Laurie BERTHON

ECORIDE

Documentation technique

Introduction

Ce document a pour but de présenter les étapes de réalisation de l'application web Ecoride.

Pour rappel :

Ecoride est une startup nouvellement crée en France. Elle a pour objectif de réduire l'impact environnemental des déplacements en encourageant le covoiturage. L'aspect écologique est un élément central de l'existence d'Ecoride. Par la création d'une plateforme intuitive et moderne, José, le directeur technique, souhaite faire de son entreprise la principale plateforme de covoiturage pour les voyageurs soucieux de l'environnement et recherchant des solutions économiques pour leurs trajets en voiture.

Nous avons donc été missionnés pour le développement de cette application web avec un besoin précis :

- Page d'accueil présentant l'entreprise et comprenant un menu, les mentions légales et l'adresse mail de l'entreprise ainsi qu'une possibilité de recherche d'itinéraires
- Le menu doit permettre l'accès à la page de contact, à la connexion, à la recherche de covoiturages et le retour à l'accueil
- Les covoiturages ainsi que leurs détails doivent être accessible aux visiteurs
- Il doit être possible de filtrer la recherche de covoiturages en fonction du prix, de la durée, de l'aspect écologique et de la note du conducteur
- La participation à un covoiturage n'est accessible qu'aux utilisateurs connectés de même que la création d'un trajet
- Un système de rôle doit être mis en place pour gérer les types d'utilisateur (passager, conducteur ou les deux)
- Les utilisateurs doivent pouvoir gérer leur profil et leurs actions (trajet pour les conducteurs et participation pour les passagers
- Un espace dédié à l'administrateur doit lui permettre d'accéder à des statistiques et gérer la plateforme ainsi que les comptes employés
- Un espace dédié aux employés créés par l'administrateur doit leur permettre de modérer les avis et régler les litiges en cas de passager mécontent.

Conception

1. Identité visuelle

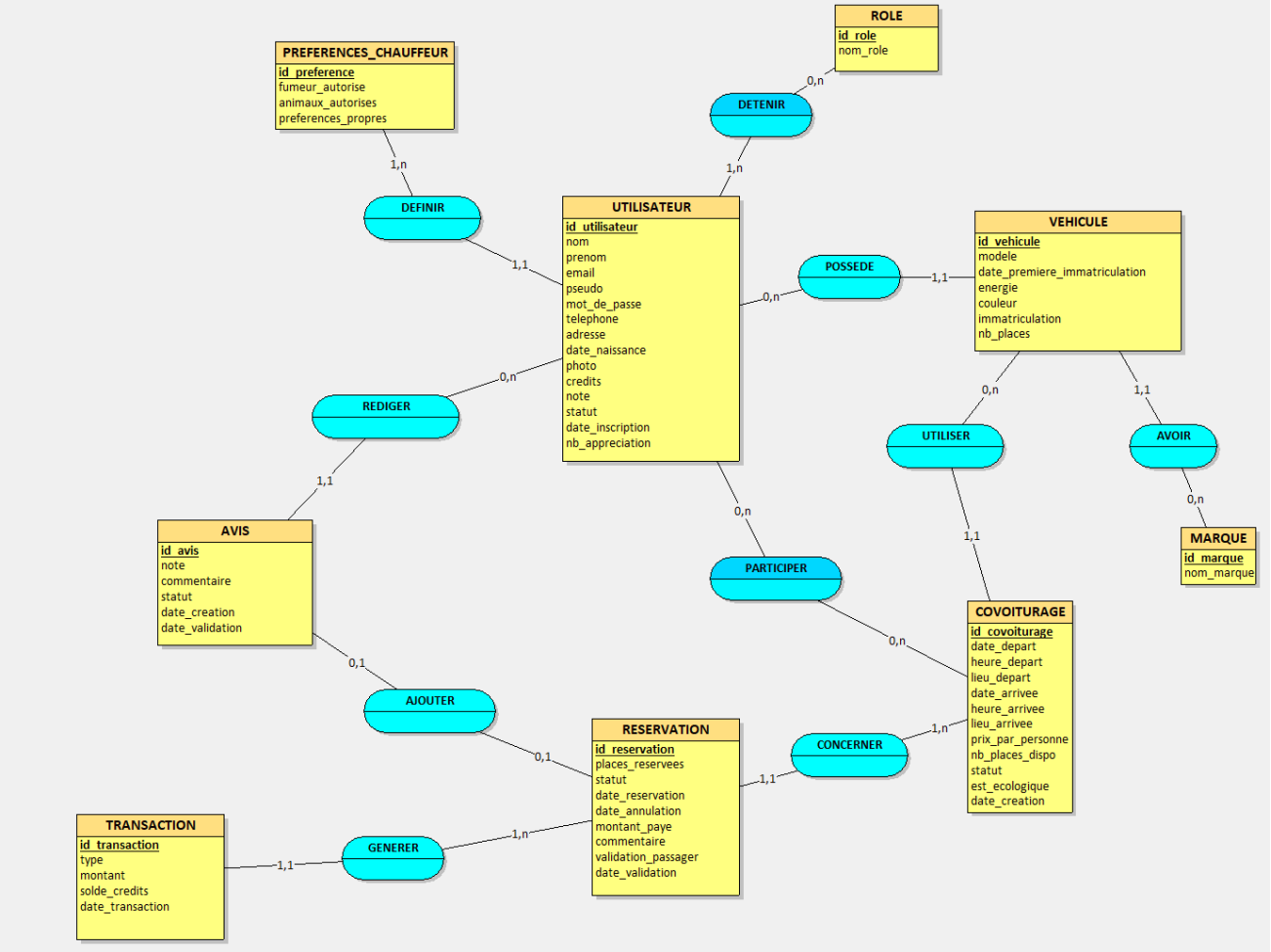
Mes analyses ont commencé par la partie design. Dans un premier temps j'ai fait des recherches sur les concurrents (peu nombreux sur le marché) afin d'analyser leur plateforme notamment BlaBlaCar qui est le concurrent principal. Je me suis donc inspirée de ces concurrents pour créer mon univers graphique.

