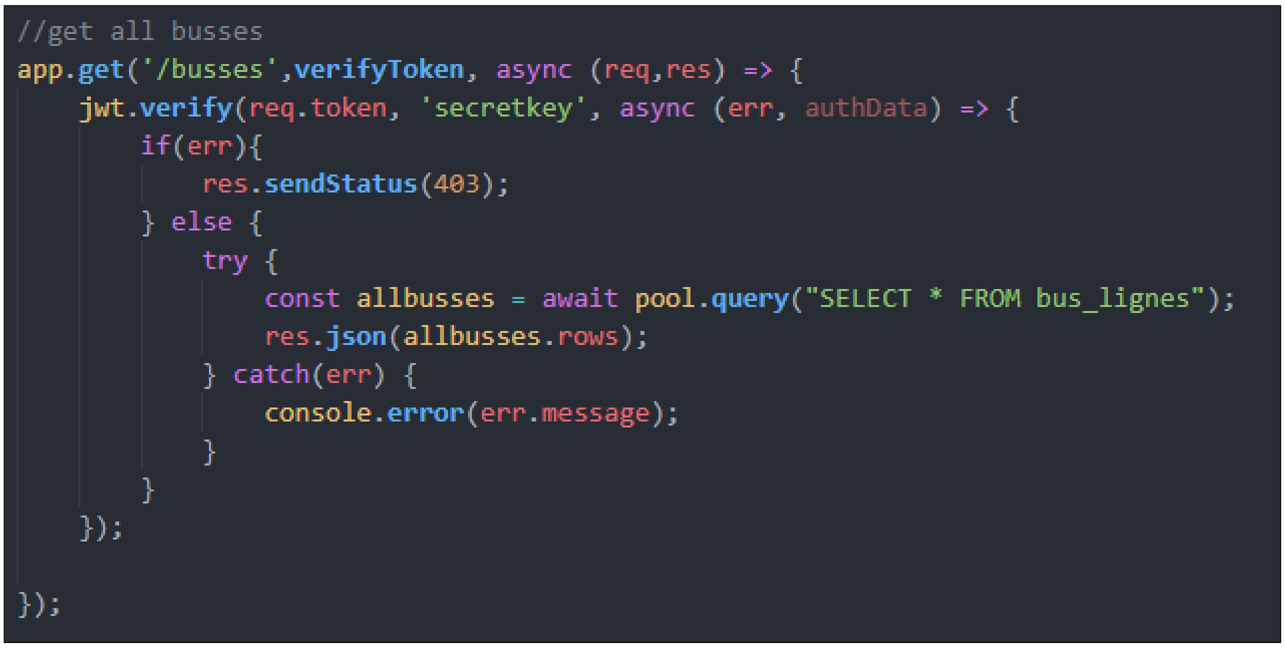
I used HTML,CSS, and EJS to design the interactive web pages used in my app. Example

2.4 All the HTTP Methods used

2.4.1 GET Requests

GET /busses

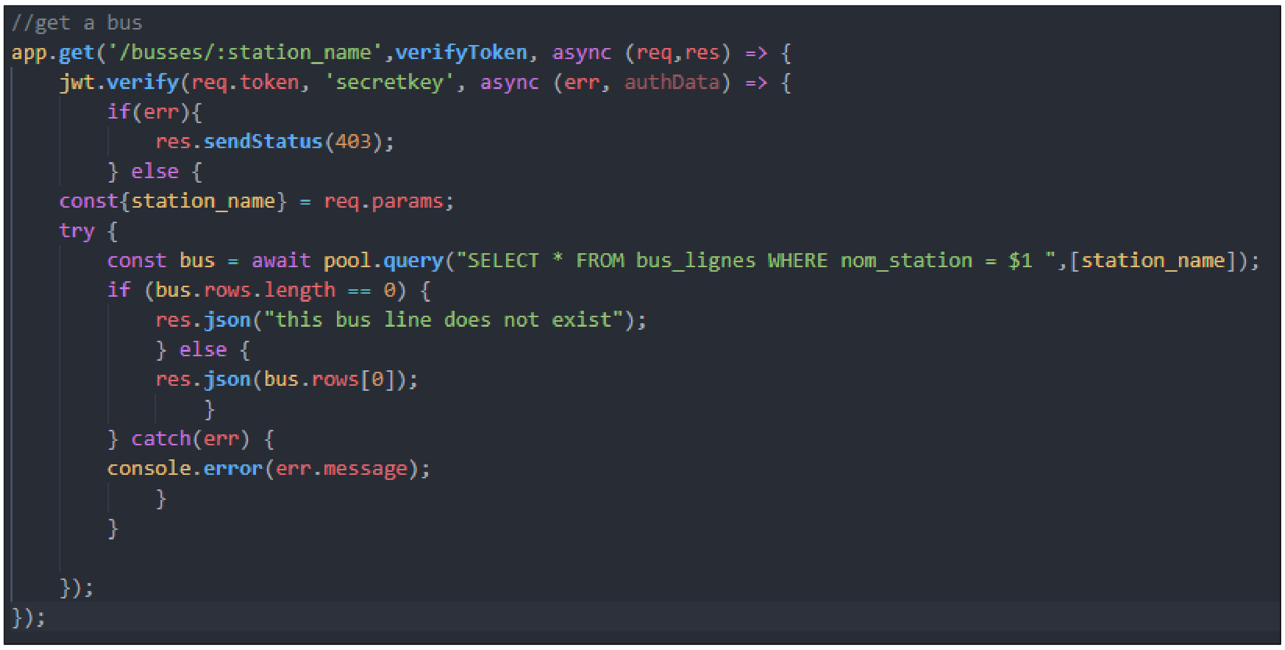
Get the station name, number of the station,and line number for all the Tunisian public bus stations available.



```
//get all busses
app.get('/busses',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
            try {
                const allbusses = await pool.query("SELECT * FROM bus_lignes");
                res.json(allbusses.rows);
            } catch(err) {
                console.error(err.message);
            }
        }
    });

});
```

GET /busses/StationName

Get the station name, number of the station,and line number for a specific Tunisian public bus station and the station name in the request URI.

