Documentation technique

SOMMAIRE

EcoRide

1. Réflexions initiale technologique sur le sujet

Technologies choisies :

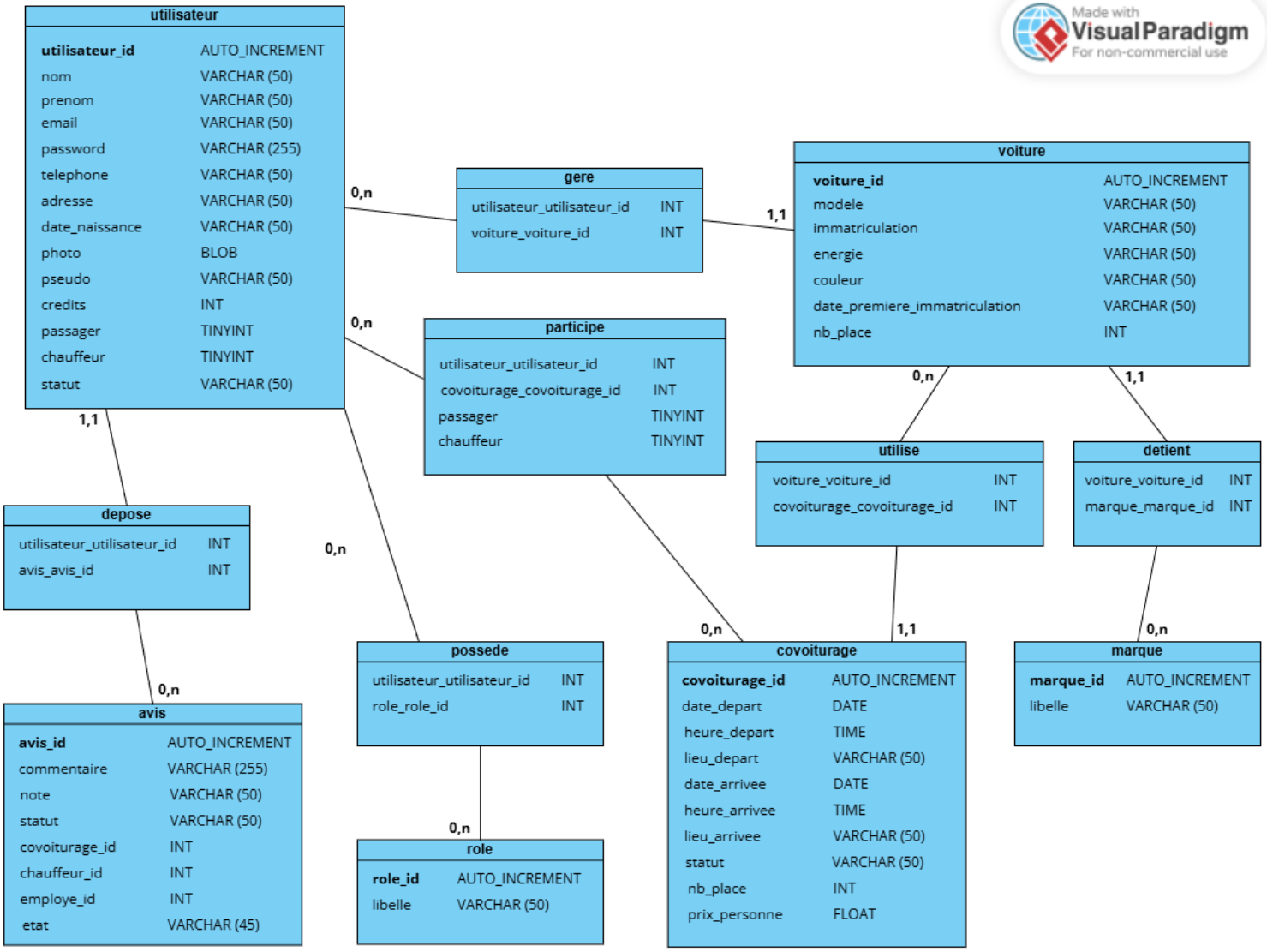
- IDE : Visual Studio Code
- HTML5 (développement natif)
- CSS3 (développement natif)
- JavaScript (sans framework, avec la bibliothèque Chart.js)
- PHP pour le développement back-end, avec accès à la base de données via PDO (requêtes SQL préparées)
- MySQL Workbench pour ma base de données relationnelles
- MongoDB Compass pour ma base de données NoSQL
- Serveur local et construction d'images Docker pour le déploiement et l'exécution de l'application en environnement local via Docker Desktop
- Serveur VPS pour le déploiement et l'exécution de l'application en environnement de production, fourni par OVH.
- Système exploitation : Windows 11