José précise que l'aspect écologique est très important et il souhaite que la plateforme reflète le respect de l'environnement avec des couleurs représentant la nature et l'écologie. J'ai donc fait une sélection de couleurs douces, pastel (voir charte graphique).

J'ai donc créé un logo reprenant les couleurs de l'univers graphique souhaité. Le logo a été réalisé avec Illustrator.

2. Diagrammes

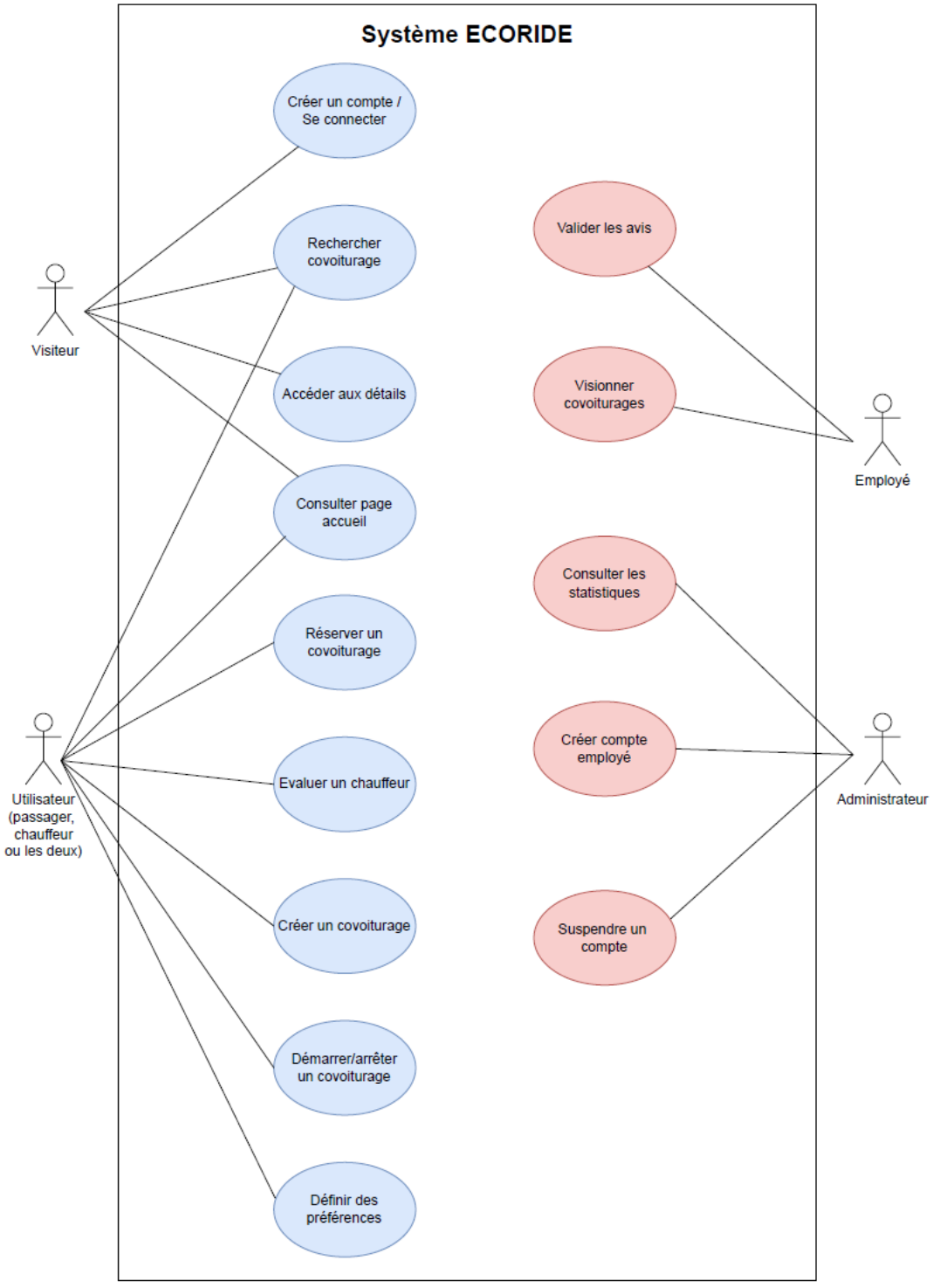
A partir de l'analyse du besoin et du schéma fourni dans l'énoncé, j'ai réalisé mon modèle conceptuel de données :



J'ai fait le choix de créer une entité « rôle » pour gérer la complexité des types d'utilisateur. L'entité « reservation » a été ajoutée afin de tracer les différentes participations aux covoiturages. L'entité « transaction » va permettre de gérer les crédits.

Il est important de noter que j'ai conservé une entité « avis » pour garder une visibilité globale mais **les avis seront gérés en base de données NoSQL**.

Un diagramme de cas d'utilisation a également été réalisé :



Pour finir, j'ai réalisé deux diagrammes de séquences :