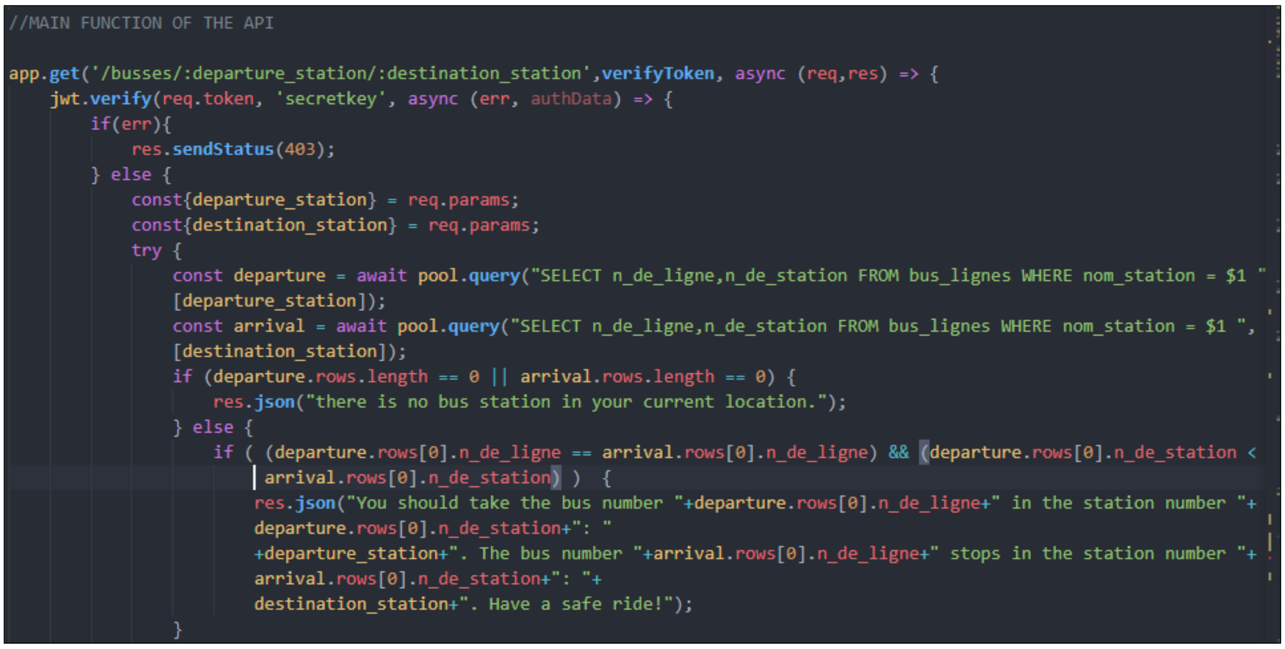


```
//get a bus
app.get('/busses/:station_name',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
    const{station_name} = req.params;
    try {
        const bus = await pool.query("SELECT * FROM bus_lignes WHERE nom_station = $1 ",[station_name]);
        if (bus.rows.length == 0) {
            res.json("this bus line does not exist");
            } else {
            res.json(bus.rows[0]);
                }
        } catch(err) {
        console.error(err.message);
            }
        }

    });
});
```

GET /busses/departureStation/arrivalStation

Get the station name, number of the station,and line number for the specific Tunisian public bus that you should take after specifying the departure and destination locations in the body of the request and the departure and arrival station names in the request URI.

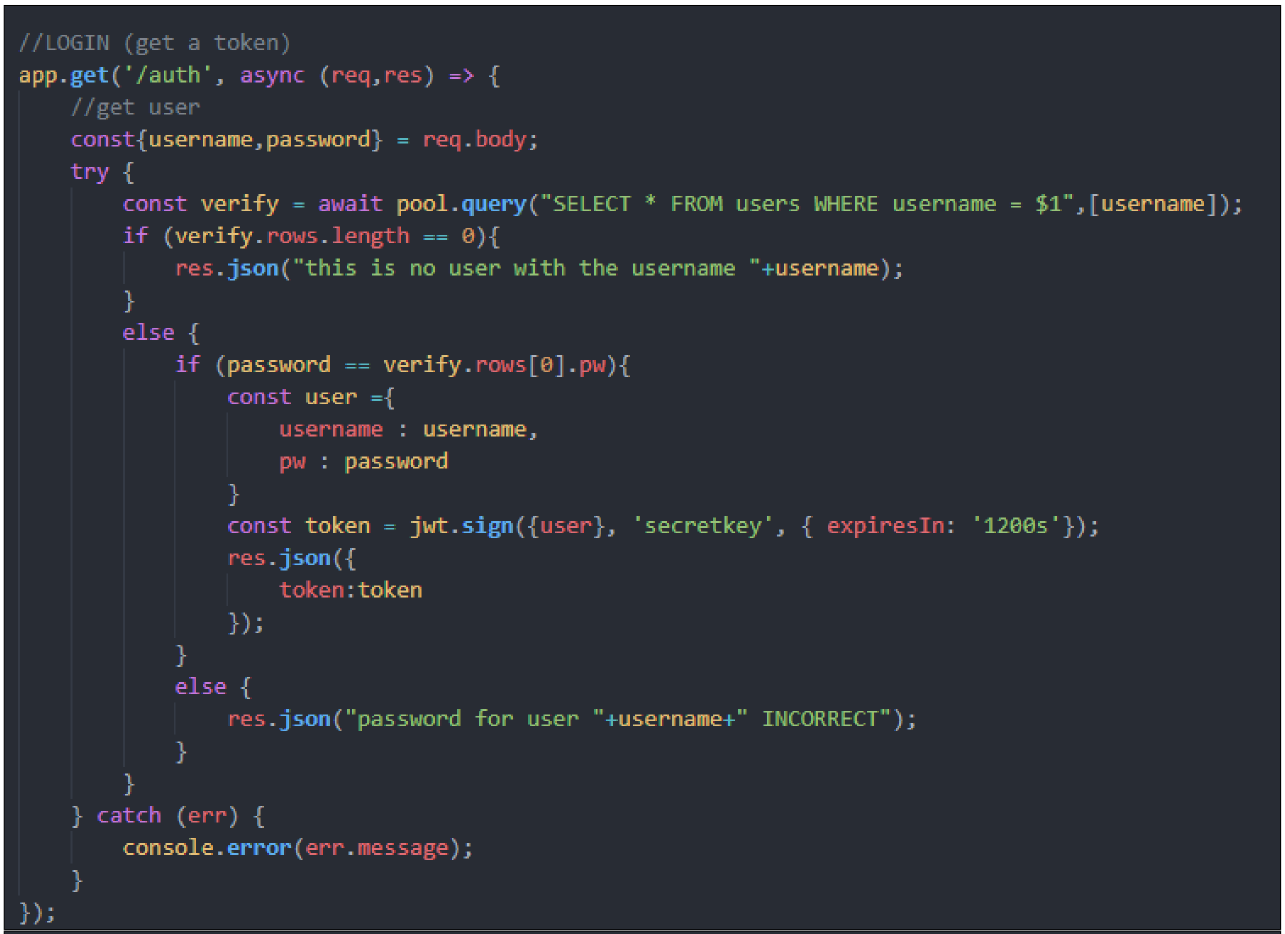
(Due to the length of this function, this is only the first and most important part. Full code is attached in the folder)



