Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER-PROJET

Certification professionnelle visée

Exploiter les outils de développement de la chaîne DEVOPS

Sommaire

Table des matières

Liste des compétences du référentiel

A1. AMELIORATION DU PROCESSUS DE DEVELOPPEMENT D'APPLICATIONS A L'AIDE DES PRINCIPES DE L'INTEGRATION CONTINUE

C1. Assurer le versionnage d'un code source d'une application organisée en fonctionnalités et lots à l'aide d'un logiciel de contrôle de version de manière à garantir la fiabilité du code source dans un environnement multi-contributeurs
C2. Contrôler l'exécution du code source à l'aide de tests et d'outils d'analyses statiques du code source afin de minimiser le risque d'erreur dans un contexte de livraison continue
C3. Automatiser les phases de tests unitaires et d'analyses statiques du code source lors du partage des sources à l'aide d'un outil d'intégration continue de manière à prévenir les erreurs potentielles

A2. MISE EN ŒUVRE DES CONDITIONS PREALABLES A LA LIVRAISON CONTINUE

C4. Concevoir un processus de livraison continue à l'aide d'outils d'automatisation de manière à l'intégrer au processus de développement
C5. Développer l'architecture d'une application en micro-services à l'aide d'outils et de bibliothèques logicielles adaptées afin de réduire la complexité globale du système

A3. CONCEPTION ET MISE EN ŒUVRE D'UN SYSTEME DE VEILLE TECHNOLOGIQUE POUR AIDER A LA PRISE DE DECISION

C6. Concevoir un système de veille technologique permettant la collecte, la classification, l'analyse et la diffusion de l'information aux différents acteurs de l'organisation afin d'améliorer la prise de décisions techniques

A4. DIFFUSION DES METHODES DEVOPS AUPRES DES EQUIPES DE L'ENTREPRISE/ORGANISATION

C7. Accompagner les collaborateurs au sein de l'équipe projet dans la sensibilisation et l'acculturation des méthodes d'organisation et de production DEVOPS de manière à optimiser le cycle de livraison d'un projet

Introduction

Dans le cadre de notre année de formation, nous avons deux examens à présenter : le titre CDA et la certification DEVOPS. Pour cela, nous avons réalisé un projet fil rouge qui nous a permis de mettre en lumière les compétences qui ont fait de nous des développeurs et concepteurs d'applications, mais également d'utiliser les outils et la méthodologie DEVOPS sur nos projets.

C'est pour ce dernier point que s'inscrit ce rapport : préciser les logiques DEVOPS que nous avons mises en œuvre sur le projet chef d'œuvre et en quoi cela nous a aidé.

Le projet a débuté tôt dans l'année et nous n'avions pas encore tous les outils pour mener à bien ce projet. Mais à la lecture de ce dossier, mon objectif est de vous démontrer que je possède la connaissance générale et les capacités, de mettre en œuvre une démarche professionnelle et une utilisation judicieuse des outils DEVOPS.

La problématique abordée concerne le défi rencontré par les associations caritatives, à savoir le faible niveau de dons, souvent motivé par des préoccupations telles que "Je manque de transparence quant à l'utilisation de mes contributions" ou "Je crains que mes dons ne soient détournés pour des achats non liés à la cause". Pour relever ce défi, nous avons opté pour une approche technologique solide et contemporaine, en choisissant de développer une Progressive Web App (PWA) utilisant React.

Au moment de l'écriture de ce rapport, nous avons pu avancer sur le back de notre projet permettant l'achat et la mise en panier des articles et estimons la réalisation du projet à hauteur de 50 %.

Néanmoins, nous avons mis le point sur les outils et les méthodes de travail qui nous ont permis de répondre au mieux à vos attentes et à la présentation orale qui se présentera deux semaines plus tard.

Mon projet chef d'œuvre a été réalisé avec Zélie.

De mon côté, j'ai préféré la partie back/bdd et nous ferons attention à bien rester synchronisés pour valider ensemble les différents points du référentiel.

Je vous présenterais mon projet avec chaque compétence validée ensemble de la 1ère à la dernière.

Pour chacune de ces étapes, je présenterai :

- le cheminement de pensée qui m'a amené à sélectionner une solution
- la réalisation de cette solution

C1. utilisation de Git

Git permet de mettre en ligne mon code et bien d'autres choses encore.