Diagramme de séquence - covoiturage

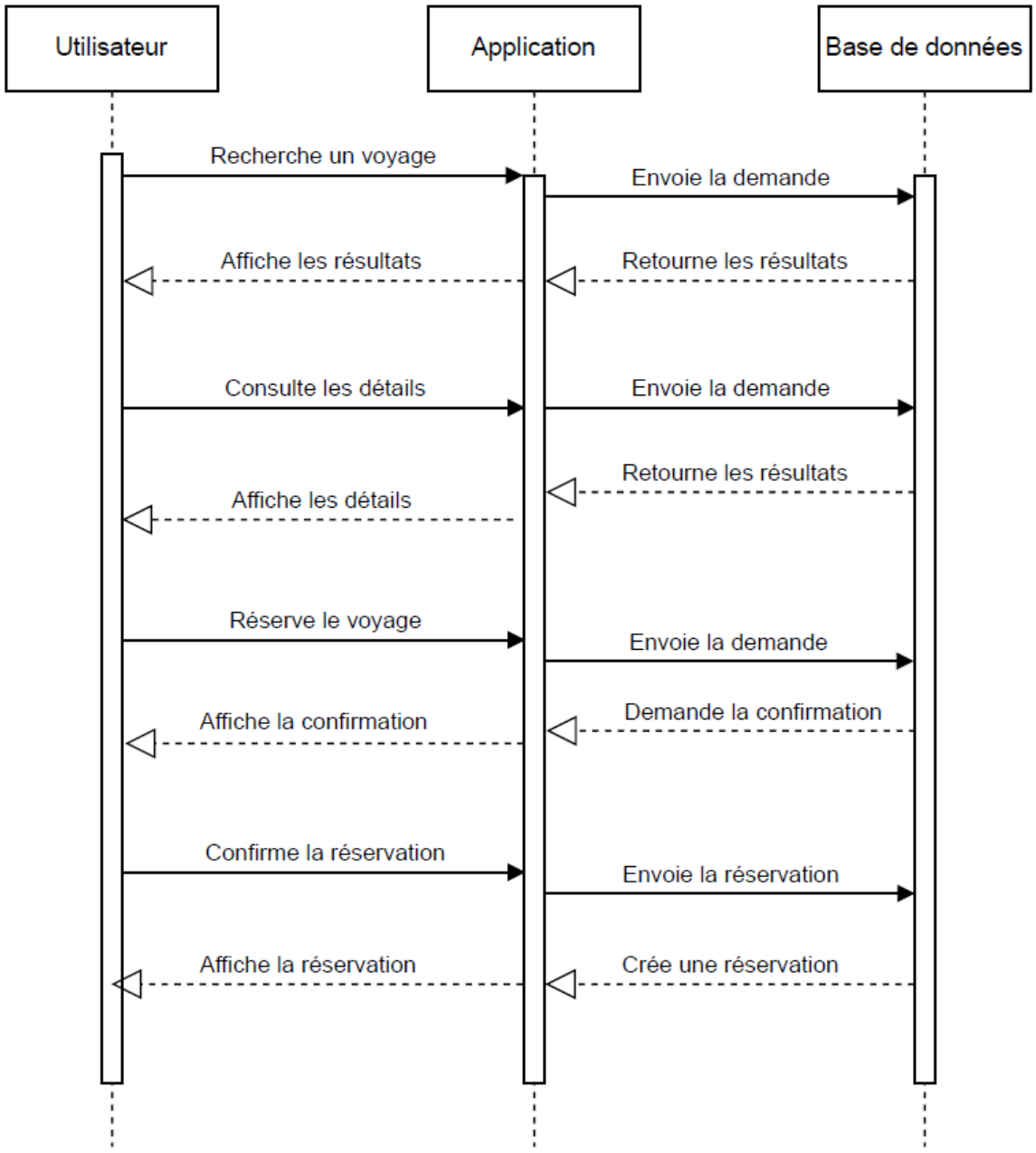
