

Compte rendu d'activité

Description d'avancement du projet existant à mon arrivée :	2
Description de mes tâches :	2
Tache n°1 :	2
Le besoin :	2
La solution que je veux apporter :	2
Pourquoi avoir choisi cette solution ?	3
Résolution étape par étape :	3
Problèmes rencontrés au cours de ces étapes :	5
Solutions apportées :	6
Tache n°2 :	7
Le besoin :	7
La solution réalisée :	7
Tache n°3 :	7
Résumé des réalisations :	9
Activity Report in numbers:	10

Description d'avancement du projet existant à mon arrivée :

Un front host sur Vercel (un hébergeur gratuit) assez complet pour un rendu minimal avec des besoins d'amélioration qui ont été exprimés par l'équipe dont notamment :

1. Pouvoir trier les cours et les vacataires
2. Implémenter une authentification
3. Petits changements à l'écran d'ajout et de modification des vacataires

Une API presque complète qui est elle aussi host sur une autre machine vercel qui nécessite certaines modifications pour accepter les améliorations liées à l'authentification.

Une base mongodb

Note :

Ni le front ni le back n'ont de test implémenté.

Manque de doc utilisateur.

Description de mes taches :

Tache n°1 :

Le besoin :

- Pas de moyen de tester les modifications apportées à l'api en local sans une longue manipulation à chaque lancement de la machine. Ce qui est problématique pour l'environnement de travail sur lequel on est : Plusieurs sessions d'1h30 par journée. Chaque installation pouvait durer 30 minutes et fallait changer les appels du front à l'api à chaque lancement en local. Et fallait éviter de commit ces changements au front
- Une façon facile de transmettre le logiciel final à l'iut au cas où notre solution est adoptée n'avait pas encore été mise en place.

La solution que je veux apporter :

- Un environnement de développement Docker avec lequel on peut lancer tous les services avec une commande sur n'importe quelle machine qui disposerait de docker.

Pourquoi avoir choisi cette solution ?

J'ai choisi cette solution parce que Docker est une solution très bonne pour lancer plusieurs services avec facilité. En plus, l'utilisation de docker permet que l'environnement de travail soit tout le temps le même indépendamment de la machine sur laquelle on exécute celui-ci. De plus, même si on dirait que ça apporte les mêmes avantages qu'une machine virtuelle celui-ci est beaucoup plus facile à manipuler et beaucoup plus léger.

Résolution étape par étape :

Création du Dockerfile pour le front en angular:

La création du Dockerfile n'a pas eu beaucoup de problèmes

```
# syntax=docker/dockerfile:1
FROM node:18
WORKDIR /code
ENV PATH /code/node_modules/.bin:$PATH
COPY package.json package.json
RUN npm install
RUN npm install -g @angular/cli@7.3.9
EXPOSE 4200
COPY . .
CMD npm start
```

Création du Dockerfile pour l'api

```
# syntax=docker/dockerfile:1
FROM node:18
WORKDIR /code
ENV PATH /code/node_modules/.bin:$PATH
COPY package.json package.json
RUN npm install
EXPOSE 3000
COPY . .
CMD npm start
```

Puis une fois celui-ci créé on peut passer à la création du docker compose:

```
compose.yaml
1  # Use root/example as user/password credentials
2  version: '3.1'
3
4  services:
5
6      mongo:
7          image: mongo
8          volumes:
9              - mongodb_volume:/data/db
10         environment:
11             MONGO_INITDB_ROOT_USERNAME: root
12             MONGO_INITDB_ROOT_PASSWORD: example
13             MONGO_INITDB_DATABASE: test
14
15         mongo-express:
16             image: mongo-express
17             ports:
18                 - 8081:8081
19             environment:
20                 ME_CONFIG_MONGODB_ADMINUSERNAME: root
21                 ME_CONFIG_MONGODB_ADMINPASSWORD: example
22                 ME_CONFIG_MONGODB_URL: mongodb://root:example@mongo:27017/
23
24         api:
25             build: sae5.01-gestion_vacataires-API
26             depends_on:
27                 - mongo
28             ports:
29                 - 3000:3000
30
31         angular:
32             build: sae5.01-gestion_vacataires/s5.01-app-gestion-vacataires
33             ports:
34                 - 80:4200
35
36     volumes:
37         mongodb_volume:
```

J'en ai profité pour rajouter un service nommé mongo-express qui est un outil graphique de gestion de la base de données mongodb.