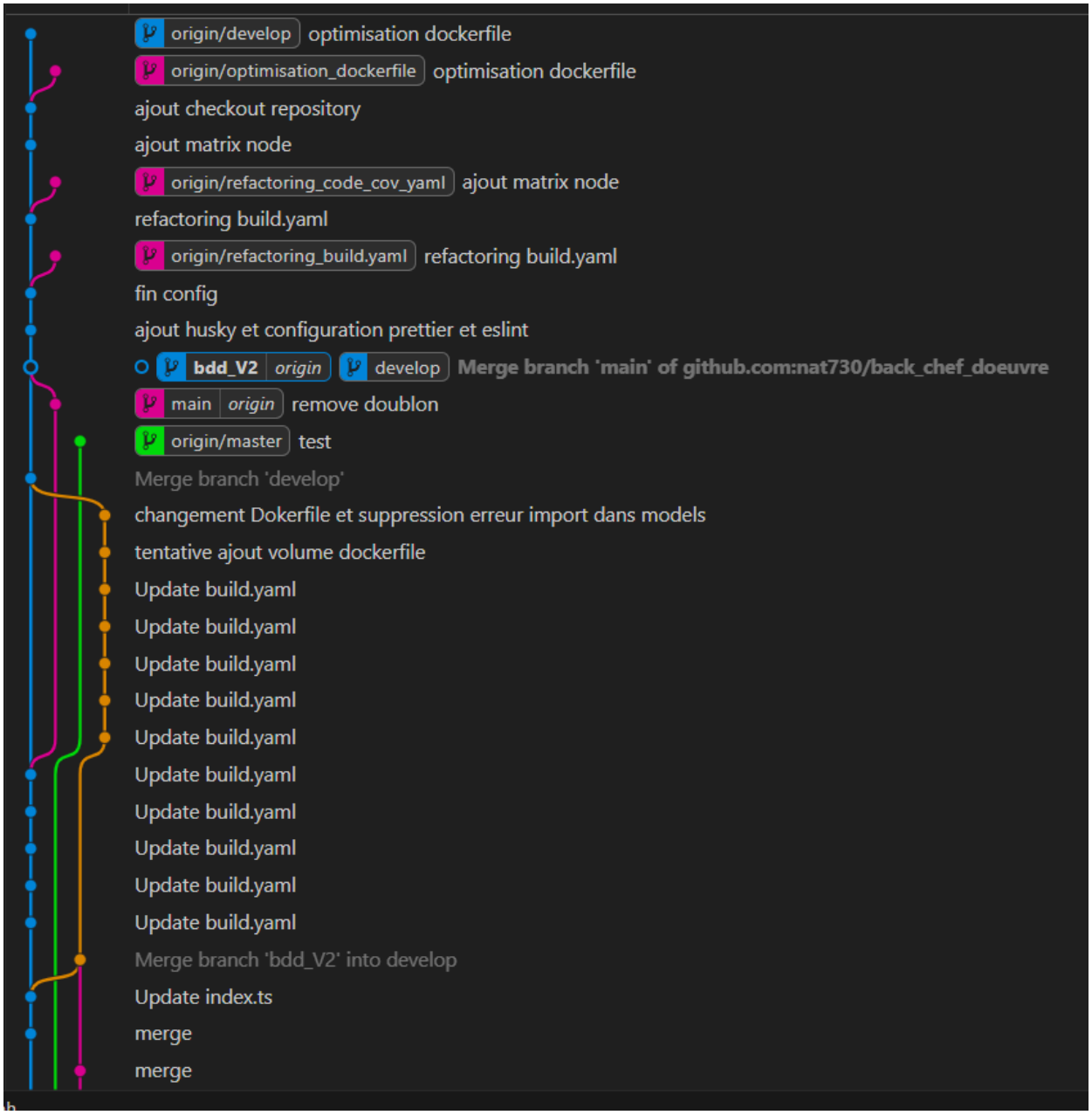
```
/home/nat/dev/chef_doeuvre/front_chef_doeuvre:git add .
/home/nat/dev/chef_doeuvre/front_chef_doeuvre:git commit -m test
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
/home/nat/dev/chef_doeuvre/front_chef_doeuvre:git push
Everything up-to-date
```

Voici les commandes Git essentielles : git add <nomdesfichiers>, git commit -m <nom_du_commit>, git push, qui permettent de mettre en ligne le code local.