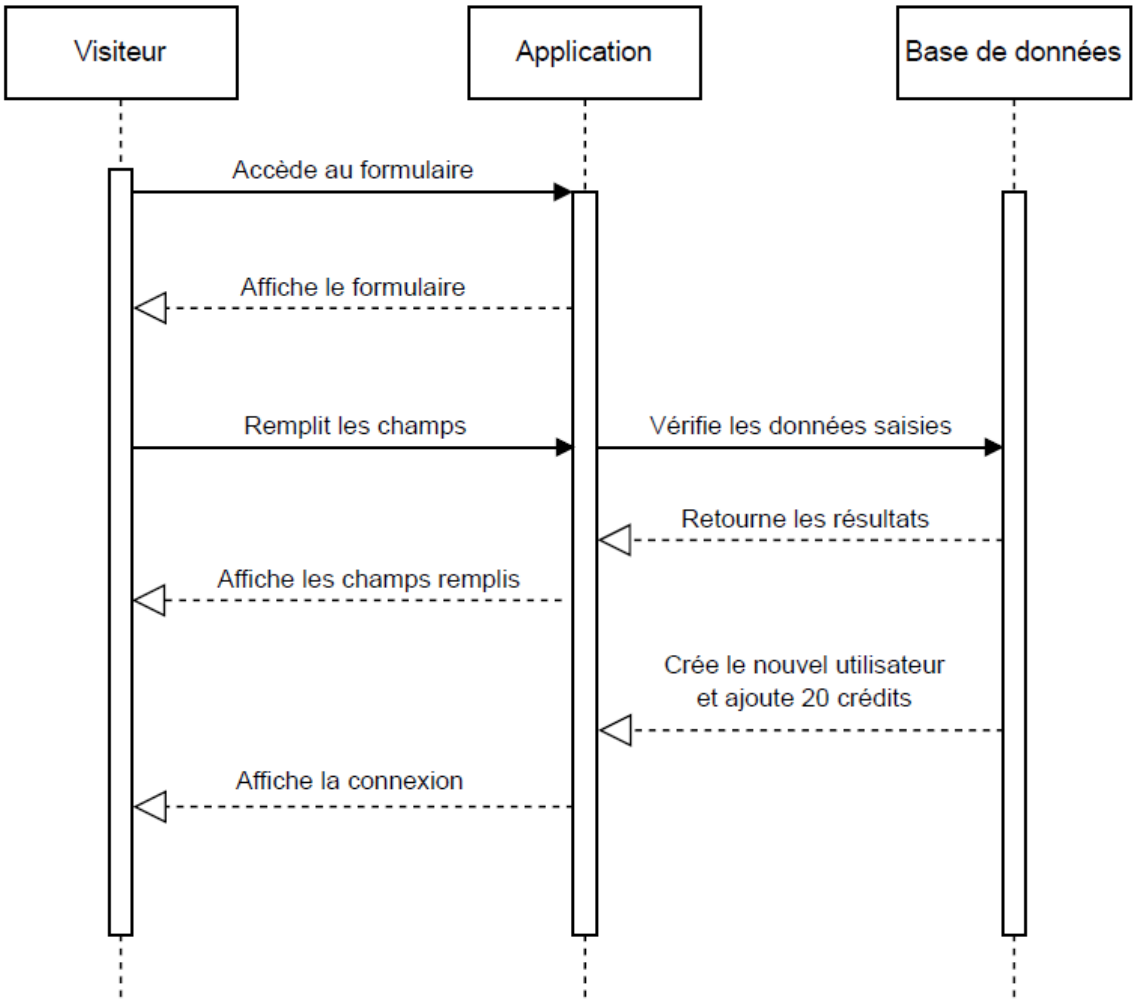