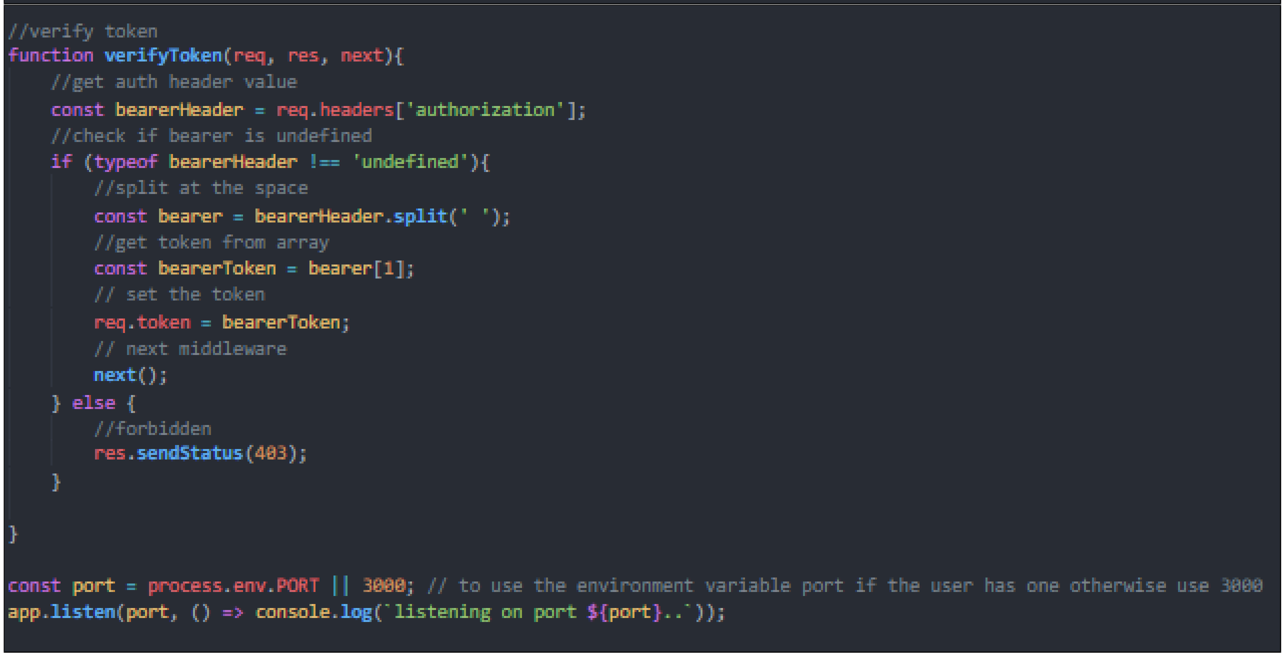
```
//MAIN FUNCTION OF THE API

app.get('/busses/:departure_station/:destination_station',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
            const{departure_station} = req.params;
            const{destination_station} = req.params;
            try {
                const departure = await pool.query("SELECT n_de_ligne,n_de_station FROM bus_lignes WHERE nom_station = $1 "
                [departure_station]);
                const arrival = await pool.query("SELECT n_de_ligne,n_de_station FROM bus_lignes WHERE nom_station = $1 ",
                [destination_station]);
                if (departure.rows.length == 0 || arrival.rows.length == 0) {
                    res.json("there is no bus station in your current location.");
                } else {
                    if ( (departure.rows[0].n_de_ligne == arrival.rows[0].n_de_ligne) && (departure.rows[0].n_de_station <
                        arrival.rows[0].n_de_station) )  {
                        res.json("You should take the bus number "+departure.rows[0].n_de_ligne+" in the station number "+
                        departure.rows[0].n_de_station+": "
                        +departure_station+". The bus number "+arrival.rows[0].n_de_ligne+" stops in the station number "+
                        arrival.rows[0].n_de_station+": "+
                        destination_station+". Have a safe ride!");
                }
```

GET /auth

Generate a JWT authentication token.



```
//LOGIN (get a token)
app.get('/auth', async (req,res) => {
    //get user
    const{username,password} = req.body;
    try {
        const verify = await pool.query("SELECT * FROM users WHERE username = $1",[username]);
        if (verify.rows.length == 0){
            res.json("this is no user with the username "+username);
        }
        else {
            if (password == verify.rows[0].pw){
                const user ={
                    username : username,
                    pw : password
                }
                const token = jwt.sign({user}, 'secretkey', { expiresIn: '1200s'});
                res.json({
                    token:token
                });
            }
            else {
                res.json("password for user "+username+" INCORRECT");
            }
        }
    } catch (err) {
        console.error(err.message);
    }
});
```

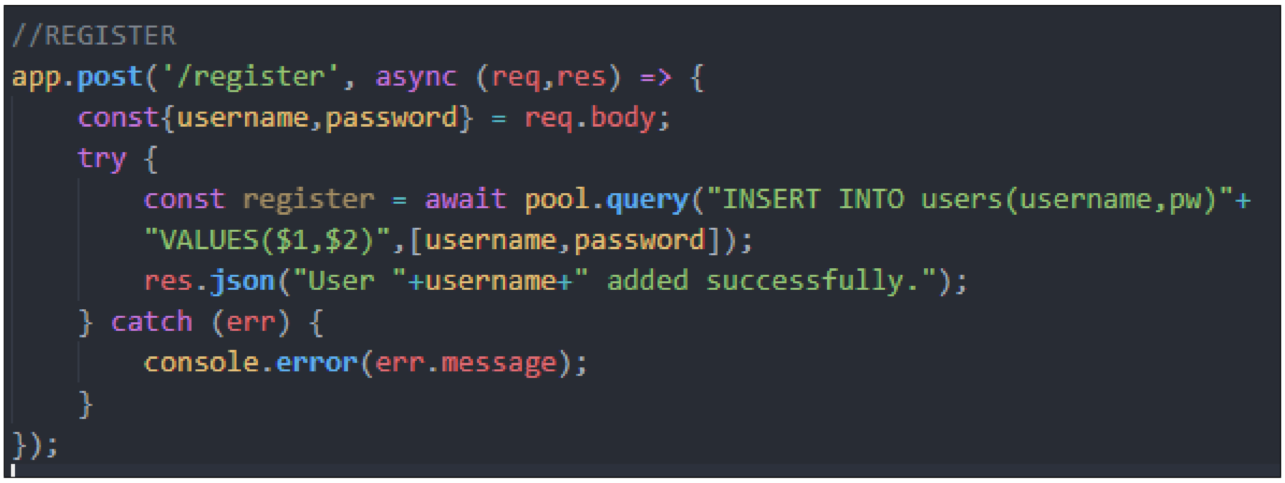


```
//verify token
function verifyToken(req, res, next){
    //get auth header value
    const bearerHeader = req.headers['authorization'];
    //check if bearer is undefined
    if (typeof bearerHeader !== 'undefined'){
        //split at the space
        const bearer = bearerHeader.split(' ');
        //get token from array
        const bearerToken = bearer[1];
        // set the token
        req.token = bearerToken;
        // next middleware
        next();
    } else {
        //forbidden
        res.sendStatus(403);
    }

}

const port = process.env.PORT || 3000; // to use the environment variable port if the user has one otherwise use 3000
app.listen(port, () => console.log(`listening on port ${port}..`));
```