Sur cette image, on peut voir les trois commandes avec l'indication "on branch main, your branch is up-to-date with origin/main", ce qui signifie que le code est similaire en local et en ligne. Le "." après git add est une expression régulière (regex) qui permet de sélectionner tous les documents à partir de ce dossier.

Git permet également de conserver un historique de ses commits.

```
commit 1c8cd2faaff48914b666974e043196ee4b5d96c1 (HEAD -> storybook, origin/storybook)
Author: nat730 <81553917+nat730@users.noreply.github.com>
Date:   Mon Feb 12 17:05:57 2024 +0100

    Delete package-lock.json

commit 63420943d599686f59e0d592f6b968edae09e1b6
Author: nat730 <natanaelcolart@gmail.com>
Date:   Mon Feb 12 17:04:55 2024 +0100

    repair

commit ccf757c4dd4ab713c475fcd2cfc6cf3932e6fdcc
Author: nat730 <81553917+nat730@users.noreply.github.com>
Date:   Mon Feb 12 17:03:07 2024 +0100

    Delete package.json

commit c95428dd29ba0f1990d5e2fdbf92369d62f690c9
Author: nat730 <natanaelcolart@gmail.com>
Date:   Mon Feb 12 16:55:38 2024 +0100

    ajout pacakge lock to gitignore

commit 2d4928d18b19bddf0b526ff6981b5ce16c9fbe67
Author: nat730 <natanaelcolart@gmail.com>
Date:   Mon Feb 12 16:51:04 2024 +0100

    add storybook
```


origin/develop optimisation dockerfile
origin/optimisation_dockerfile optimisation dockerfile
ajout checkout repository
ajout matrix node
origin/refactoring_code_cov_yaml ajout matrix node
refactoring build.yaml
origin/refactoring_build.yaml refactoring build.yaml
fin config
ajout husky et configuration prettier et eslint
bdd_V2 origin develop Merge branch 'main' of github.com:nat730/back_chef_doeuvre
main origin remove doublon
origin/master test
Merge branch 'develop'
changement Dokerfile et suppression erreur import dans models
tentative ajout volume dockerfile
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Update build.yaml
Merge branch 'bdd_V2' into develop
Update index.ts
merge
merge

J'ai utilisé trois fonctionnalités principales avec GitHub :

a. Les pull requests (PR)

J'ai configuré quatre paramètres :

restrict deletions : Seuls les utilisateurs disposant de permissions "bypass" sont autorisés à supprimer les branches communes.

Reequire linear history :J'ai restreint les possibilités de fusion (merge) pour conserver un historique des commits.