Diagramme de séquence - création de compte utilisateur



3. Maquettes

Pour terminer la conception j'ai réalisé les maquettes (wireframes et mockups) qui sont disponibles dans le dossier « maquettes ».

Réflexions initiales technologiques sur le sujet

Lors des phases d'analyse et de conception, j'ai vite découvert que le projet était complexe et que beaucoup de fonctionnalités devaient être développées pour offrir une expérience utilisateur optimale et une performance robuste.

Pour la partie Back-end, j'ai fait le choix de développer cette application en PHP vanilla afin de comprendre en profondeur le fonctionnement de ce langage avant de passer par un framework comme Symfony. De plus, cela apporte plus de contrôle sur la stabilité du projet sur le long terme.

En voyant l'ampleur du projet, j'ai décidé d'utiliser une architecture MVC afin de séparer les responsabilités. Cette façon d'organiser mon projet le rend plus clair, plus compréhensible et modifiable. Les modèles gèrent la logique métier et les accès aux données, les vues gèrent l'affichage des pages et les contrôleurs orchestrent les flux utilisateur.

Pour le stockage des données, j'ai choisi une base de données MySQL pour les données relationnelles. MySQL est une solution fiable, bien documentée et adaptée aux application web structurées. Concernant la connexion aux bases de données (MySQL et MongoDB), j'ai mis en place le pattern Singleton pour mes classes Database et MongoDb.

Pour la partie Front-end, L'interface est construite en HTML et CSS pour les styles. J'ai également utilisé Bootstrap qui facilite grandement le responsive et le style des pages. Enfin pour la partie dynamique des interfaces, j'ai utilisé Javascript pour l'interactivité côté client et le développement des filtres de recherche afin d'éviter le rechargement de page et permettre l'affichage des résultats en temps réel, ce qui optimise l'expérience utilisateur.

Enfin, afin de faciliter le déploiement et l'installation en local, j'ai utilisé Docker.