2.4.2 POST Requests

POST /register

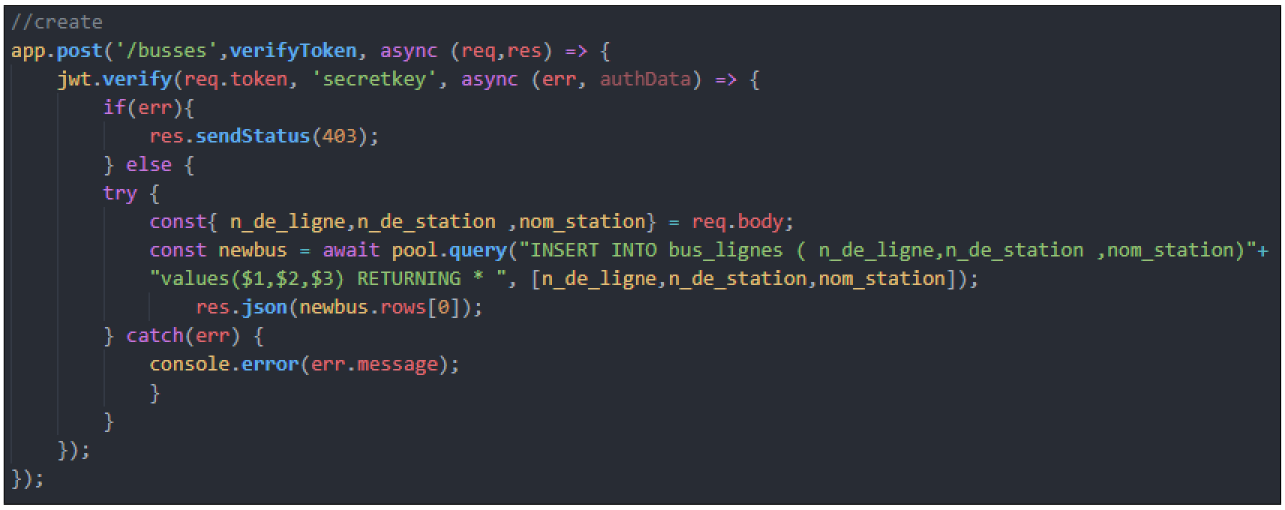
Register in the API after specifying a valid username and password in the body of the request.



```
//REGISTER
app.post('/register', async (req,res) => {
    const{username,password} = req.body;
    try {
        const register = await pool.query("INSERT INTO users(username,pw)"+
        "VALUES($1,$2)",[username,password]);
        res.json("User "+username+" added successfully.");
    } catch (err) {
        console.error(err.message);
    }
});
```

POST /busses

Add a new public bus station by specifiying the station name, number of the station,and line number in the body of the request.

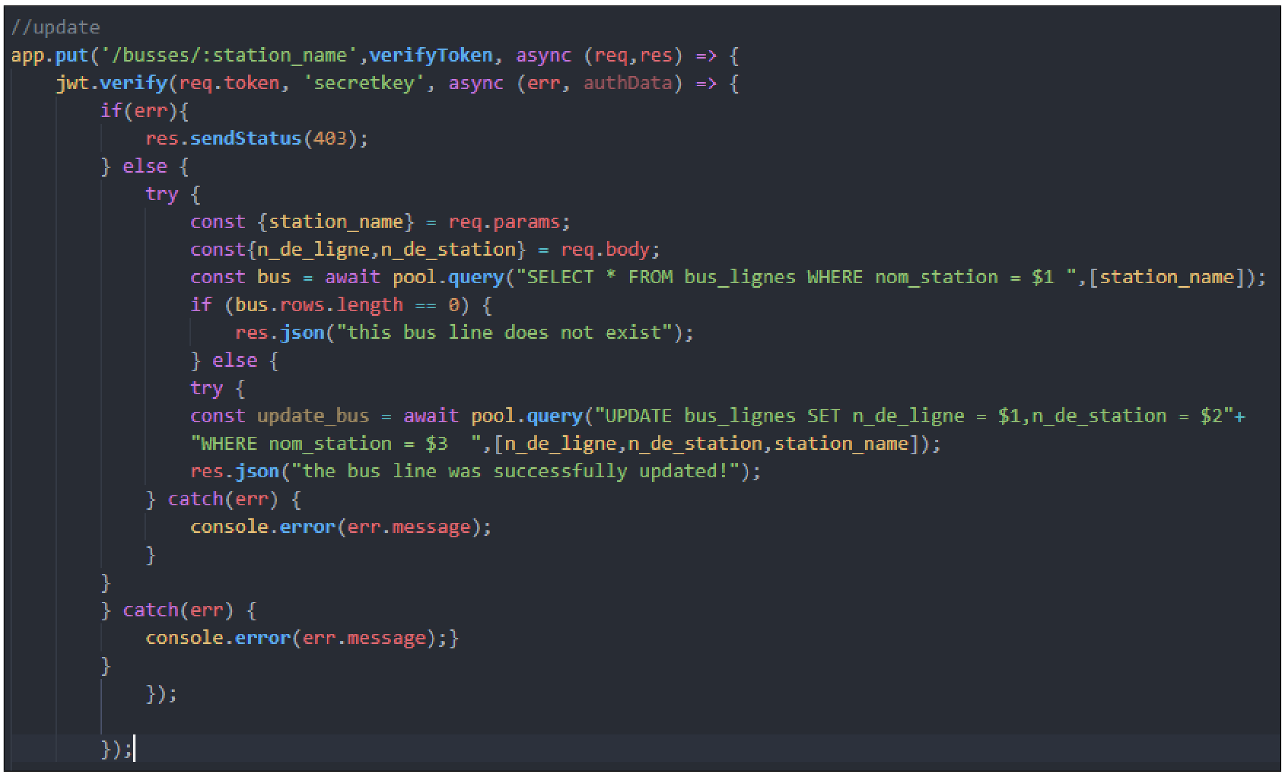


```
//create
app.post('/busses',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
        try {
            const{ n_de_ligne,n_de_station ,nom_station} = req.body;
            const newbus = await pool.query("INSERT INTO bus_lignes ( n_de_ligne,n_de_station ,nom_station)"+
            "values($1,$2,$3) RETURNING * ", [n_de_ligne,n_de_station,nom_station]);
                res.json(newbus.rows[0]);
        } catch(err) {
            console.error(err.message);
            }
        }
    });
});
```

2.4.3 PUT Requests

PUT /busses/StationName

Update the number of the station or line number of a specific Tunisian public bus station after specifying the updated value in the body of the request and the station name in the request URI.