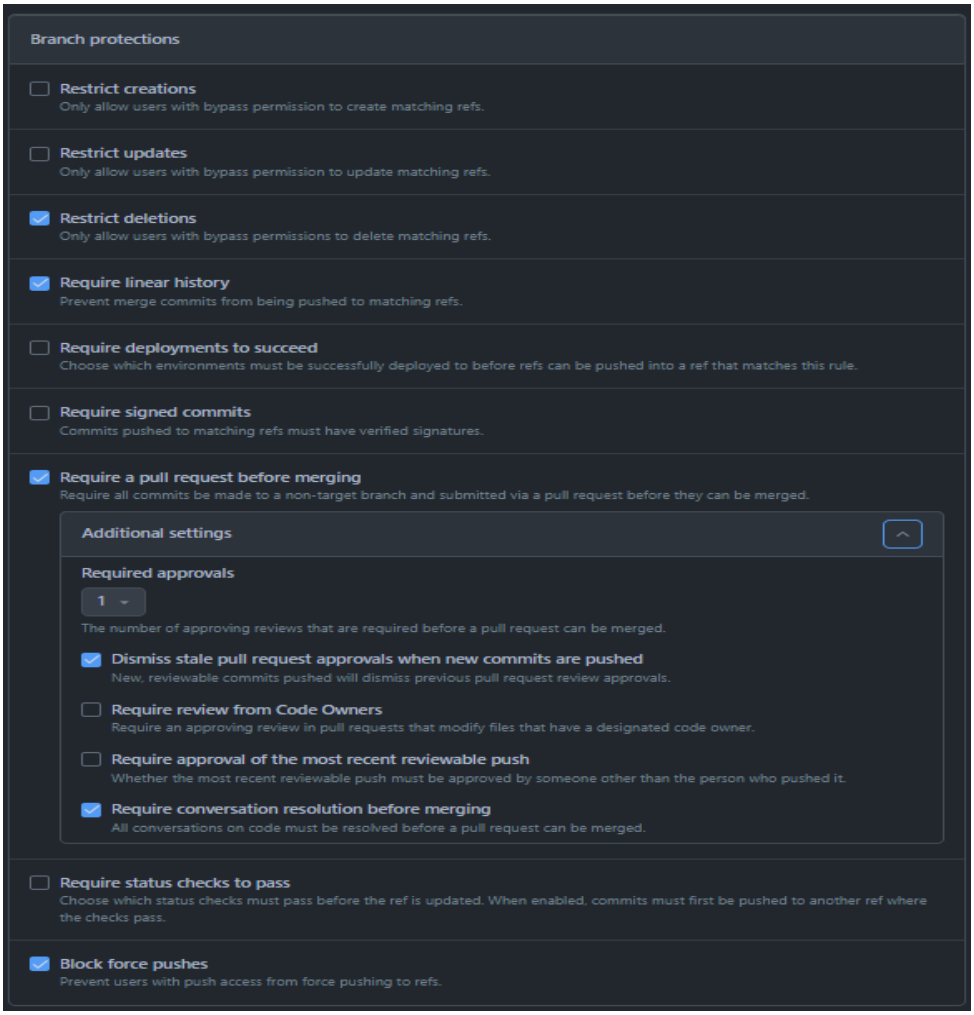
Require J'ai rendu obligatoire la création d'une PR pour éviter les push qui pourraient altérer le code.

Sous ces règles principales, j'ai mis en place des sous-règles pour les PR :

Dès qu'une mise à jour est effectuée sur une branche en attente de validation, cette règle met automatiquement à jour la PR avec les modifications

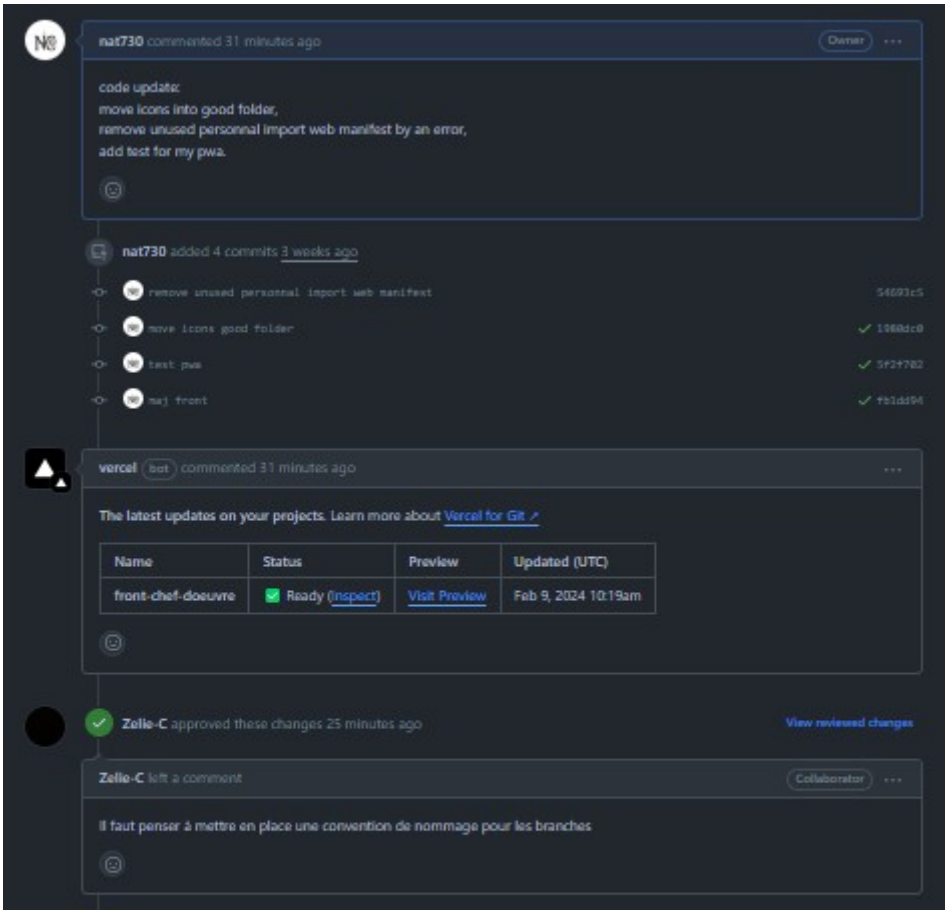
Tout commentaire doit être résolu pour que la PR soit acceptée