Mais ce n'est pas si simple, les appels à l'api sont codés en dur, et donc même si on lance ce fichier avec docker compose up -d l'api qui serait contactée n'est pas celle qui est lancée en local

Même problème pour la base de données contactée par l'api mais celle-là peut facilement être paramétrée dans une variable d'environnement

J'ai donc apporté ces changements dans le code. Pour changer l'api du front j'ai changé les appels dans l'api et j'ai utilisé la fonctionnalité de proxy dont dispose angular. Donc tous les appels vers /api/ fais au front angular sont redirigés vers l'api paramétré par l'archive proxy-conf.json

```
1 {
2   "/api": {
3     "target": "http://api:3000",
4     "secure": false
5   }
6 }
7
```

Ceci à des avantages, si jamais on veut que l'api se trouve dans une autre adresse il faut tout simplement changer cette archive ou en utiliser une autre en changeant la commande de lancement.

Aussi, pour les connexions depuis eduroam par exemple, (un des wifis de ut2), tous les ports autres que le 80 et le 443 sont bloqués. Donc l'ancienne api sur le port 3000 n'est pas une solution viable. Ceci fait que toute communication se fasse sur le port 80

Un exemple d'archive souhaitée est le suivant

```
1 {
2   "/api": {
3     "target": "http://localhost:3000",
4     "secure": false
5   }
6 }
```

Celui-ci pointerait vers une api sur localhost

Problèmes rencontrés au cours de ces étapes :

Plusieurs, entre d'autres surtout une explication des nouvelles commandes à utiliser pour l'équipe.

Pendant que j'implémentais ces modifications, j'ai monétairement laissé inaccessible l'ancienne API host sur vercel sur laquelle ils se connectaient pour faire leurs tests. Cette indisponibilité de service à durée une matinée pendant que les initiaux avaient cours et que nous les alternants pas encore.

Mon erreur provient du fait que j'étais conscient que cette erreur allait se produire et que j'allais la corriger dès mon arrivée le lendemain matin. Sauf que je n'avais pas calculé que les initiaux avaient cours avant nôtre arrivée. Je me suis donc sincèrement excusé à mon équipe pour ces heures d'indisponibilité du service.

Je n'aurais pas dû commit avant d'expliquer le nouveau fonctionnement.


Aussi, les initiaux ne répondaient que rarement aux questions et étaient que rarement disponibles.

De plus, sous ce nouveau mode de fonctionnalité, le hot reload du front en angular n'était plus disponible.


Solutions apportées :

Annexe 1 : La liste des instructions données à mon équipe pour ce nouvel environnement de travail.

Petit tuto pour tout lancer sur la machine sur laquelle vous êtes d'un coup:

 **Stemon8** 05/10/2023 23:20
Pour les machines de l'ut si jamais elles n'ont pas docker d'installé (tuto ubuntu)
<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository> (editado)

Docker Documentation
Install Docker Engine on Ubuntu
Jumpstart your client-side server applications with Docker Engine on Ubuntu.
This guide details prerequisites and multiple methods to install.



a faire une fois par machine si ce n'est pas déjà cloné

```
git clone https://github.com/SAE-IUT/sae5.01-gestion_vacataires.git
git clone https://github.com/SAE-IUT/sae5.01-gestion_vacataires-API.git
```

puis:

```
nano compose.yaml
```

puis vous collez ce contenu:

```
# Use root/example as user/password credentials
version: '3.1'

services:
  mongo:
    image: mongo
    volumes:
      - mongodb_volume:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
      MONGO_INITDB_DATABASE: test
  mongo-express:
    image: mongo-express
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: example
      ME_CONFIG_MONGODB_URL: mongodb://root:example@mongo:27017/
  api:
    build: sae5.01-gestion_vacataires-API
    depends_on:
      - mongo
    ports:
      - 3000:3000
  angular:
    build: sae5.01-gestion_vacataires/s5.01-app-gestion-vacataires
    ports:
      - 80:4200

volumes:
  mongodb_volume:
```

vous sortez avec control + x et vous sauvegardez

Pour prendre en compte les modifs que vous avez apportées avant chaque lancement faites
puis pour lancer

```
docker compose up --build
```

ceci vous donne accès au front sur

<http://localhost/>

à l'api sur <http://localhost:3000/>

et à la base mongo db avec user `admin` et password `pass` sur :