2. Configuration de mon environnement de travail

Configuration de mon environnement de travail :

- J'ai utilisé **Visual Studio Code** comme environnement de développement pour sa simplicité d'utilisation et ses nombreuses extensions facilitant le développement et l'optimisation du code.
- J'ai créé un **compte GitHub** afin d'héberger mon application en ligne et de gérer le versionnement du projet, en respectant les bonnes pratiques de Git, avec une branche de développement et des branches dédiées aux différentes fonctionnalités.
- J'ai relié mon projet local à mon dépôt GitHub afin de pouvoir gérer et fusionner les différentes branches du projet.
- J'ai installé **Docker Desktop** afin d'exécuter mon application en local à l'aide de conteneurs Docker intégrant un serveur Apache.
- Pour la gestion de la **base de données relationnelle**, j'ai utilisé **MySQL Workbench**, ce qui m'a permis de créer les tables nécessaires et d'y insérer les données en m'appuyant sur mon diagramme de classes.
- Pour la gestion de la **base de données NoSQL**, j'ai utilisé **MongoDB Compass**, notamment pour la gestion des préférences des conducteurs.
- Enfin, pour le **déploiement de l'application**, j'ai créé des conteneurs Docker afin de la mettre en ligne sur un serveur VPS chez **OVHcloud**.