La dernière règle empêche les utilisateurs d'utiliser push -f pour contourner les règles établies.



Voici à quoi ressemble une PR :

Une partie commentaire décrivant les modifications apportées par la PR (idéalement en anglais).
Les commits de chaque modification, accompagnés d'un titre explicatif.
La mise en développement, le cas échéant (comme le déploiement sur Vercel dès que la PR est acceptée).
Un commentaire indiquant les raisons du refus ou de l'acceptation.



C2. les tests

tests unitaires :

Les tests unitaires permettent de vérifier si un composant fonctionne individuellement, mais cela ne suffit pas ; nous avons également besoin de tests d'intégration. Un exemple concret dans notre quotidien :

Étant en phase de prototypage, je n'ai pas encore eu le temps d'effectuer mes tests unitaires. Je pourrais en discuter plus en détail à l'oral si nécessaire.

Test End to End (e2e):

Les tests E2E permettent de simuler le comportement réel de l'utilisateur. Par exemple, en demandant au code de cliquer sur le bouton "Panier" puis de vérifier s'il arrive bien sur la page "Panier".

```
import { test, expect } from '@playwright/test';

test('test', async ({ page }) => {
  await page.goto('http://localhost:5173/');
  await page.getByRole('button', { name: '18' }).first().click();
  await page.getByRole('button', { name: 'panier' }).click();

  const patesElement = page.getByText('pates');
  const quantityTextElement = page.getByText('Quantity:');

  expect(patesElement).not.toBeNull();
  expect(quantityTextElement).not.toBeNull();
});
```

Eexemple de test unitaire

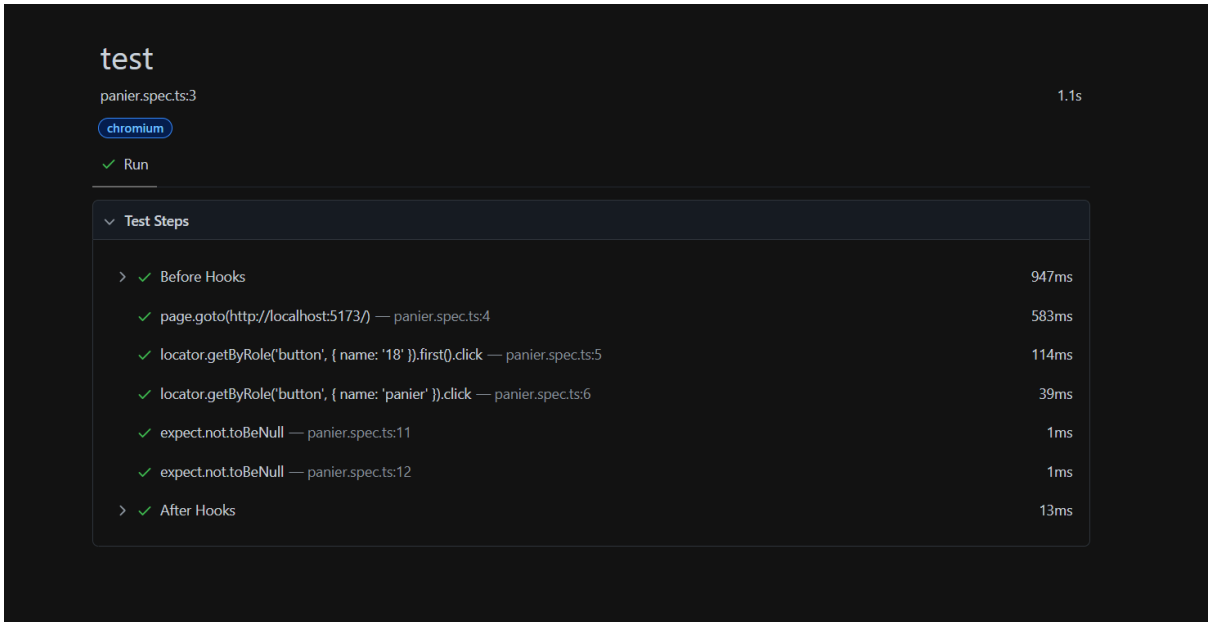
Les "await" correspondent aux instructions données au code. Dans cet exemple (de haut en bas) :