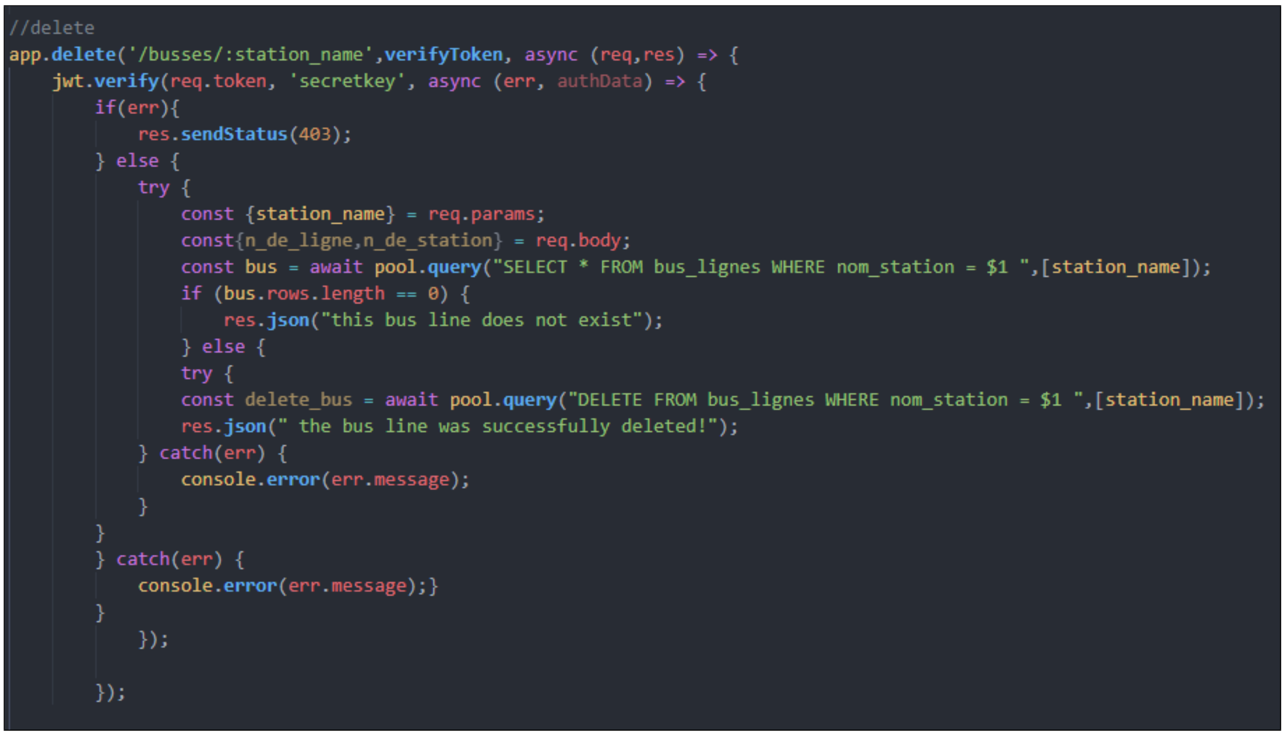


```
//update
app.put('/busses/:station_name',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
            try {
                const {station_name} = req.params;
                const{n_de_ligne,n_de_station} = req.body;
                const bus = await pool.query("SELECT * FROM bus_lignes WHERE nom_station = $1 ",[station_name]);
                if (bus.rows.length == 0) {
                    res.json("this bus line does not exist");
                } else {
                try {
                const update_bus = await pool.query("UPDATE bus_lignes SET n_de_ligne = $1,n_de_station = $2"+
                "WHERE nom_station = $3  ",[n_de_ligne,n_de_station,station_name]);
                res.json("the bus line was successfully updated!");
            } catch(err) {
                console.error(err.message);
            }
        }
        } catch(err) {
            console.error(err.message);}
        }
            });

        });
```

2.4.4 DELETE Requests

DELETE /busses/StationName

Delete a specific Tunisian public bus station from the database after specifying its name in the body of the request and the station name in the request URI.

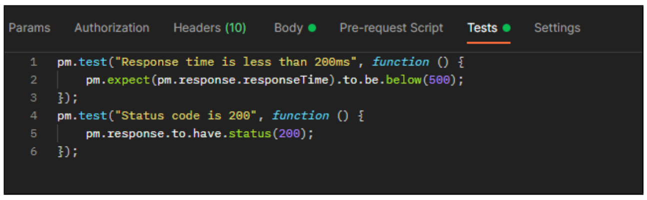


```
//delete
app.delete('/busses/:station_name',verifyToken, async (req,res) => {
    jwt.verify(req.token, 'secretkey', async (err, authData) => {
        if(err){
            res.sendStatus(403);
        } else {
            try {
                const {station_name} = req.params;
                const{n_de_ligne,n_de_station} = req.body;
                const bus = await pool.query("SELECT * FROM bus_lignes WHERE nom_station = $1 ",[station_name]);
                if (bus.rows.length == 0) {
                    res.json("this bus line does not exist");
                } else {
                try {
                const delete_bus = await pool.query("DELETE FROM bus_lignes WHERE nom_station = $1 ",[station_name]);
                res.json(" the bus line was successfully deleted!");
            } catch(err) {
                console.error(err.message);
            }
        }
        } catch(err) {
            console.error(err.message);}
        }
            });

        });
```

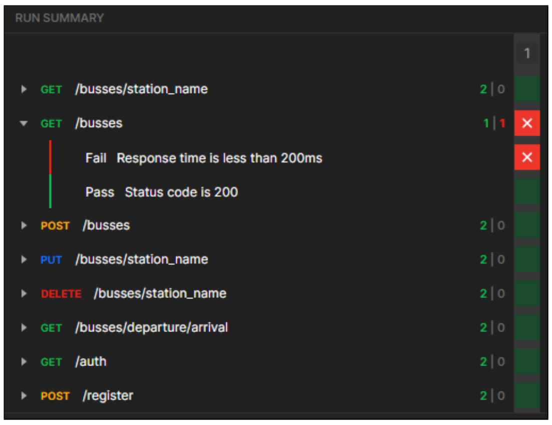
2.5 Postman Contribution

Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated. [8]

I have used Postman for the automatic testing of my API requests, as well as some snippets such as "Response Time less than 200ms","Status Code is 200"..etc.



```
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

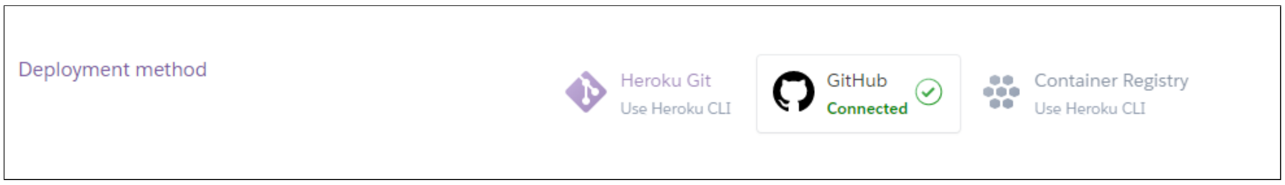


```
RUN SUMMARY
                                                        1
GET     /busses/station_name                      2 | 0
GET     /busses                                   1 | 1  ✕
            Fail   Response time is less than 200ms      ✕
            Pass   Status code is 200
POST    /busses                                   2 | 0
PUT     /busses/station_name                      2 | 0
DELETE  /busses/station_name                      2 | 0
GET     /busses/departure/arrival                 2 | 0
GET     /auth                                     2 | 0
POST    /register                                 2 | 0
```

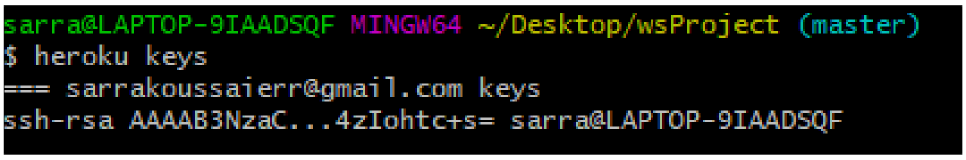
2.6 Heroku Contribution

Heroku is a cloud platform as a service supporting several programming languages. [9]

I hosted my app in the cloud using Heroku under the name "webservproject" (https://webservproject.herokuapp.com/) after adding the Procfile that specifies the commands that are executed by the app on startup including my app's web server (web: node app.js).
I have also tried integrating Heroku Postgres which is a managed SQL database service provided directly by Heroku. I have also deployed my app and connected it to my GitHub repository "WebServProject" under the master branch. In terms of security measures, I have generated



SSH Keys and added them to my heroku account. I have also tried adding an SSL Certificate using CloudFlare but found out that Heroku automatically generates one for you, which is why, my app is now completely secure with both SSL certificate and SSH keys configured.

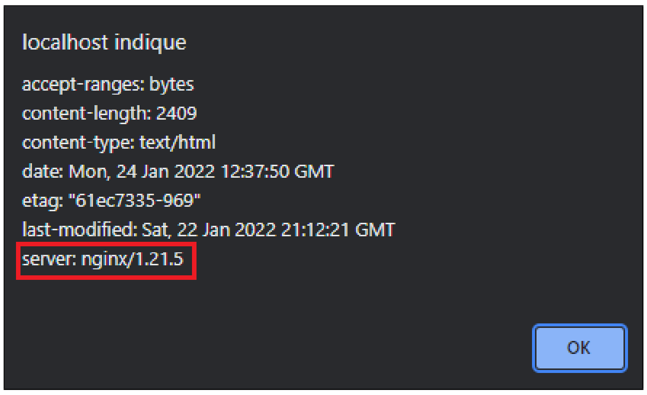


```
sarra@LAPTOP-9IAADSQF MINGW64 ~/Desktop/wsProject (master)
$ heroku keys
=== sarrakoussaierr@gmail.com keys
ssh-rsa AAAAB3NzaC...4zIohtc+s= sarra@LAPTOP-9IAADSQF
```

2.7 NGINX Contribution

NginX is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. [10]

I have used NginX for web serving since it excels at serving static content quickly. I did so by downloading it and then modifying its nginx.conf file after activating it.

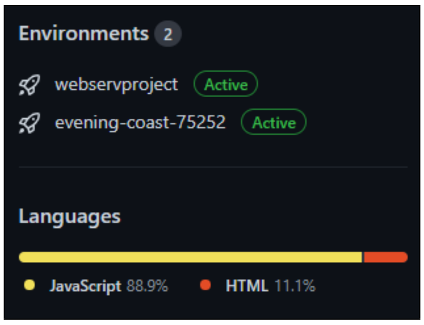


