Programmation distribuée - Rapport de projet

Tiffany Fettah Levacher - Yvonne Lam

Sommaire

1. Présentation du projet **2**
2. Création de nos services **3**
3. Création des images Docker **5**
4. Déploiement Kubernetes **7**
5. Sécurisation du Cluster **7**
6. Projets Google Labs **8**
a. Google labs - Yvonne Lam 8
b. Google labs - Tiffany Fettah Levacher 8

1. Présentation du projet

Ce projet vise à développer une application web permettant le suivi et la notation du cycle menstruel, en s'appuyant sur une architecture distribuée. L'application est composée d'un backend développé en NestJS, d'un frontend en React, et d'une base de données MongoDB, le tout conteneurisé avec Docker et orchestré par Kubernetes.

L'architecture repose sur des principes de programmation distribuée, avec une séparation claire des services et une communication sécurisée via Istio (mTLS, Virtual Service & Gateway). La gestion des accès est assurée par un système RBAC, limitant les permissions aux ressources critiques.

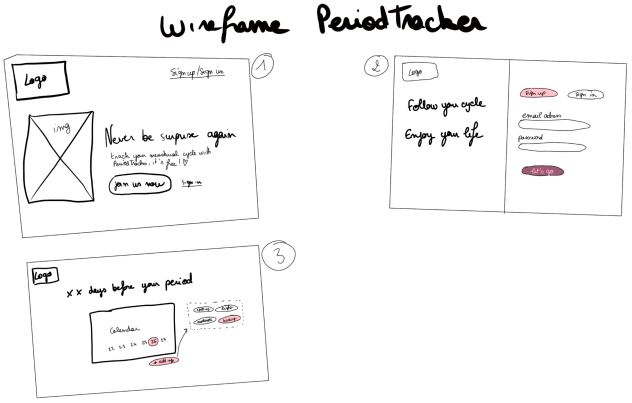
Les fonctionnalités implémentées sont les suivantes :

- Créer un compte, se connecter ;
- Avoir les informations de son cycle menstruel (GET) ;
- Rentrer des informations sur son cycle menstruel (POST) ;
- Modifier les informations sur son cycle menstruel (PUT).

2. Création de nos services

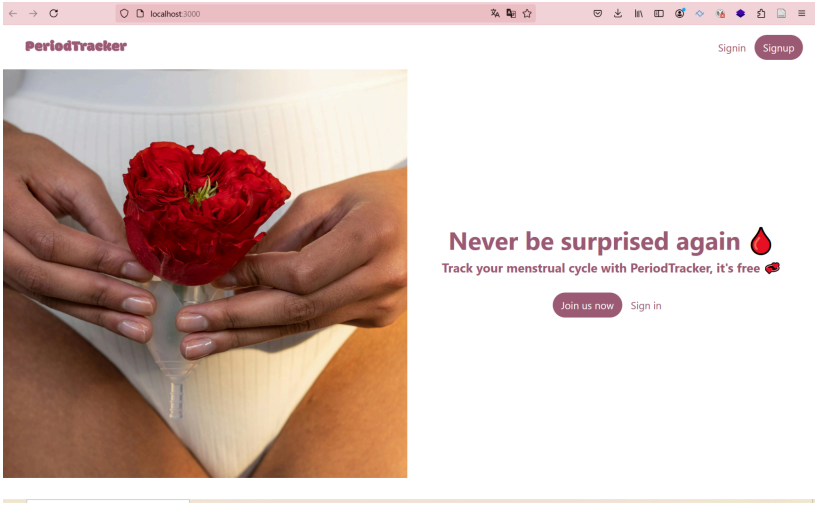
Nous avons commencé par créer nos différents services.
Tout d'abord, nous avons un backend en NestJs qui interagit avec une base de données en MongoDB. Le backend implémente toutes les fonctionnalités précédemment mentionnées, et se charge de gérer toutes les données de l'utilisateur, c'est-à-dire les informations d'inscription de l'utilisateur et les dates de son cycle menstruel. Pour la gestion de l'authentification, nous avons utilisé des token JWT qui sont envoyés au front lors de la connexion. Par ailleurs, la validation des données est faite grâce à des dto (data transfer object).