Le code est instruit pour se rendre sur le serveur de test.Ensuite, il est demandé d'interagir en premier lieu avec le bouton nommé "18". Par la suite, il est demandé d'interagir avec le bouton nommé "Panier".

Concernant les "expect" qui représentent les résultats attendus :

Nous vérifions que la constante "pateElement" n'est pas nulle, ce qui indique la présence de l'élément associé aux pâtes. De même, nous vérifions que la constante "quantityTextElement" n'est pas nulle, ce qui indique la présence de l'élément associé à la quantité.

```
/home/nat/dev/chef_doeuvre/front_chef_doeuvre:npx playwright test

Running 1 test using 1 worker
  1 passed (1.2s)

To open last HTML report run:

  npx playwright show-report
```



Push reussi sur vercel

C3. utilisation de Github

GitHub Actions me sert essentiellement à faire ma CI/CD. J'effectue des push sur Docker et/ou sur Vercel. Voici à quoi ressemble du code YAML pour push sur Vercel :

le nom du test

dès qu'il y a un push ou une pull request sur la branche main.

Si le push n'est pas fait au bout de 1h, l'annule.

Je fais ça sur la dernière version d'Ubuntu et les dépendances utilisées.

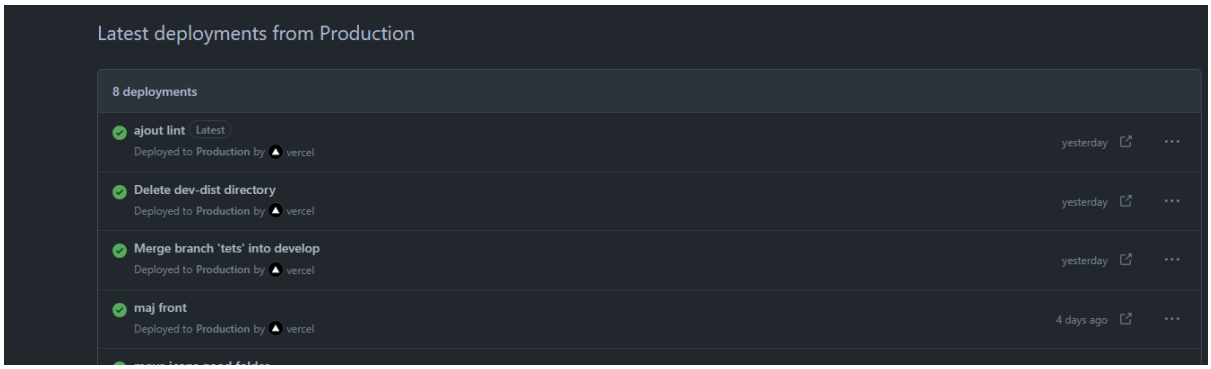
Toutes les paramètres d'installation sont spécifiés (on utilise la version 18 de Node.js, on installe les dépendances nécessaires).

```yaml
name: Playwright Tests
on:
  push:
    branches: main
  pull_request:
    branches: main
jobs:
  test:
    timeout-minutes: 60
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - uses: actions/setup-node@v3
      with:
        node-version: 18
    - name: Install dependencies
      run: npm ci
    - name: Install Playwright Browsers
      run: npx playwright install --with-deps
    - name: Run Playwright tests
      run: npx playwright test
    - uses: actions/upload-artifact@v3
      if: always()
      with:
        name: playwright-report
        path: playwright-report/
        retention-days: 30
```

C4. Déploiement continue (CD)

On met en place l'intégration continue sur Vercel et Render grâce à GitHub et DockerHub.