```
localhost indique
accept-ranges: bytes
content-length: 2409
content-type: text/html
date: Mon, 24 Jan 2022 12:37:50 GMT
etag: "61ec7335-969"
last-modified: Sat, 22 Jan 2022 21:12:21 GMT
server: nginx/1.21.5
```

2.8 Git Contribution

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. [11]

I have used Git-Version Control to save changes and coordinate between the different repositories that I had (local, remote, and on heroku).



Environments 2

webservproject Active

evening-coast-75252 Active

Languages

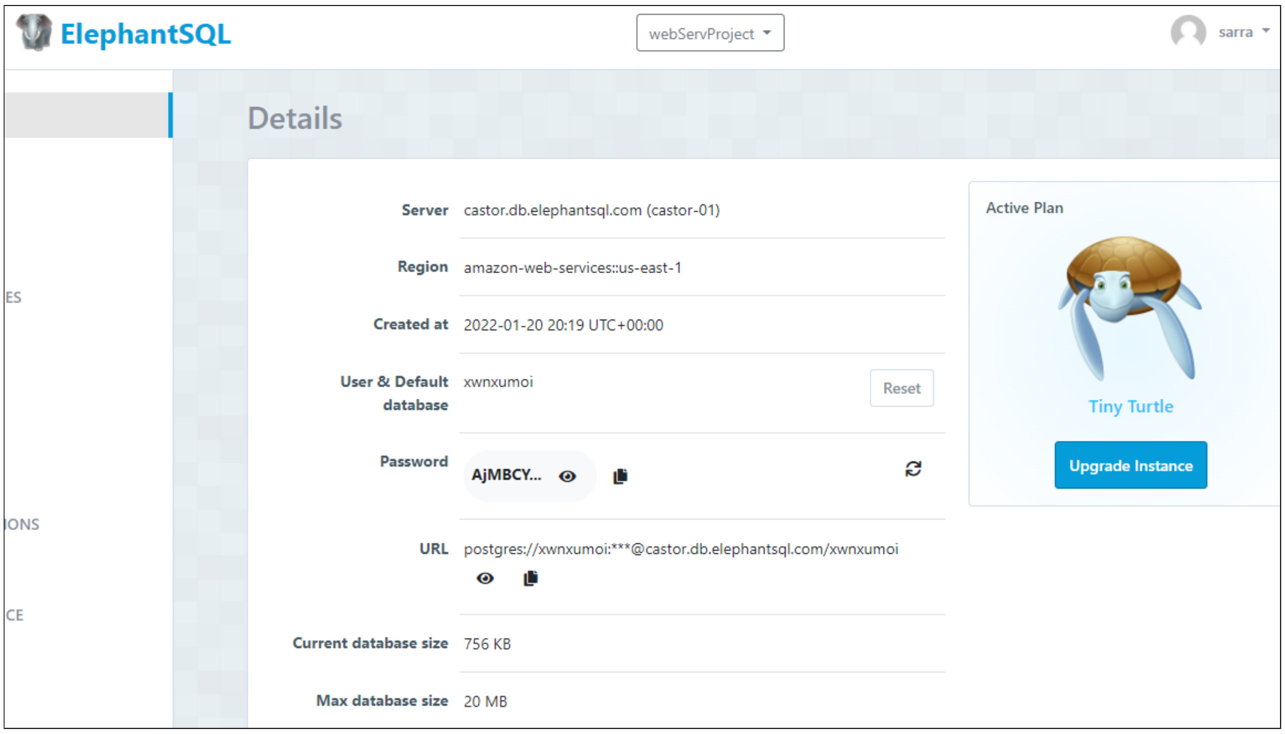
● JavaScript 88.9% ● HTML 11.1%

2.9 ElephantSQL Contribution

ElephantSQL is a free PostgreSQL database hosting service. It's a cluster of managed PostgreSQL servers in cloud-hosted databases that let you store data in a structured way that can easily be accessed, managed, and updated. It, however, does not replace Postico/pgAdmin and does not display your tables, columns, or any of the data inside of your database. [12]

After hosting my app in the cloud using Heroku, I noticed that the database wasn't accessible anymore. After doing some research and thinking, I realized that that was because my database was locally stored and therefore, Heroku has no way of reaching it.

Which is why, ElephantSQL has been one of the greatest discoveries that I made during this project, that enabled me to host my database in the cloud quite easily.

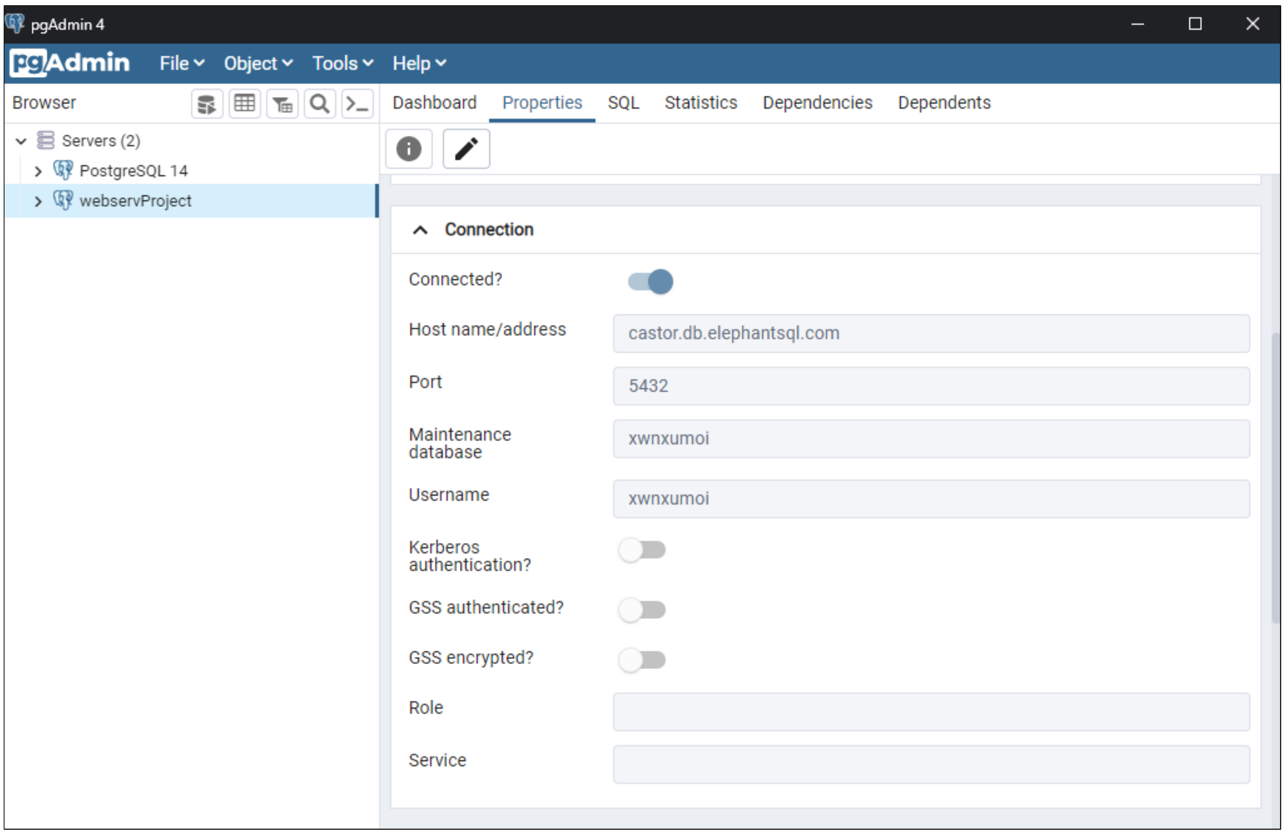


Server	castor.db.elephantsql.com (castor-01)
Region	amazon-web-services::us-east-1
Created at	2022-01-20 20:19 UTC+00:00
User & Default database	xwnxumoi
Password	AjMBCY...
URL	postgres://xwnxumoi:***@castor.db.elephantsql.com/xwnxumoi
Current database size	756 KB
Max database size	20 MB

2.10 PostgreSQL Contribution

PostgreSQL is a free and open-source relational database management system emphasizing extensibility and SQL compliance.

After hosting my database in the cloud using ElephantSQL, I used PostgreSQL to easily view and manage my data sets locally.



Connected?	✓
Host name/address	castor.db.elephantsql.com
Port	5432
Maintenance database	xwnxumoi
Username	xwnxumoi
Kerberos authentication?	
GSS authenticated?	
GSS encrypted?	
Role	
Service	

2.11 Database Structure

Source of my database : Data.Transport.Tn .

The database consists of 2 tables which no relations between them.

2.11.1 Tables

Table "bus lignes"

containing 3 columns :

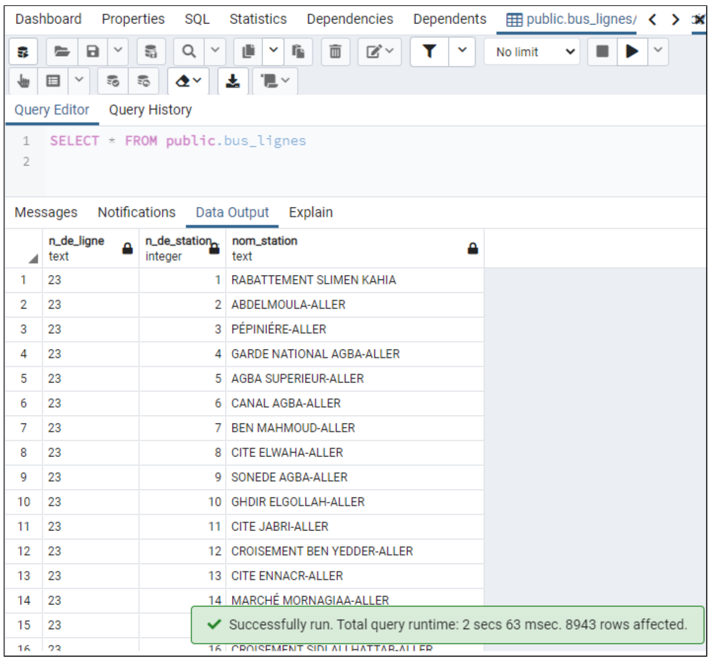
1. n de ligne : Name of the line of type text
2. n de station : Number of the station of type integer
3. nom station : name of the station of type text

Table "users"

containing 2 columns :

1. username : name of the user of type text

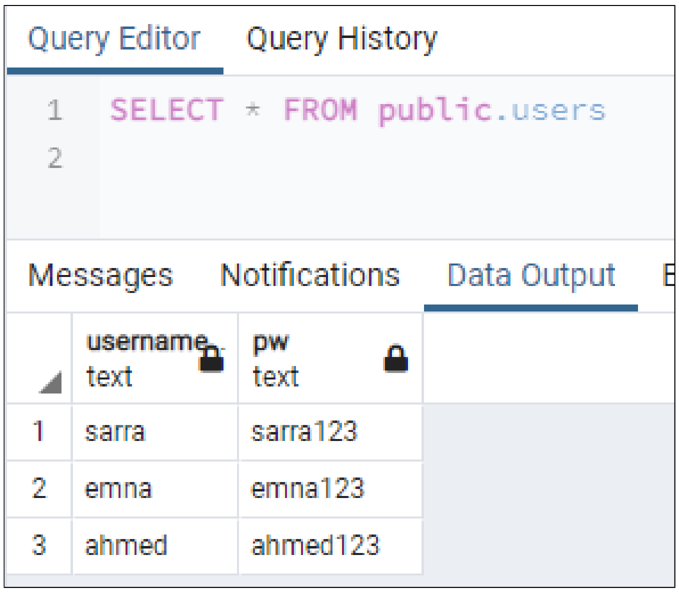
2. pw : password of the user of type text