Ensuite, le service front a été créé en React. Avant de commencer, un wireframe a été déterminé pour créer l'aspect visuel de l'interface utilisateur :

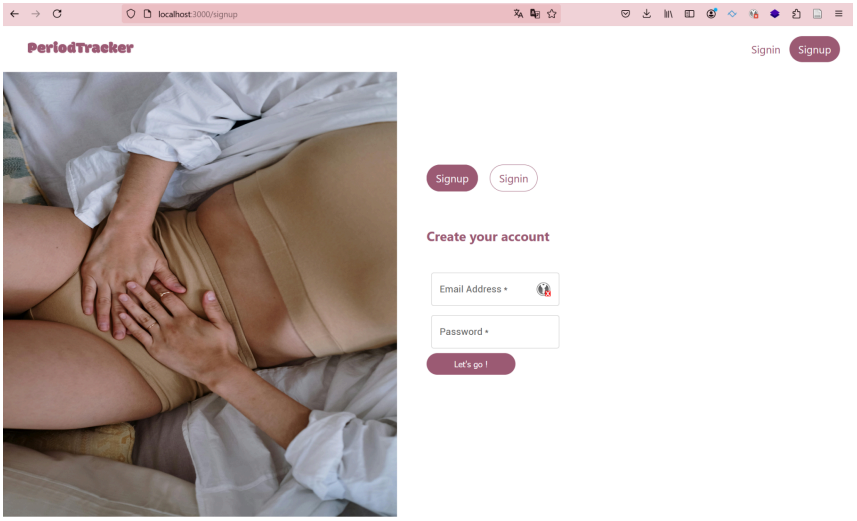


En React, ce wireframe se matérialise en 4 pages front :

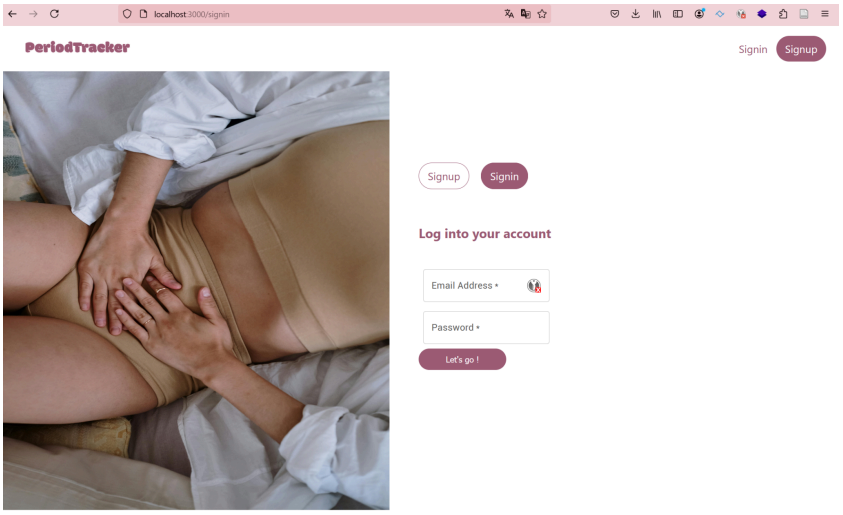
1. La page d'accueil :



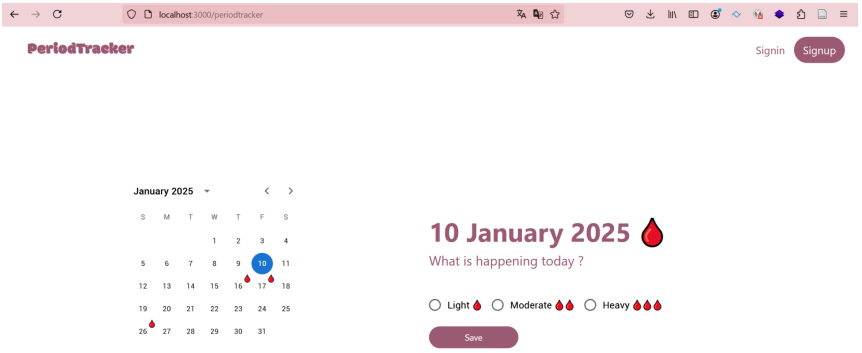
2. La page d'inscription :



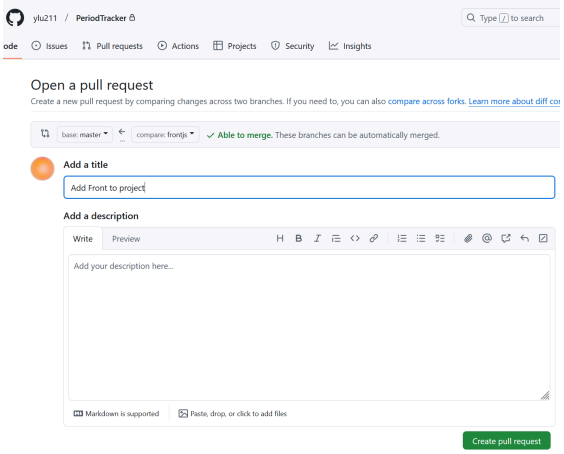
3. La page de connexion :

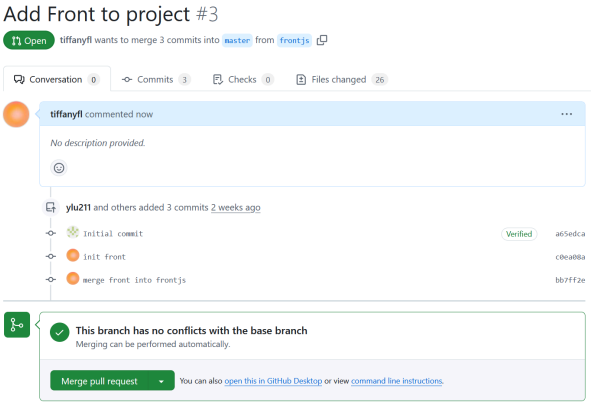


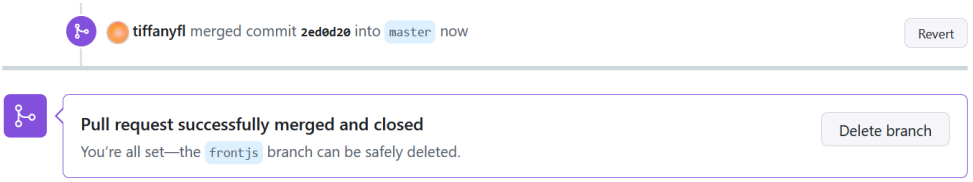
4. La page de suivi du cycle menstruel :



Nous avons travaillé sur github avec des branches, ce qui permet de séparer distinctement le travail en cours. Après avoir créer une branche, nous devons donc la pousser sur le dépôt distant et effectuer une pull request.







3. Création des images Docker

Après avoir développé nos différents services, nous avons créé des images Docker pour chaque service avec des Dockerfiles.

```
# Utilisation de l'image de base
FROM node:22-alpine

# Définir le répertoire de travail dans le conteneur
WORKDIR /app

# Copier les fichiers nécessaires dans le conteneur
COPY package*.json ./

# Installer les dépendances
RUN npm install

# Copie du reste des fichiers de l'application
COPY . .

# Build de l'application NestJs
RUN npm run build

# Exposer le port utilisé par le seveur
EXPOSE 3001

# Lancer le serveur
CMD ["npm", "run", "start"]
```

Dockerfile du service back

```
# Use the latest LTS version of Node.js
FROM node:22-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of your application files
COPY . .

# Expose the port your app runs on
EXPOSE 3000

# Define the command to run your app
CMD ["npm", "start"]
```

Dockerfile du service front

Chaque image a par la suite été publié sur Docker Hub et sont trouvables ci-après :

- https://hub.docker.com/r/yvonnelu/backend-backend/tags
- https://hub.docker.com/r/ynafi/periodtracker-front/tags

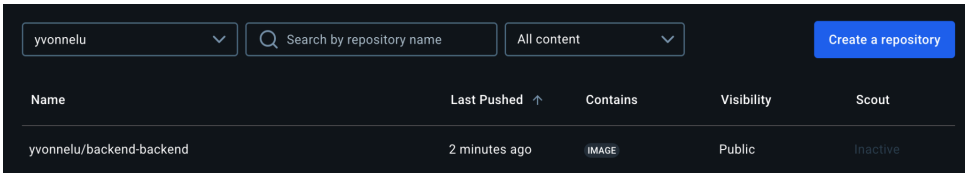


Image Docker du back sur Docker Hub

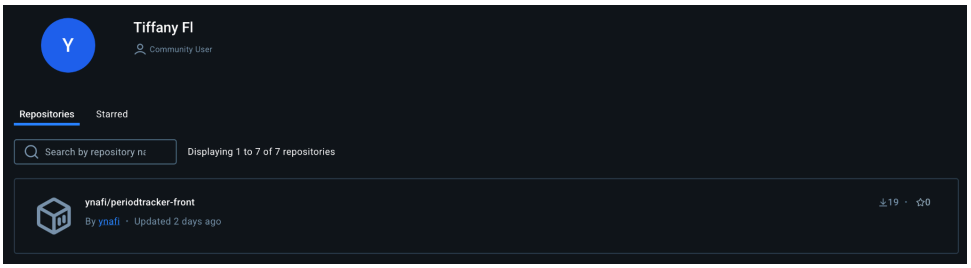


Image Docker du front sur Docker Hub

Par ailleurs, afin d'assurer que le back se lance avec la base de donnée, un docker-compose a également été réalisé. Pour tester les requêtes de l'API back, il est donc possible d'effectuer un docker compose up -d --build.

```
→  PeriodTracker git:(master) ✗ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
yvonnelu/backend-backend      latest    5dac471c0bb2   31 hours ago     304MB
ynafi/periodtracker-front     latest    e7c530b7d2d5   33 hours ago     930MB
```

Images Docker réalisés

4. Déploiement Kubernetes

Nous avons déployé nos différents services en utilisant un fichier de configuration YAML (`front-back-db.yml`), permettant d'orchestrer l'ensemble de l'infrastructure avec Kubernetes. Ce fichier définit le déploiement du frontend (React), du backend (NestJS) et de la base de données (MongoDB) sous forme de pods, gérés par des Deployments et exposés via des Services. Nous avons également configuré des ConfigMap pour stocker les variables d'environnement non sensibles, comme l'URI de connexion à la base de données et des Secret pour les données sensibles, comme le JWT secret.

```
→  PeriodTracker git:(master) ✗ kubectl get all
NAME                                          READY   STATUS    RESTARTS      AGE
pod/backend-f657cd99d-rsj5w                   2/2     Running   2 (92m ago)   26h
pod/mongodb-77fdf745dd-8kzr5                  2/2     Running   2 (92m ago)   26h
pod/periodtracker-front-5585485748-pphfj      2/2     Running   2 (92m ago)   26h

NAME                                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/backend-service                  ClusterIP   10.104.215.76    <none>        3001/TCP         31h
service/kubernetes                       ClusterIP   10.96.0.1        <none>        443/TCP          32h
service/mongodb                          ClusterIP   10.105.69.75     <none>        27017/TCP        27h
service/periodtracker-front-service      NodePort    10.109.194.151   <none>        3000:32058/TCP   31h

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend                  1/1     1            1           31h
deployment.apps/mongodb                  1/1     1            1           31h
deployment.apps/periodtracker-front      1/1     1            1           31h

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-f657cd99d                   1         1         1       31h
replicaset.apps/mongodb-77fdf745dd                  1         1         1       31h
replicaset.apps/periodtracker-front-5585485748      1         1         1       27h
```

Liste de toutes les ressources de notre application

L'Istio Gateway que nous avons mis en place permet de gérer le trafic entrant vers notre application en contrôlant l'accès aux services déployés sur Kubernetes. Elle joue un rôle clé dans l'exposition du frontend et du backend en définissant une entrée unique pour les requêtes externes.

Associée à un VirtualService, la Gateway route les requêtes selon leur chemin :

- Les appels vers /api/ sont redirigés vers le backend.
- Les autres requêtes sont envoyées vers le frontend.

5. Sécurisation du Cluster

Pour sécuriser notre cluster, nous avons utilisé différentes méthodes.

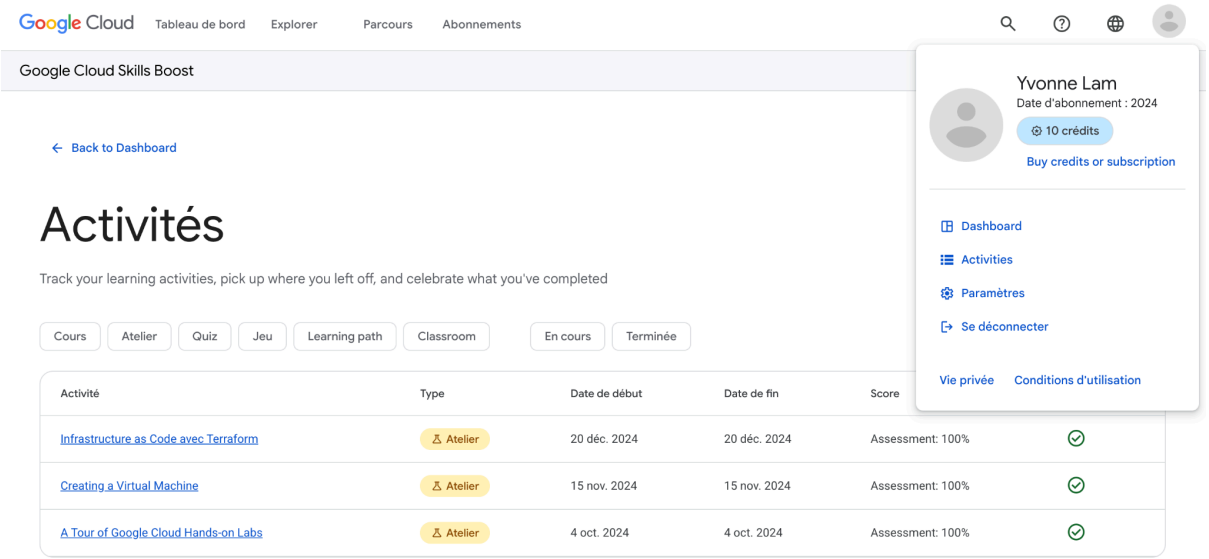
Tout d'abord, nous avons implémenté **mTLS** dans notre infrastructure Kubernetes, via Istio, ce qui permet d'assurer une communication sécurisée entre les différents services. En activant mTLS en mode STRICT dans la ressource PeerAuthentication, nous imposons un chiffrement de bout en bout et une authentification mutuelle entre les microservices (backend, frontend et base de données).

De plus, les DestinationRules appliquées sur chaque service spécifient que le trafic interne utilise ISTIO_MUTUAL, garantissant que seules les connexions authentifiées et chiffrées sont acceptées.

Ensuite, nous avons mis en place **RBAC** pour sécuriser et restreindre l'accès aux ressources sensibles. Pour cela, nous avons défini un ServiceAccount spécifique au backend, auquel nous avons attribué un Role limitant ses permissions aux actions get, list et watch sur les ressources ConfigMaps, Secrets, Pods et Services. Ce rôle est ensuite lié au ServiceAccount via un RoleBinding, ce qui garantit que seules les permissions nécessaires sont accordées au backend.

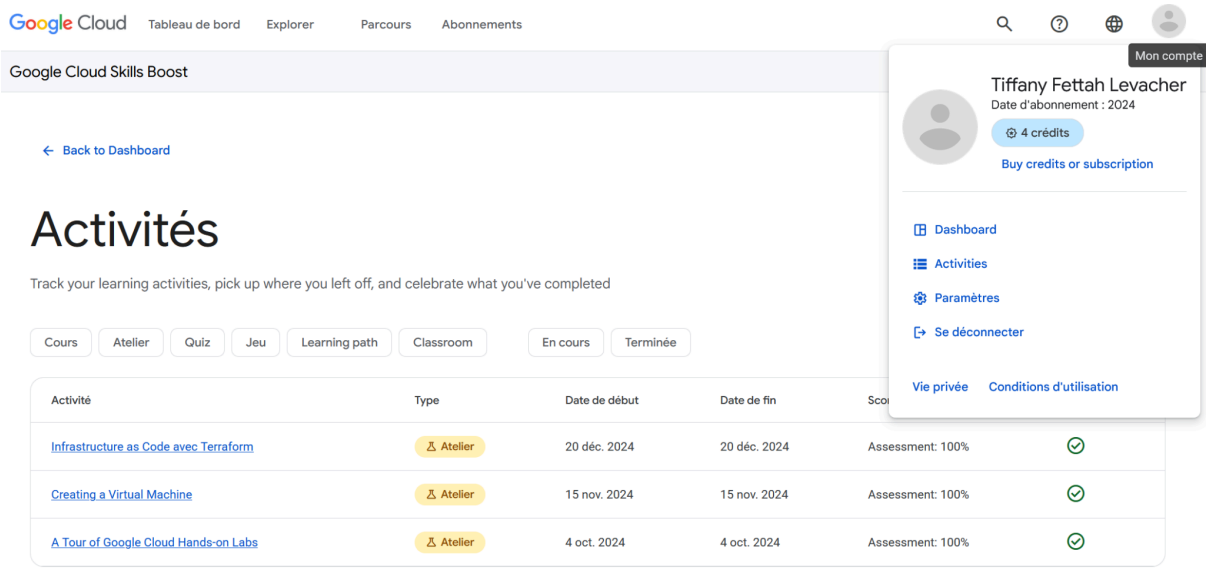
6. Projets Google Labs

a. Google labs - Yvonne Lam



Activités du profil - Yvonne LAM

b. Google labs - Tiffany Fettah Levacher



Activités du profil - Tiffany Fettah Levacher