Dès qu'un code est push sur la branche "develop" (qui sert à vérifier que le site fonctionne sans bug avant de le mettre en ligne) ou sur "main" (la branche contenant le livrable), une image Docker est automatiquement créée. Cette image est ensuite poussée sur DockerHub. Render récupère alors cette image et la met en ligne. Pour Vercel, dès que les tests passent grâce à la CI (voir section précédente), le code est mis en ligne, permettant ainsi d'accéder au site.



Déploiement sur Vercel.

```yaml
name: Build

on:
  push:
    branches:
      - "main"
      - "develop"

jobs:
  build:
    runs-on: ubuntu-latest

    steps:

      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Build Docker image
        run: |
          docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}
          docker build -t ${{ secrets.DOCKERHUB_USERNAME }}/back_projet:latest-${{ github.ref_name
}}-${{ runner.os }} .
          docker push ${{ secrets.DOCKERHUB_USERNAME }}/back_projet:latest-${{ github.ref_name }}-${{
runner.os }}
```

Création et push de l'image sur DockerHub.

C6. Méthode pour notre veille

Je mène ma veille grâce à diverses ressources, outils et méthodes :

Vidéos YouTube : Je commence par regarder des vidéos sur YouTube où je note les informations importantes dans un document Markdown.

Analyse des sources : Ensuite, je recherche des informations sur la personne ou l'entreprise qui a créé la vidéo pour comprendre leur approche du sujet. Par exemple, un commercial n'abordera pas le sujet de la même manière qu'un développeur.

Consultation de Reddit : Si j'ai encore du temps disponible, je me renseigne sur des sites comme Reddit pour obtenir des informations supplémentaires ou des perspectives différentes.

Comparaison des informations : Si j'ai déjà vu le sujet en cours ou lors de veilles précédentes, je compare les nouvelles informations avec les anciennes pour enrichir ma compréhension et mettre à jour mes connaissances.

```
# Résumé

Le narrateur explore la différence entre les applications monolithiques et les applications microservices dans le domaine du développement informatique.

Il prend l'exemple d'une application d'entreprise monolithique qui gère divers aspects tels que la comptabilité, la finance, les RH, etc. Un inconvénient notable de ce type d'application est la nécessité de déployer une instance complète de l'application lorsque des parties spécifiques, comme le module de comptabilité, sont fortement sollicitées.

En contraste avec l'architecture monolithique, il explique que découper une application en petits modules, dans le cas des microservices, permet à chaque développeur de se spécialiser dans son domaine. Cela favorise l'agilité en permettant aux développeurs de choisir le langage de programmation le plus adapté à chaque module.

L'avantage clé des microservices réside dans la flexibilité et la facilité de réutilisation des modules. Chaque développeur peut travailler de manière autonome sur son module, simplifiant ainsi la maintenance et la scalabilité de l'application.

Le narrateur souligne l'importance du découplage des applications dans un monde concurrentiel, où les entreprises ont besoin de solutions agiles pour répondre rapidement aux besoins des clients. Il mentionne que de nombreuses entreprises optent pour des architectures microservices compatibles avec la conteneurisation (docker).

En conclusion, le narrateur encourage les entreprises à envisager la transition des applications monolithiques aux microservices pour rester compétitives et réactives aux évolutions du marché.

Source : [Vidéo YouTube](https://www.youtube.com/watch?v=2-wzv4aJyIA) (datée du 10 janv. 2022)

**Chaîne YouTube Xavki :** Des tutoriels et formations gratuites en français sur les outils Linux, DevOps et open source, avec plus de 1400 vidéos.

**Narrateur/Dirane TAFEN - Ingénieur informatique DEVOPS :** Après l'obtention de mon master en informatique, j'ai débuté ma carrière en tant qu'ingénieur administrateur système. J'ai découvert le DevOps qui a bouleversé ma carrière, m'offrant de nouvelles opportunités et la possibilité de devenir freelance avec un revenu cinq fois supérieur. En parallèle de mes prestations quotidiennes, je dispense des formations en France et à l'étranger via EazyTraining, un centre de formation à distance pour apprendre le DevOps, entre autres.

![Alt text](image.png)
```

