Installation Guide
Project ‘FindMeAPet’

Content

Illustrations

1. Introduction

1.1. Purpose of the Document

The intend of this document is to shine a light on the process of building the proper setup for working, maintaining, and deploying this software product.

The whole system is divided into 3 main components: web front-end, mobile front-end and a backend to link business requirements and perform them separately when needed.

This separation is important due to how the organization manages their internal processes and how they want certain requirements to be met.

They want a separate interface and platform for the user experience related to people not related to the organization, this would correspond to the web front-end. The mobile front-end would correspond to the internal processes and users of the organization.

Finally, a backend is necessary to properly respond to the requirements performed by these components together or separately.

In this guide you'll find instructions and suggestions on the required and recommended software to install, the hardware and computer resources necessary to sustain the needs of the system, the versioning systems used, deployment instructions, the database configuration and management, the workflow suggested and other related topics (see Table of Contents).

2. System Requirements

The system has diverse needs that must be met for it to run properly. First, there are the software requirements that the system requires either to function or to be worked on.

There're also the hardware requirements which would need to be met to have a proper experience when working or running the project.

2.1. Software Requirements

- Git +2.31.0v
- Node.js +16.1.0v
- Ionic +5.0v
- Prettier 2.2.1v (as a VSCode plugin is recommended)
- Sonar +8.9.0v
- Express +4.16.4v
- (Recommended) VSCode +1.58.2v
- (Recommended) Google Chrome +92.0v*

> Note:
> The developers might use the web browser their choice, however, a chromium based is highly r ecommended to avoid any potential compatibility issues.

2.2. Hardware Requirements

- 4Gb of RAM
- I3 6th or newer generation Intel Processor (or equivalent from another brand)
- At least 2 GB storage space

2.3. Operating Systems

If the operating system is relatively modern (has 64bit support) and other requirements are met, the OS can be the one preferred by the engineer, however, bear in mind that not all requirements or dependencies can be met by operating systems. Considering the 3 families of operating systems we recommend:

- Windows 10
- MacOS +10.15.0v (Catalina or older)
- Ubuntu +20.04v (Focal Fossa or older)

2.4. Database.

- MySQL +8.0.24v

> Note:
> It possible to use SQL versions as old as 5.0v, however, there'll need to be adjustments to the encodings o f the entities since support for the new standard ("CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci") it' s not possible; therefore, we strongly discourage using older versions of SQL.

3. Pre-Installation

3.1. Credentials to auth GitHub

To get access to the remote repository, you can go to https://github.com and authenticate using the following credentials:

Email: findmepet2021@gmail.com
Password: $oftware2021

3.2. Download from repository

To get a copy of the projects in your local machine, you should clone it from its remote repositories on GitHub. Each project (backend, Front-end web, and Front-end mobile) has its own repository, so you can clone just the needed ones.

Open the shell in the folder you want to clone the project and write the following command to clone the application repository.

Backend:

```
> git clone https://github.com/GPA-Espol/FindMeAPet_Backend.git
```

Front-end web:

```
> git clone https://github.com/GPA-Espol/FindMeAPet_Web.git
```

Front-end mobile:

```
> git clone https://github.com/GPA-Espol/FindMeAPet_Mobile.git
```

3.3. Git branch management (Backend, Front web, Front mobile).

For each project, a different branch strategy was adopted:

3.3.1. Backend

The backend branching was based in six important concepts:

- For each new user story, create a new branch.
- For naming the branch, follow the convention:
  FMP- [User Story ID]- [User Story name] (e.g., FMP-15-Formulario-Adopcion).
- Merge user story branches into the “dev” branch using pull requests.
- Don’t commit directly to the “main” branch.
- Only merge the “dev” branch into the “main” branch when all user stories of a new release are integrated into the “dev” branch.
- Keep a high quality, up to date “main” branch.

3.3.2. Front-end web:

The Front-end web branching was based in six important concepts:

- For each new user story, create a new branch.
- Name each branch as its user story name (e.g., FiltroAdoptar).
- Merge user story branches into the “development” branch using the git merge command.
- Don’t commit directly to the “master” branch.
- Only merge the “development” branch into the “master” branch when all user stories of a new release are integrated into the “development” branch.
- Keep a high quality, up to date “master” branch.

3.3.3. Front-end mobile:

The Front-end mobile branching was based in six important concepts:

- For each new developer, create a new branch.
- Name each branch as the developer owner.
- Merge developer branches into the “develop " branch using the git merge command without fast-forward strategy.
- Don’t commit directly to the “master” branch.
- Only merge the “develop” branch into the “master” branch when all features of a new release are integrated into the “develop” branch.
- Keep a high quality, up to date “master” branch.

