PLACEMENT OPPORTUNITIES WITH ENTREPRENEURIAL REACH

Co-funded by the Erasmus+ Programme of the European Union

Manual for deploying POWER project components

The first part of this document contains basic information about what IT skills and effort is required by partners in the POWER project. The skills required and an estimate of effort to complete the IT work is divided into parts based on the IT components required.

Further technical details should be discussed by the IT personnel appointed by the partners to complete the tasks and the developers working with the EUF.

1. Central platform

Short description: A central platform containing the tools required for Placement Opportunity management.
Contains the user interface for Company representatives to add Placement Opportunities and for Institution staff to review those opportunities.
Has an API, that exposes the Placement opportunities to the middleware (point 2.)

Provided by: EUF

Documentation: User's guide will be provided for Institution partners.

IT work required from partners:

- No IT work is required from the partners

2. Lightweight middleware

Short description: The middleware is used to hide the credentials used by the React application (point 3.) when fetching information from the POWER platform's API (point 1.). Has no other goal.

Technologies used:

- *NodeJS* and *NPM*

  or

- *PHP* and *composer*

Provided by: Middleware code is provided by EUF

Other required tasks: Hosting, deployment and monitoring of the middleware is done by the partners

Documentation: A step-by-step guide will be provided by EUF with instructions on deploying the middleware application to a university server

IT work required by partners:

- Host middleware on a university server (doesn't need a new server, can be hosted together with other university web applications)
  - Required environment: *PHP* or *NodeJs*
- Ensure connectability of the middleware to an API hosted on this platform (detailed in point 1.) for the React application to fetch data from
- Potentially includes: Setting firewall rules, letting requests and responses through

Estimated time and field of work: 2 days for an IT administrator

3. React application

Short description: An application embeddable into the university websites. Displays a list of public Placement Opportunities and institution-specific Placement Opportunities on the webpage it is used on.

Technologies used: *React*, *NPM*

Provided by: React application code is provided by EUF

Other required tasks: Hosting, deployment and monitoring of the React application is done by the partners

Documentation: A step-by-step guide on deploying and embedding the application will be provided by EUF.

IT work required from partners:

- Clone the code from github
- Run *npm run build:widget*, in order to compile the app into a *js* and *css* files,
- Copy those files into the university server, which are then used as a library (imported with *script* and *link* tags in the *head* tag)

Estimated time and field of work: 2 days for an IT administrator or web administrator / developer, preferably with some basic web developer skills.

Optionally: The React app is customizable for the Institution's preferences if they have the capacity and motivation to do so.

Summary of IT Skills required

- Basic server administration skills
- Web server configuration skills
- Basic usage of git
- Basic knowledge of NPM
- Optionally basic knowledge of composer
- Optionally basic knowledge of docker

Summary of Estimated IT Effort

- IT administration work: 2-3 days
- Website administration / deployment work: 2-3 days

Deployment guides by system component

This chapter contains a guide on how to deploy the software components that are present in the POWER project architecture.

Each of these software components have a github repository for easy access and seamless deployment by Institution staff. The repositories contain the same deployment information. This is general practice in software development.
These repositories are not public, but anyone who wants to use the POWER platform, should and will be provided access.

1. **Central platform**

   This piece of software does not have to be deployed by the partners, it is maintained and managed by the project administrators.

2. **Lightweight middleware**

   There are two middleware solutions in the POWER project. These two are functionally equivalent, but they use a different stack of development tools for the convenience of the project partners and other stakeholders.

   SlimPHP middleware solution deployment steps

   Github repository: https://github.com/EuropeanUniversityFoundation/power-middleware-slim

   System requirements

   This POWER middleware solution is based on Slim PHP. According to the official documentation, system requirements are:

- Web server with URL rewriting (Apache or Nginx recommended)
- PHP 7.4 or newer

Note: You do not have to perform the steps detailed in the official installation guide. In order to leverage this repository:

- Git must be installed on the server.
- Composer 2 must be present on your server. See the official composer documentation on how to install it.

Deployment steps

1. Create a domain / subdomain for the middleware on the hosting server.
2. Create a site in your webserver (Apache or Nginx) on the hosting server, that has PHP 7.4+ and URL rewriting installed and is accessible by the website that contains the POWER React app.
3. Enter the root directory of the site and clone the repository there: `git clone git@github.com:EuropeanUniversityFoundation/power-middleware-slim.git`
4. Run `composer install` to install the dependencies of this repository.
5. Set the document root (the home directory served by the site) to the `./public` directory.
6. Duplicate the `power_settings.example.php` file with the name `power_settings.php` and edit its content.
7. Insert the API key you received from us, between the quotes:

```
<?php
```

```
$power_settings = [
  'api_key' => '[Enter your API key here]',
  'base_url' => 'https://power.uni-foundation.eu',
];
```

If you don't have an API key, check the 'How to start using the POWER platform' on the POWER platform site and follow the instructions there.