Exemple d'une veille

C7. Dev ops comme méthode de travail

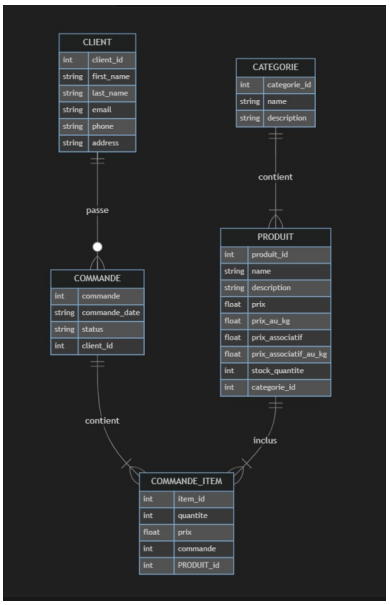
Nous avons utilisé la méthodologie Scrum pour notre projet. Nous nous réunissons une fois par jour pour discuter de nos avancées et nous nous rassemblons régulièrement pour communiquer sur les problèmes rencontrés au cours de la journée.

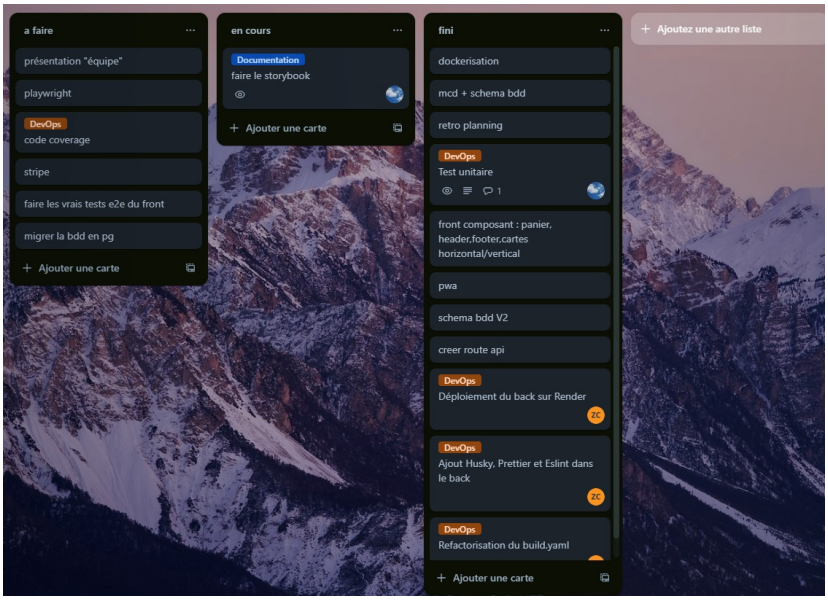
Pour organiser notre travail, nous avons utilisé deux types de tableaux Kanban : des prévisionnels pour estimer notre temps de travail et des réels pour visualiser ce qu'il reste à faire et ce que nous avons déjà accompli.

Nous avons également mis en place des "sprints" qui nous permettent de planifier nos tâches à court terme.

En plus de cela, nous utilisons Trello, qui nous permet de suivre nos activités tâche par tâche, offrant ainsi une vision claire du pourcentage de travail effectué.

Pour faciliter la compréhension du projet par les non-développeurs, nous avons créé un résumé ainsi qu'un schéma conceptuel des données (MCD).





Conclusion

Grâce à cette étude de cas, j'ai pu approfondir mes connaissances sur Jenkins, un outil qui permet d'automatiser la CI/CD à grande échelle, ainsi que sur Kubernetes et Swarm, des outils de gestion de conteneurs Docker.

J'ai trouvé enrichissante la démarche consistant à documenter nos explications, même si je préfère généralement la mise en pratique à la théorie.

Dans le futur, j'ai l'intention d'appliquer ce que j'ai appris lors de cette étude de cas, notamment en explorant davantage Kubernetes et Jenkins.

Mon principal défi a été la compréhension du sujet au début, qui me semblait trop vague, mais qui a rapidement été clarifié par nos formateurs. Je rencontre également des difficultés avec mes problèmes de concentration pour produire des documents "propres".