<http://0.0.0.0:8081/> (editado)

(évidemment que les accès sont différents pour le serveur final ;)

Annexe 2 : Modifications au tutoriel pour réimplémenter le hot reload au front angular

si vous voulez le live reload pour le serveur de dev

```
docker compose up mongo mongo-express api -d --build
```

```
cd sae5.01-gestion_vacataires/s5.01-app-gestion-vacataires/
```

```
ng serve --host 0.0.0.0 --proxy-config proxy.confdev.json
```

Tache n°2 :

Le besoin :

- L'api n'a pas de test case

La solution réalisée :

J'ai donc implémenté des tests basiques sur l'api avec jest

```
describe("GET /api/vacataires", () => {  
  it("should return all vacataires", async () => {  
    const res = await request(app).get("/api/vacataires");  
    expect(res.statusCode).toBe(200);  
  });  
});  
  
describe("GET /api/modules", () => {  
  it("should return all vacataires", async () => {  
    const res = await request(app).get("/api/modules");  
    expect(res.statusCode).toBe(200);  
  });  
});
```

Pas de problèmes trouvés sur cette tâche

Tache n°3 :

Cette tâche consistait à trouver une alternative du host sur vercel qui imposait que le front, l'api et la bd soit chacune sur une machine différente qui faisait des appels très lents.

J'ai donc voulu utiliser un serveur mis à disposition par l'iut pour le mettre en place. Sauf que le serveur est sujet à un firewall très limité et ne pouvait pas encore être utilisé. En attendant je l'ai mis sur un serveur personnel.

En plus, j'ai voulu mettre en place la sécurisation https sauf que je suis tombé sur assez de contretemps qui ne m'ont pas permis de le faire à temps.

Résultat:

Un serveur qui host deux serveurs,

<http://vacataires-dev.mvallew.com/connexion>

et

<http://vacataires.mvallew.com/connexion>

Le premier traque la branche dev et le deuxième la branche main

Avec des cron job le premier se met à jour toutes les dix minutes et le deuxième deux fois par jour

Résumé des réalisations :

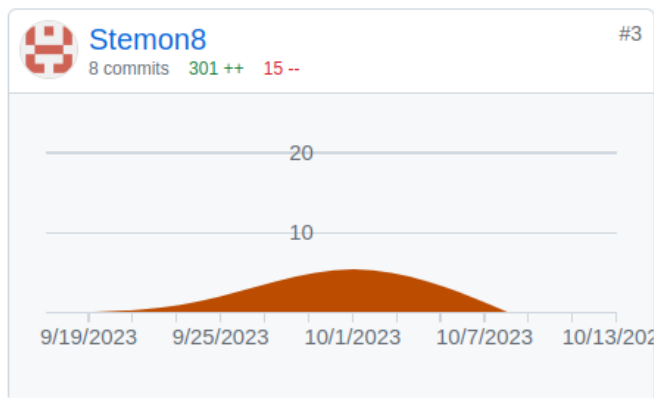
J'ai donc fait 3 tâches qui améliorent l'environnement de travail :

- Tests pour l'api
- Dockerisation du projet
- Mise en place d'un serveur temporaire sur lequel nos changements sont affichés

Activity Report in numbers:

```
marco@marco-Surface-Laptop-4:~/OneDrive/BUT/S5/sae5.01-gestion_vacataires$ cat ../log-front.log
"2023-10-06 11:56:46 +0200";"api modifications for dockerfile compat"
"2023-10-05 22:20:50 +0200";"docker compose necessary modifications + readme changes"
"2023-10-03 11:45:49 +0200";"Create docker-publish.yml"
"2023-10-03 11:43:59 +0200";"docker additions"
marco@marco-Surface-Laptop-4:~/OneDrive/BUT/S5/sae5.01-gestion_vacataires$ cat ../log-api.log
"2023-10-06 11:47:38 +0200";"api path changes"
"2023-10-05 22:03:29 +0200";"Change config to allow for local connections and add custom launch port feature"
"2023-10-05 09:24:54 +0200";"Create docker-publish.yml"
"2023-10-05 09:02:34 +0200";"Dockerfile addition + Very basic readme creation"
```

Angular front:



Rest api:

