Installation Manual

Students:

Salvador Orozco Villalever - A07104218

Juan Andrés Reynoso Mazoy - A01328249

Ricardo Rodiles Legaspi - A01325081

Jesús Alejandro Coles Moguel - A01171039

Raúl Barranco Cortés - A01090517

Date: October 18th, 2018

Subject: Business Solution Development Capstone Project

Professor: Dr. Juan Manuel González Calleros

INSTALLATION MANUAL
TABLE OF CONTENTS

		Page #
1.0	GENERAL INFORMATION	3
1.1	System overview	3
1.2	System requirements	3
2.0	INSTALLATION STEPS	4
2.1	Obtain source	4
2.2	Install database server	4
2.3	Install Docker CE (Community Edition)	5
2.4	Add configuration files	5
2.5	Create Docker images and run containers	7
2.6	Test correct installation	7
2.0	TROUBLESHOOTING	8

GENERAL INFORMATION

1.1 System overview

CorpusWeb is a web application and thus needs to run on a server. It can be managed and installed using Docker containers, and since the code is open source, it can also be forked and modified.

1.2 System requirements

In order to keep the installation procedure as simple as possible, it is highly recommended to use a Debian-based operating system, such as Ubuntu, to run the web app.

Additionally, a minimum of 2GB of memory is required for a system with few users, and should be increased as demanded by load.

As the system requires to run PostgreSQL, a server with at least 2GB and 64Bit architecture is required. The database may be hosted in the same server as the web application itself, or on a different server.

2.0 INSTALLATION STEPS

2.1 Obtain source

The source files may be obtained by either cloning the Github repository or by another method (e.g. flash drive, CD).

The source files are organized into three distinct directories:

- **`docs`** holds the system's documentation.
- **`frontend`** holds the Angular part of the application, which renders the HTML, CSS and JavaScript for the web interface.
- **`backend`** holds the REST API written in Node.js, which is responsible for handling, storing, and retrieving the data.

2.2 Install database server

The installation of the database server, as mentioned, can be installed either in the same server as the web app, or on a separate server. Please make sure that the port 5432 (default PostgreSQL port) is open.

To install PostgreSQL, please follow the next guides for the corresponding operating system:

- Ubuntu: https://www.postgresql.org/download/linux/ubuntu/
- Debian: https://www.postgresql.org/download/linux/debian/

Then, a non-admin user must be created for the database, and granted permission to create databases. This can be achieved using the following:

```
sudo -u postgres createuser <user name>

sudo -u postgres psql

alter user <user name> with encrypted password '<new password>';

alter user <user name> createdb;
```

To allow the user to connect, modify the following file:
/etc/postgresql/10/main/pg_hba.conf

Scroll down, and edit the line that reads:

```
local   all     postgres                peer
```

and change 'peer' to 'md5' like so:

```
local   all     postgres                md5
```

Add a line at the bottom of the file with the following format:

```
host  all <user name>  <public ip of server with web app>/32  md5
```

Next, modify the following file: /etc/postgresql/10/main/postgresql.conf

Under the section "CONNECTIONS AND AUTHENTICATION", remove the '#' (comment character) in front of `listen_addresses = 'localhost'`, and change `localhost` to `*` like so:

```
listen_addresses = '*'
```

Restart the PostgreSQL service to save the changes like so:

```
sudo service postgresql restart
```

2.3 Install Docker CE (Community Edition)

On the server where the web app will be hosted, install Docker like so:

- Ubuntu: https://docs.docker.com/install/linux/docker-ce/ubuntu/
- Debian: https://docs.docker.com/install/linux/docker-ce/debian/

Once done, make sure to run the demo app included in the instructions to verify that it runs correctly.

2.4 Add configuration files

There are two configuration files that need to be modified with the correct information for the server.

First, create a file in the path backend/server/config/config.js and add the following:

```
{

 "development": {

   "username": "<PostgreSQL user name>",
```

```
    "password": "<PostgreSQL user password>",
    "database": "database_development",
    "host": "<public ip address of server where PostgreSQL was installed>",
    "port": "5432",
    "dialect": "postgres"
  },
  "test": {
    "username": "<PostgreSQL user name>",
    "password": "<PostgreSQL user password>",
    "database": "database_test",
    "host": "<public ip address of server where PostgreSQL was installed>",
    "port": "5432",
    "dialect": "postgres"
  },
  "production": {
    "username": "<PostgreSQL user name>",
    "password": <PostgreSQL user password>,
    "database": "database_production",
    "host": "<public ip address of server where PostgreSQL was installed>",
    "dialect": "mysql"
  }
}
```

Next, modify the file that is named frontend/src/environments/environment.prod.ts

Change

```
baseUrl: 'http://localhost:8000/'
```

To

```
baseUrl: 'http://<public ip of the server where the web app will run>:8000/'
```

2.5 Create Docker images and run containers

Now we will create the Docker images using the Dockerfiles included in both the frontend and the backend.