4. Installation the system

4.1. Backend

4.1.1. Environment

In the backend there are 3 environments:

- Development
- Test
- Production

It is worth mentioning that these three environments share the variables of the credentials of the connection to the database, which are:

- Username
- Password
- Database
- Host

Additionally, all environments share the variables of the port and the secret json web token for the authentication.

PORT: Number port for the server

JWT_SECRET_KEY_FMAP: json web token for authentication

```
$env:PORT="3500"
$env:JWT_SECRET_KEY_FMAP="$$Gp4_2021"
```

Illustration 1 Shared variables in all environments

Manually it is only necessary to enter the credentials of the development environment

4.1.1.1. Development environment

In this environment, for the connection to the database, there needs to be a previously created connection, so make sure that MySQL is installed, and its service is up.

We go inside the project to the 'src' folder and look for the dbconfig.js file

```
development: {
  username: 'root',
  password: 'root',
  database: 'database_dev',
  host: '127.0.0.1',
  dialect: 'mysql',
},
```

Illustration 2 Example of credential values for the development´s database connection

As we can see, there are the variables mentioned above, where:

- username: is the username of MySQL Server
- password: is the password of the MySQL Server user
- database: is the name of the database *
- host: is the name of the host from which the user connects to the MySQL Server.

- dialect: ‘mysql’ this value is by default and is not modified

> Note:
> The name of the database is at your discretion

In the ‘src’ folder there is a file called database.js where the following values are placed inside the Sequelize instance which creates the connection with the database.

```
const sequelize = new Sequelize(
  config.development.database,
  config.development.username,
  config.development.password,
  {
    host: config.development.host,
    dialect: config.development.dialect,
  }
);
```

Illustration 3 Sequelize configuration in the development environment

In addition, as the last requirement, it is necessary to install Nodemon globally

```
> npm install -g nodemon
```

or you can also install Nodemon as a development dependency:

```
> npm install --save-dev nodemon
```

4.1.2. **Test environment**

A .env file must be created which will have the credentials of the database for the production environment.

This file must contain the following variables:

CI_DB_USERNAME: user of the database

CI_DB_PASSWORD: password of the database

CI_DB_NAME: database name

```
$env:CI_DB_USERNAME="rootr"
$env:CI_DB_PASSWORD="root"
$env:CI_DB_NAME="database_test"
```

Illustration 3 Example of credential values for the test´s database connection

In the src folder, there is a file called database.js where the following values are placed inside the Sequelize instance.

```
const sequelize = new Sequelize(
  config.test.database,
  config.test.username,
  config.test.password,
  {
    host: config.test.host,
    dialect: config.test.dialect,
  }
);
```

Illustration 4 Sequelize configuration in the test environment

4.1.2.1. Production environment

A .env file must be created which will have the credentials of the database for the production environment.

This file with variables:

PROD_DB_USERNAME: user of the database

PROD_DB_PASSWORD: password of the database

PROD_DB_NAME: database name

PROD_DB_HOSTNAME: is the name of the host from which the user connects to the MySQL Server.

```
$env:PROD_DB_USERNAME="gpa_user"
$env:PROD_DB_PASSWORD="gpa_user"
$env:PROD_DB_NAME="proyecto_gpa"
$env:PROD_DB_HOSTNAME="te-learning.com"
```

Illustration 5 Example of credential values for production´s database connection

This .env file should be saved in the .gitignore file

In the src folder, there is a file called database.js where the following values are placed inside the Sequelize instance.

```
const sequelize = new Sequelize(
  config.production.database,
  config.production.username,
  config.production.password,
  {
    host: config.production.host,
    dialect: config.production.dialect,
  }
);
```

Illustration 6 Sequelize configuration in the production environment

4.1.3. **Installation**

As with other components, once the project has been cloned into a local repository, the respective dependency must be installed. Thanks to having Node.js installed (see Software Requirements) we can make use of Node's packet manager (npm) to do this automatically.

Go to the project's root folder (backend-find-me-pet), you can verify you're at the root of the project if you find a package.json file in the current directory. Once inside the root of the project, you need only to install project dependencies.

```
> npm install
```

This will effectively install all dependencies specified in package.json and package-lock.json automatically (and locally, so no need to worry about it installing it system-wide).

