

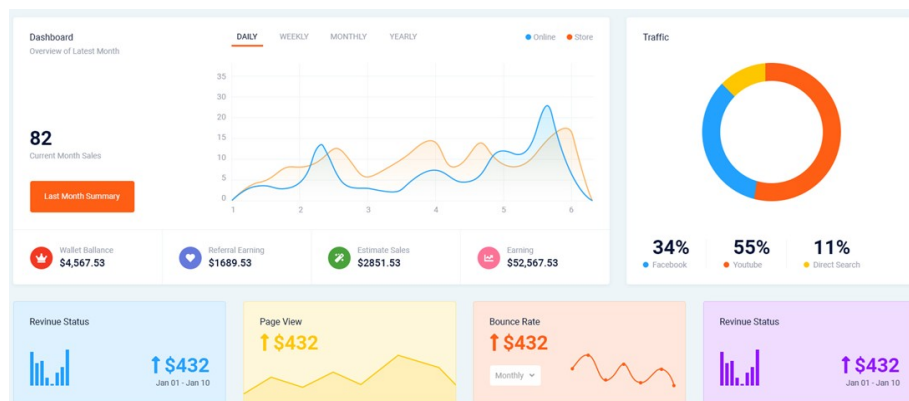


B5 - Application Development

B-DEV-500

Dashboard Bootstrap

All your information in the blink of an eye





Dashboard Bootstrap

repository name: DEV_dashboard_bootstrap_\$ACADEMICYEAR
repository rights: ramassage-tek
language: node.js, react.js
compilation: docker-compose build && docker-compose up



- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

INTRODUCTION

The Dashboard bootstrap will be divided into **2 main parts** :

- The docker composes,
- Create User in a database and call connection trough back-end and front-end.



Username

Password

Login

Create profile

The objective of this bootstrap is to give you the basic basics to start your project properly. Initially it will be a question of creating your first docker-compose including / understanding 3 services (the server, the client and the data base).

In a second step it will be a question of making a call from your front-end towards your back-end !



Although the project can be done in Node.js, .NET and Java, the bootstrap will be done in **Node.js** for the back-end and in **React.js** for the front-end.



DOCKER-COMPOSE

Before starting the **docker-compose** itself, you will need to create a **Dockerfile** specific to each of the services you want to create.

```
Terminal
~/B-DEV-500> ./ls -R
.:
client/ db/ docker-compose.yml server/

./client:
Dockerfile

./db:
Dockerfile

./server:
Dockerfile
```

+ +SERVER

As explained just before, you must create a service called “server” which will be a docker of a version of **node.js** :

- version: alpine,
- port: 5000:80,
- install dependencies with “npm”.

+ +CLIENT

Now it's about creating the service called “client” which will be a docker of a version of **react.js** :

- version: latest,
- port: 8080:8080,
- install dependencies with “npm”.

+ +DB

Also, you must create a database service called “db” which will be a docker of a version of **mariadb** :

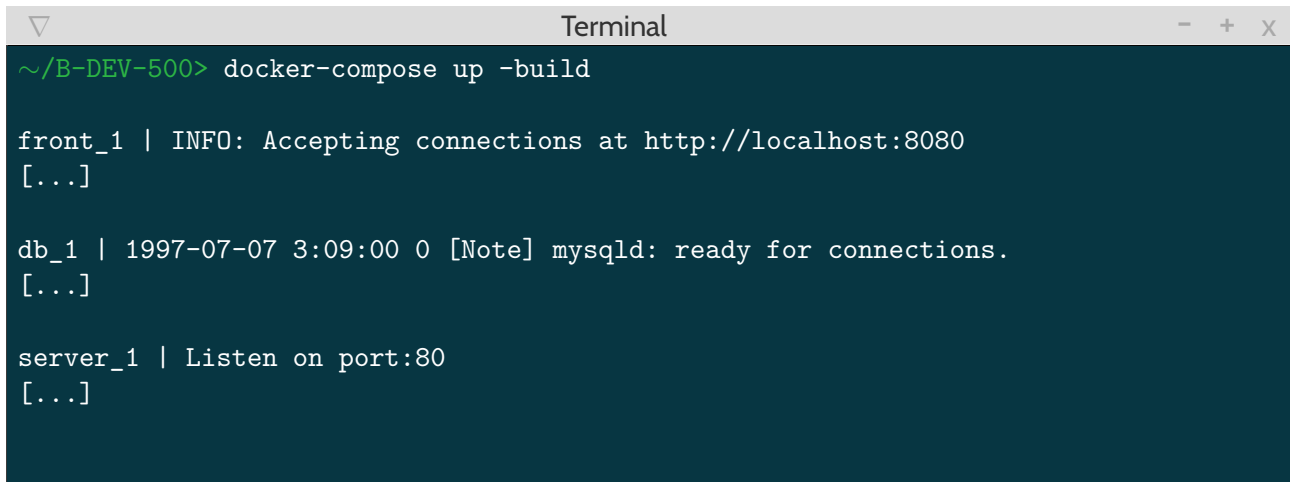
- version: latest,
- port: 3306:3306.



+ +NETWORKS

To be able to let your dockers communicate, you will have to create networks :

- back-db-tier: link “db” to “server”,
- front-back-tier: link “client” to “server”.



```
~/B-DEV-500> docker-compose up -build

front_1 | INFO: Accepting connections at http://localhost:8080
[...]

db_1 | 1997-07-07 3:09:00 0 [Note] mysqld: ready for connections.
[...]

server_1 | Listen on port:80
[...]
```



VERY SIMPLE AUTHENTICATION

Now that your 3 **services** are working, you will have to link them together. To do this, you will create an authentication system.

You'll do it in 3 parts :

- Creating Database
- Connecting Back-end to Database
- Connecting Front-end to Back-end

+ +CREATE USER IN DATABASE

To start, create a user table on your **database**. These tables must be created automatically when the docker image is created. This table will contain a unique identifier, a **user name** and a **password**. You will initially be able to store passwords in clear, but you must think quickly to find a way to store them securely.

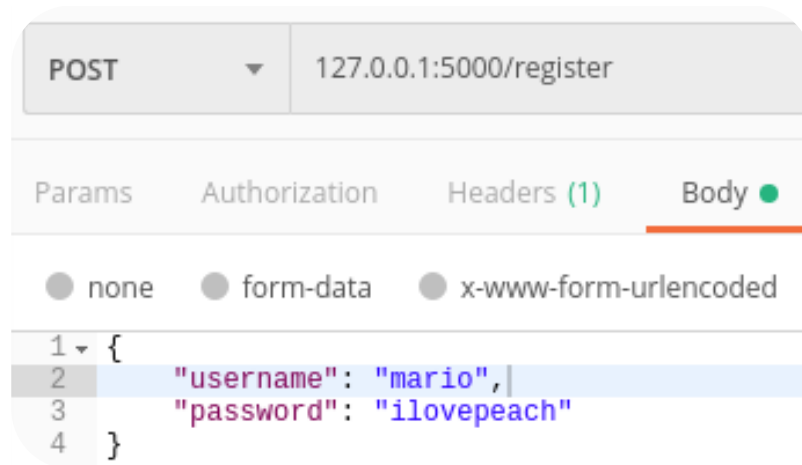
```
Terminal
~/B-DEV-500> cat schema.sql

CREATE TABLE users (
id INTEGER NOT NULL AUTO_INCREMENT,
username VARCHAR(255) NOT NULL,
password VARCHAR(255) NOT NULL,
CONSTRAINT PRIMARY KEY (id)
);
```

+ +LINK BACK-END TO DATABASE

Now that your database is ready to host your users, you need to ensure that your backend can communicate with the database. Once the connection established creates a function which creates a user as well as a function which makes it possible to retrieve the information of a user from an identifier or a username.

Once you have tested these two functions, create two routes on your API, one that allows you to create a user and one that allows you to log in (so verify that the username and password are valid). You can test your routes with tools such as Postman.



+ +LINK FRONT-END TO BACK-END

Now that everything is ready to welcome users, you can create the Register and Login forms on your Front-End to allow your future users to easily use your services! Do not worry about the style of your page for now, start by having functional forms.



Well, now that you have an operational department, it's time to start conquering the APIs for your project!

+ +NON-EXHAUSTIVE LIST OF APIS

Simple API to beginners :

- <https://www.thecocktaildb.com/api.php/>
- <https://pokeapi.co/>
- <https://data.strasbourg.eu/pages/accueil/>