```sql
SELECT * FROM public.bus_lignes
```

	n_de_ligne text	n_de_station integer	nom_station text
1	23	1	RABATTEMENT SLIMEN KAHIA
2	23	2	ABDELMOULA-ALLER
3	23	3	PÉPINIÉRE-ALLER
4	23	4	GARDE NATIONAL AGBA-ALLER
5	23	5	AGBA SUPERIEUR-ALLER
6	23	6	CANAL AGBA-ALLER
7	23	7	BEN MAHMOUD-ALLER
8	23	8	CITE ELWAHA-ALLER
9	23	9	SONEDE AGBA-ALLER
10	23	10	GHDIR ELGOLLAH-ALLER
11	23	11	CITE JABRI-ALLER
12	23	12	CROISEMENT BEN YEDDER-ALLER
13	23	13	CITE ENNACR-ALLER
14	23	14	MARCHÉ MORNAGIAA-ALLER
15	23		
16	23	16	CROISEMENT SIDI ALI HATTAB-ALLER

✓ Successfully run. Total query runtime: 2 secs 63 msec. 8943 rows affected.



```sql
SELECT * FROM public.users
```

	username text	pw text
1	sarra	sarra123
2	emna	emna123
3	ahmed	ahmed123

Chapter 3

Conclusion

The purpose of this project was to build an API with not only a practical purpose, but also a social and economical one. I am extremely passionate about bettering our country's future and facilitating our lives through efficient technologies and I hope that that was conveyed throughout my work.

Tunisia still has a long way to go to reach complete digitisation, but that shouldn't discourage us to contribute to its betterment, instead, it should only motivate us to do more.

This project has been a pleasure to work on. Although I faced more problems than I ever anticipated, I am glad that I had the chance to learn as much as I did and grow both academically and personally.

I am grateful to our professor Dr.Montassar Ben Messaoud who paved our learning path to lead us here, and I would like to personally thank him for his contribution and guidance.

Sarra Koussaier

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Appendix A

Response examples

A.1 GET Requests

A.1.1 GET /busses

```
[
    {
        "n_de_ligne": "23",
        "n_de_station": 1,
        "nom_station": "RABATTEMENT SLIMEN KAHIA"
    },
    {
        "n_de_ligne": "23",
        "n_de_station": 2,
        "nom_station": "ABDELMOULA-ALLER"
    },
    {
        "n_de_ligne": "23",
        "n_de_station": 3,
        "nom_station": "P PINI RE -ALLER"
    },

        ...
]
```

A.1.2 GET /busses/StationName

```
{
    "n_de_ligne": "23",
    "n_de_station": 15,
    "nom_station": "GUITOUNI-ALLER"
}
```

A.1.3 GET /busses/departureStation/arrivalStation

```
"You should take the bus number 32A in the station number 1:
 BAB ALIOUA. The bus number 32A stops in the station number 15:
  CITE ESSALAMA. Have a safe ride!"
```

A.1.4 GET /auth

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
    J1c2VyIjp7InVzZXJuYW1lIjoic2FycmEiLCJwdyI6InNhcnJ
    hMTIzIn0sImlhdCI6MTY0Mjk1NDgxNiwiZXhwIjoxNjQyOTU2M
    DE2fQ.IRYit-pRnuXxxQtDIv8W0Ul7QuuXaPOa8pwZxXx-u18"
}
```

A.2 POST Requests

A.2.1 POST /register

```
"User sarra added successfully."
```

A.2.2 POST /busses

```
{
    "n_de_ligne": "100",
    "n_de_station": 50,
    "nom_station": "TESTING"
}
```

A.3 PUT Requests

A.3.1 PUT /busses/StationName

```
"the bus line was successfully updated!"
```

A.4 DELETE Requests

A.4.1 DELETE /busses/StationName

```
" the bus line was successfully deleted!"
```

Bibliography

[1] A. B. Youssef, “Digital transformation in tunisia: Under which conditions could the digital economy benefit everyone?,” in *ERF Working Papers Series*, pp. 1–42, Nov 2021.

[2] N. Schraepel, “Digital innovations create jobs and support transparent and effective public administration,” *GIZ deutsche gesellschaft für internationale zusammenarbeit*, p. 1, 2020.

[3] “Tunisia internal travel and local transport options.” `https://www.visatunisia.com/getting-around-tunisia/#:~:text=The%20biggest%20downside%20is%20that,contain%20one%20class%20of%20carss`. [Online; accessed 23-Jan-2022].

[4] “Node.js.” `https://en.wikipedia.org/wiki/Node.js`. [Online; accessed 23-Jan-2022].

[5] “Express.” `https://en.wikipedia.org/wiki/Express.js`. [Online; accessed 23-Jan-2022].

[6] “Jwt.” `https://en.wikipedia.org/wiki/JSON_Web_Token`. [Online; accessed 23-Jan-2022].

[7] GeeksForGeeks, “Use ejs as template engine in node.js,” *GeeksForGeeks*, p. 1, Jul 2021.

[8] G. Romero, “What is postman api test,” *encora*, p. 1, June 2021.

[9] “Heroku.” `https://en.wikipedia.org/wiki/Heroku`. [Online; accessed 23-Jan-2022].

[10] “Nginx.” `https://en.wikipedia.org/wiki/Nginx`. [Online; accessed 23-Jan-2022].

[11] “Git.” `https://git-scm.com/`. [Online; accessed 23-Jan-2022].

[12] Jackie, “How to setup a database with elephantsql,” *Medium*, p. 1, Mar 2020.