4.1.4. **Run**

4.1.4.1. **Development environment**

Go to the root folder of the project and run the following command:

```
> npm run dev
```

4.1.4.2. **Test environment**

Go to the root folder of the project and run the following command:

```
> npm run test
```

4.1.4.3. **Production environment**

Go to the root folder of the project and run the following command:

```
> npm start
```

4.2. Front end web

4.2.1. **Installation**

As with other components, once the project has been cloned into a local repository, the respective dependency must be installed. Thanks to having Node.js installed (see Software Requirements) we can make use of Node's packet manager (npm) to do this automatically.

Go to the projects root folder (gpa-web-front), you can verify you're in the root of the project if you find a package.json file in the current directory. Once inside the root of the project you need only to execute:

```
> npm install
```

This will effectively install all dependencies specified in package.json and package-lock.json automatically (and locally, so no need to worry about it installing it system-wide).

4.2.2. **Run**

Go to the root folder of the project and run the following command:

```
> npm start
```

4.2.3. **Test**

Go to the root folder of the project and run the following command:

```
> npm run test
```

This will run JEST's native script in and the so that tests start running.

4.3. Front end Mobile

4.3.1. **Installation**

Once the project is cloned, the dependencies must be installed to run the project. This can be done by opening the shell in the root folder of the project and executing the following command:

```
> npm install
```

or

```
> npm i
```

This will retrieve all the project's dependencies and put them into the "node_modules" folder.

4.3.2. **Run**

Once the dependencies are installed, the project can be executed with the following command:

```
> ionic serve [options]
```

To run the application in development environment run the command:

```
> ionic serve
```

If you want to run the application in production environment, run the command:

```
> ionic serve --prod
```

The project will be hosted in localhost at the port 8100. So, you can open your browser and go to http://localhost:8100.

To see all the options available for ionic serve visit: ionic serve: Start a local dev server for app dev/testing

4.3.3. Test

For this project Jasmine was used to describe the test cases, and karma to set the test environment and run the tests.

For running the tests, you need to run the following command:

```
> npm test
```

This will run all the test cases located on the project files with the extension. spec.ts. The test report will be generated at the directory /coverage/ngv/lcov-report.

To open the tests report you can run the command:

```
> npm run cov-report
```

5. Deployment

5.1. Backend

For the deployment. One needs only to create a build of the system with a build script that comes native to React.
To do this we must go to the root directory of the project and run the following command:

```
> npm run start
```

After this the build must be merged using the version management system (Git) to the main branch. To do this, you need to use the following commands:

```
> git checkout master
> git merge <branch>
> git checkout master
```

The project is hosted in Heroku. Build changes that are performed on the main branch of the project will go live and can be interacted with by any end-user that go to the hosting service URL.

5.2. Front end web

For the deployment. One needs only to create a build of the system with a build script that comes native to React.

To do this we must go to the root directory of the project and run the following command:

```
> npm run build
```

After this the build must be merged using the version management system (Git) to the main branch. To do this, you need to use the following commands:

```
> git checkout master

> git merge <branch>

> git checkout master
```

The project is hosted in Heroku. Build changes that are performed on the main branch of the project will go live and can be interacted with by any end-user that go to the hosting service URL.

5.3. Front end Mobile

5.3.1. Fulfill prerequisites for Script Based Deployment

To fulfill the prerequisites for script-based deployment of the mobile application, you must have installed:

- Android Studio (version 40. or newer).
- Gradle (version 6.4 or newer).
- Java JDK (version 8).

Also, you must configure the following environment variables:

1. Set **ANDROID_SDK_ROOT** to the directory where you have installed the Android SDK (e.g., C:\Users\User\AppData\Local\Android\Sdk).
2. Set **GRADLE_HOME** to the directory where you have installed Gradle. (e.g., C:\Users\user\Downloads\gradle-7.1).
3. Set **JAVA_HOME** to the directory where you have installed the Java8 JDK (e.g., C:\Program Files\Java\jdk1.8.0_201).
4. Append **%JAVA_HOME%/bin** to the PATH environment variable.
5. Append **%GRADLE_HOME%/bin** to the PATH environment variable.

5.3.2. Run deployment script.

To deploy the apk, first you must add android as a target production platform run the following command:

```
> ionic Cordova platform add android
```

Then, to generate the apk you must run the following command.

> Note:
> You don't have to add android as a target production platform every time you deploy the system, just the first time.

```
> npm run release
```

This command will:

- Run our tests and generate the coverage report.
- Open the coverage report in the browser.
- Generate the documentation report of the system.
- Build the android installer (apk) of the software in the production environment.

Then Navigate to the directory: /platform/android/app/build/outputs/apk/debug to find the apk file named “GPA.apk”.

6. Appendices

6.1. Glossary of Terms