Configuration de l'environnement de travail

Prérequis :

- IDE : vscode
- Git
- Docker et Docker-compose
- Docker desktop
- Composer (gestion des dépendances)

Configuration du projet :

Avant de commencer le développement du projet, j'ai configuré mon environnement en créant les fichiers suivants :

- composer.json : pour les dépendances et l'autoload

```json
{
    "name": "projet-ecoride/ecoride-php",
    "description": "Application de covoiturage ecologique",
    "type": "project",
    "require": {
        "php": ">=8.2",
        "nikic/fast-route": "^1.3",
        "vlucas/phpdotenv": "^5.6",
        "mongodb/mongodb": "^2.1",
        "phpmailer/phpmailer": "^6.9"
    },
    "require-dev": {
        "phpunit/phpunit": "^11.4"
    },
    "autoload": {
        "psr-4": {
            "App\\": "src/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "Tests\\": "tests/"
        }
    }
}
```

- Dockerfile :

```dockerfile
# Utiliser une image php avec Apache
FROM php:8.4-apache

# Installer les dépendances et bibliothèques nécessaires
RUN apt-get update && apt-get install -y --no-install-recommends \
    libzip-dev \
    zip \
    unzip \
    git \
    libssl-dev \
    pkg-config \
    && docker-php-ext-install pdo pdo_mysql zip \
    && apt-get clean && rm -rf /var/lib/apt/lists/*

# Installer l'extension MongoDB pour PHP
RUN pecl install mongodb && docker-php-ext-enable mongodb

# Configurer Apache pour pointer vers le répertoire 'Public'
RUN sed -i 's|/var/www/html|/var/www/html/public|g' /etc/apache2/sites-available/000-default.conf

# Activer le mod_rewrite d'Apache
RUN a2enmod rewrite

# Installer Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Définir le répertoire de travail
WORKDIR /var/www/html

# Création des répertoires de stockage et de logs avec les permissions appropriées
RUN mkdir -p storage/logs public && chown -R www-data:www-data /var/www/html

# Copier le reste de l'application
COPY . .

# Changer les permissions des fichiers pour Apache
RUN chown -R www-data:www-data /var/www/html && chmod -R 755 /var/www/html

# Exposer le port 80 pour Apache
EXPOSE 80

# Démarrer Apache en mode premier plan avec ENTRYPOINT (au lieu de CMD) pour plus de sécurité
ENTRYPOINT ["apache2-foreground"]
```

- docker-compose.yml : containers

```yaml
Run All Services
services:
  Run Service
  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: ecoride-php
    restart: always
    ports:
      - "8080:80" # Accéder à l'application via le port 8080
    volumes:
      - .:/var/www/html #permet de synchroniser le code source sans refaire un build
    depends_on:
      - db
      - mongodb
    environment:
      - DB_HOST=db
      - DB_name=${DB_name}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - MONGO_HOST=mongodb
      - MONGO_PORT=27017
      - MONGO_DATABASE=${MONGO_DATABASE}
```

- Fichier .env : définir les variables d'environnement

Déploiement de l'application

Afin de déployer son application facilement et à moindre coût, Heroku est un bon choix. En effet, Heroku fonctionne avec Git et grâce au Github Student Pack, le coup est raisonnable.

Le compte sur Heroku était créé en amont. Tout le déploiement a été fait via l'interface web d'Heroku et non Heroku CLI.

Le déploiement a été réalisé en suivant les étapes suivantes :

- Création du fichier Procfile à la racine du projet :

```
Procfile
1  web: heroku-php-apache2 public/
```

- Modification de ma classe Database et MongoDb pour accéder aux variables d'environnement qui sont différentes sur Heroku tout en gardant mon application fonctionnelle en local. Mes classes permettent donc une connexion soit en local soit sur la version déployée.

- Création de l'app sur Heroku à partir de la branche main du dépôt distant

- Ajout du add-on JawsDB MySQL pour avoir une base de données qui persiste

- Configuration des variables d'environnement (différentes des variables en local) et ajout de la variable MONGO_URI récupérée sur MongoDB Atlas.

- Déploiement

- Création d'un script php temporaire pour importer la base de données (notamment les données insérées pour les tests) du projet

- Déploiement à nouveau

- Tests sur l'application déployée : test des différents utilisateurs pour vérifier le succès de l'import des données, test des différents parcours pour vérifier que tout fonctionne

Lien de l'application déployée : https://ecoride-lb-8111321be793.herokuapp.com/

L'application est fonctionnelle mais suite aux tests, des bugs se sont révélés. Ces bugs seront corrigés par la suite.

De plus, je souhaite faire des tests utilisateurs dans mon entourage afin d'optimiser au mieux cette application avec leurs différents retours.