Testing

To test if the deployment was successful, send a GET request to `http(s)://[domain you installed to]/power-middleware/rest/public-pos`, where you should see placement opportunity data in JSON format.

Further steps

Once you're done with this deployment, you can start to deploy and setting up the POWER React Application

NodeJS middleware deployment steps

Deployment

This app can be used as an alternative of the SlimPHP middleware. To create the embeddable app start by deploying the code:

```
git clone https://github.com/EuropeanUniversityFoundation/power-middleware.git {PROJECT_ROOT}
```

If PROJECT_ROOT is omitted the directory will be named power-middleware.
Then from the command line:

```
cd {PROJECT_ROOT}
npm install              # Install all the dependencies
```

The next step is to change the *Document root* and the *Application root* for the domain. The *Document root* is the location where the static assets can be found. Usually, the *Document root* is a directory inside the *Application root* directory (e.g. *public*). Moreover, the *Application root* is where the Application startup file is located (e.g. *app.js* or *index.js*).

For example:

- *Application root*: {PROJECT_ROOT}
- *Document root*: {PROJECT_ROOT}/public

Troubleshooting

Sometimes the steps above are not enough. Therefore, you would need to enable *Phusion Passenger* for *nginx*.

On the *nginx* settings of your server add the code written below:

```
passenger_enabled on;
```

Stack

Built with Node.js and Express.

Quick start with Docker

In order to test this profile with Docker, you need `docker`, `docker-compose` and `make` installed on your system. If your system meets the requirements, follow these steps:

```
git clone git@github.com:EuropeanUniversityFoundation/power-middleware.git
cd euf-base
cp .env.example .env          # The .env file is ignored by version control
nano .env                     # Edit the environment variables if necessary
make up                       # Create and start the Docker containers
make shell node               # Access a shell in the Node container
```

In alternative, if `node` is already installed in your machine, it's possible to run the project without starting a `docker` container.

Once the `node` container is accessible, it's possible to run the following commands:

npm start

Runs the app in the development mode.
Open http://localhost:5000 to view it in your browser.

3. POWER React Application

Quick start with Docker

In order to test this profile with Docker, you need `docker`, `docker-compose` and `make` installed on your system. If your system meets the requirements, follow these steps:

```
git clone git@github.com:EuropeanUniversityFoundation/power_react_app.git
cd euf-base
```

```
cp .env.example .env          # The .env file is ignored by version control
nano .env                     # Edit the environment variables if necessary
make up                       # Create and start the Docker containers
make shell react              # Access a shell in the Node container
```

In alternative, if `node` is already installed in your machine, it is possible to run the project without starting a `docker` container.

Once the `node` container is accessible, you can run the following commands:

npm start

Runs the app in the development mode.

Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.

You may also see any lint errors in the console.

npm test

Launches the test runner in the interactive watch mode.

npm run build

Builds the app for production to the `build` folder.

It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.

npm run build:widget

Builds the app for production to the `widget` folder.

It creates one `js` file and one `css` file, which can be used as libraries to be embedded in different websites.

It needs `npm install` and `npm run build` to be able to work.

Adding Custom Environment Variables

Your project can consume variables declared in your environment as if they were declared locally in your JS files. By default you will have `NODE_ENV` defined for you, and any other environment variables starting with `REACT_APP_`.

Note: You must create custom environment variables beginning with `REACT_APP_`. Any other variables except NODE_ENV will be ignored to avoid accidentally exposing a private key on the machine that could have the same name. Changing any environment variables will require you to restart the development server if it is running.

Deployment

This React app is meant to be embedded in institutions' websites.

To create the embeddable app start by deploying the code:

```
git clone https://github.com/EuropeanUniversityFoundation/power_react_app.git {PROJECT_ROOT}
```

If `PROJECT_ROOT` is omitted the directory will be named `power_react_app`.

Then from the command line:

```
cd {PROJECT_ROOT}
npm install              # Install all the dependencies
npm run build              # Compile the files in /src and place them into
/build
npm run build:widget     # Create `index.js` and `index.css` files and place
them into /widget
```

The generated widget files can now be included in the website.

Example structure

For a very simple webpage structure like the one shown below:

```
.
└── webpage
    ├── css
    │   └── style.css
    ├── index.html
    └── powerapp
        ├── index.css
        └── index.js
```

The HTML document must include generated widget files and an HTML element to which the application will attach:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <link rel="stylesheet" type="text/css" href="powerapp/index.css" />
    <title>POWER App</title>
  </head>
```

```
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="power"></div>
    ...
    <script src="powerapp/index.js" type="text/javascript" />
  </body>
</html>
```