On the command line, navigate to the backend directory and run:

```
sudo docker build -t corpus-web_backend .
```

Then in the frontend directory, run:

```
sudo docker build -t corpus-web_frontend .
```

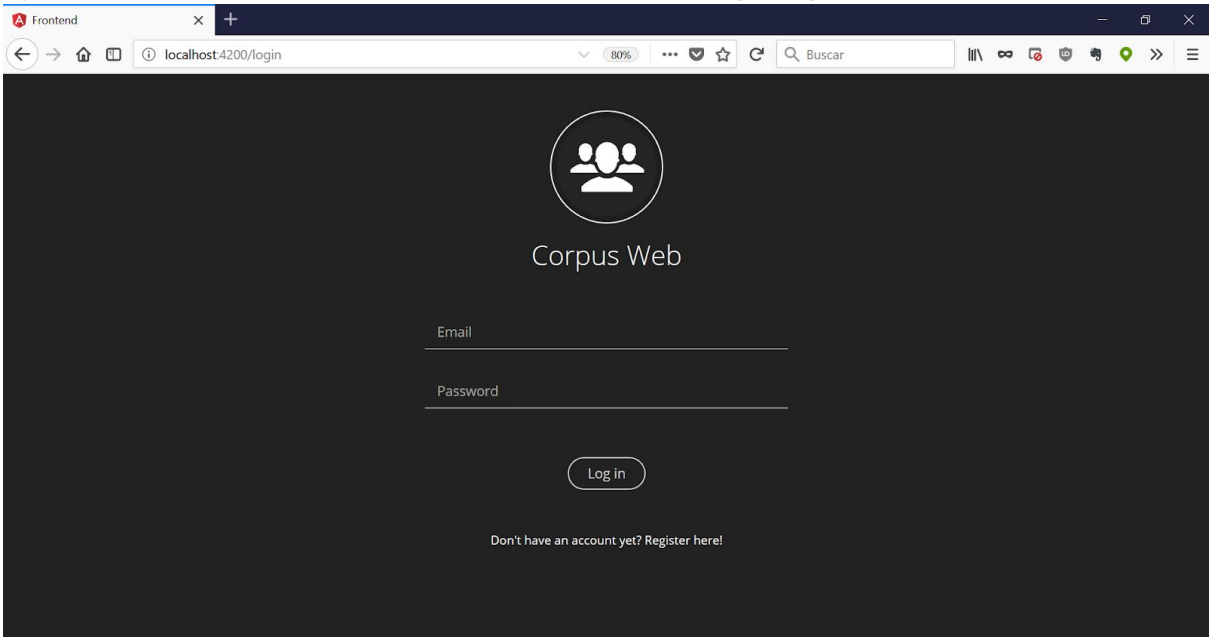
The Docker images are now created and containers for each can be started with:

```
sudo docker run --name corpus-web_backend -p 8000:8000 -d corpus-web_backend
```

```
sudo docker run --name corpus-web_frontend -p \
    <port of host where the website should be accessible>:8000 \
    -d corpus-web_backend
```

2.6 Test correct installation

In order to test a correct installation, navigate to the website by opening a web browser and navigating to the server's IP or hostname, followed by the port (if different than 80). The following page should appear:



Registration should work properly. If so, then the system is correctly installed.

3.0 TROUBLESHOOTING

- "The page is not showing"

Something is likely wrong with the frontend container. Please check the output of the following command:

```
docker logs corpus-web_frontend
```

- "I am not able to register a user"

Something is likely wrong with the backend container. Please check the output of the following command:

```
docker logs corpus-web_backend
```

- "The backend crashes whenever I try to do any actions on the frontend"

The database may not be configured properly. To fix this, enter the backend container by running:

```
docker exec -it corpus-web_backend /bin/bash
```

And then, inside the container:

```
npm install -g sequelize-cli
sequelize db:drop
sequelize db:create
sequelize db:migrate
sequelize db:seed:all
```