3. Diagramme de classe

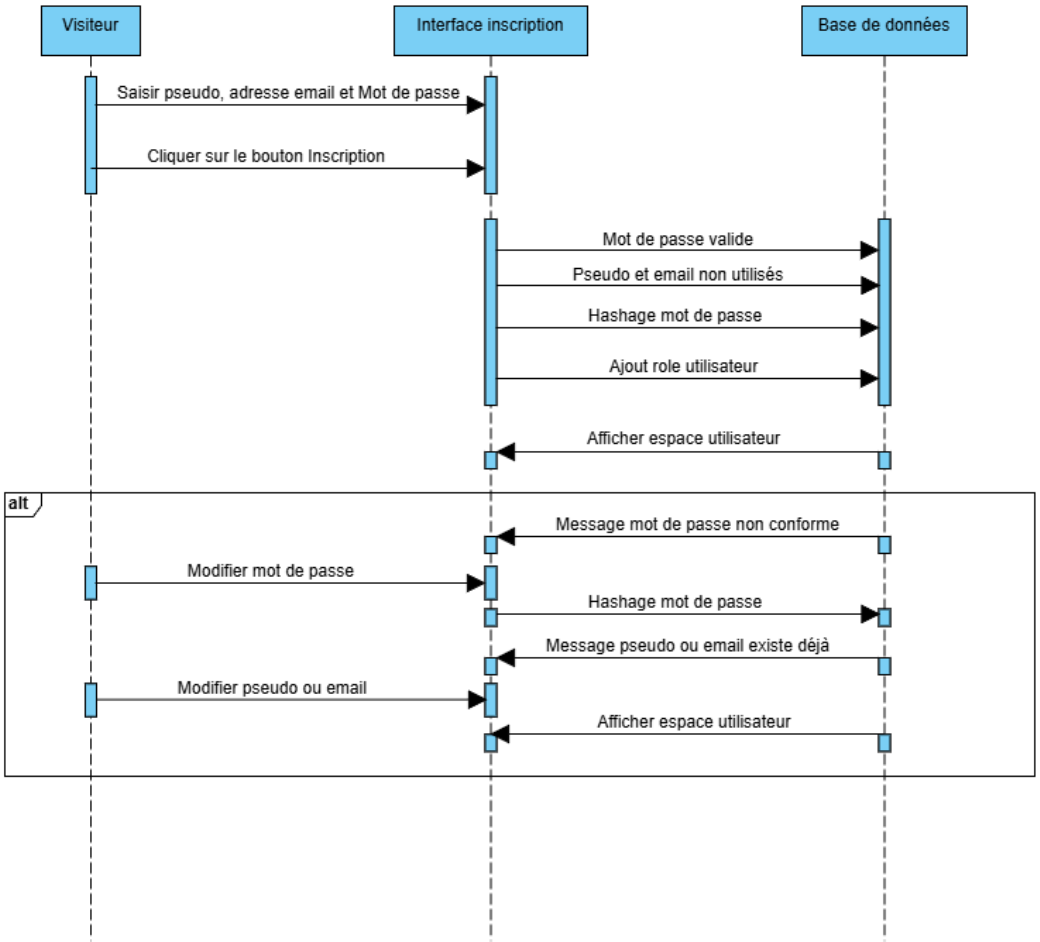


4. Diagramme d’utilisation et de séquence

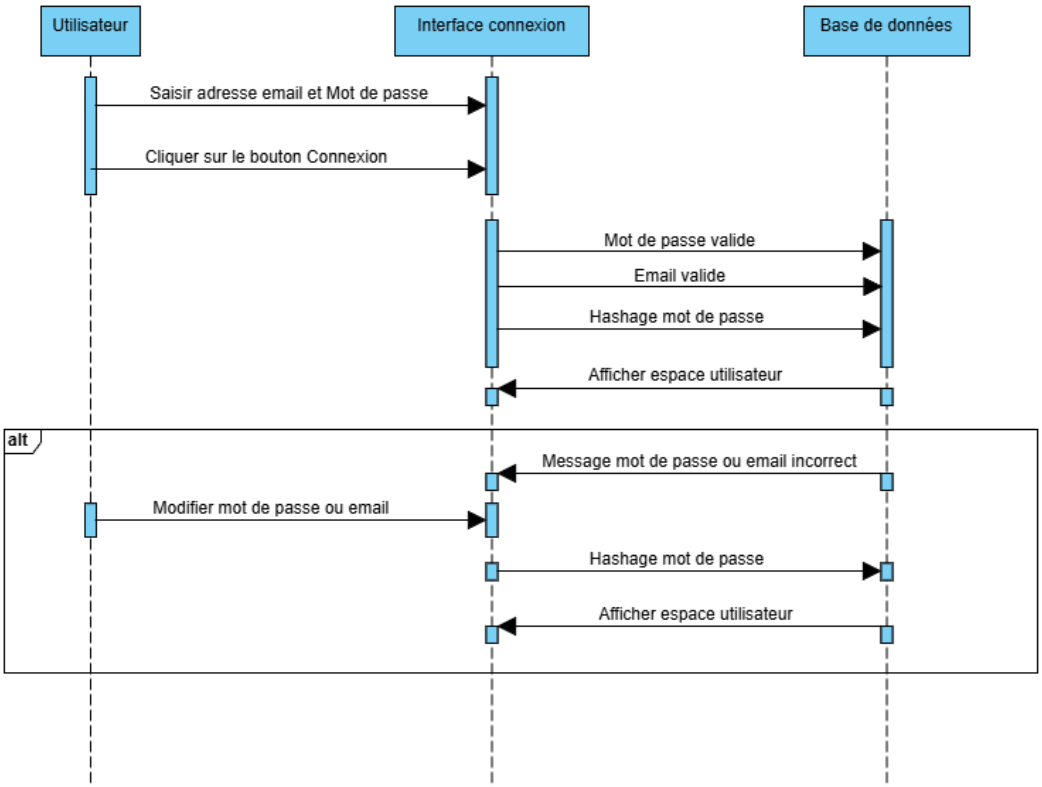
a. Diagramme d’utilisation :

b. Diagramme de séquence :

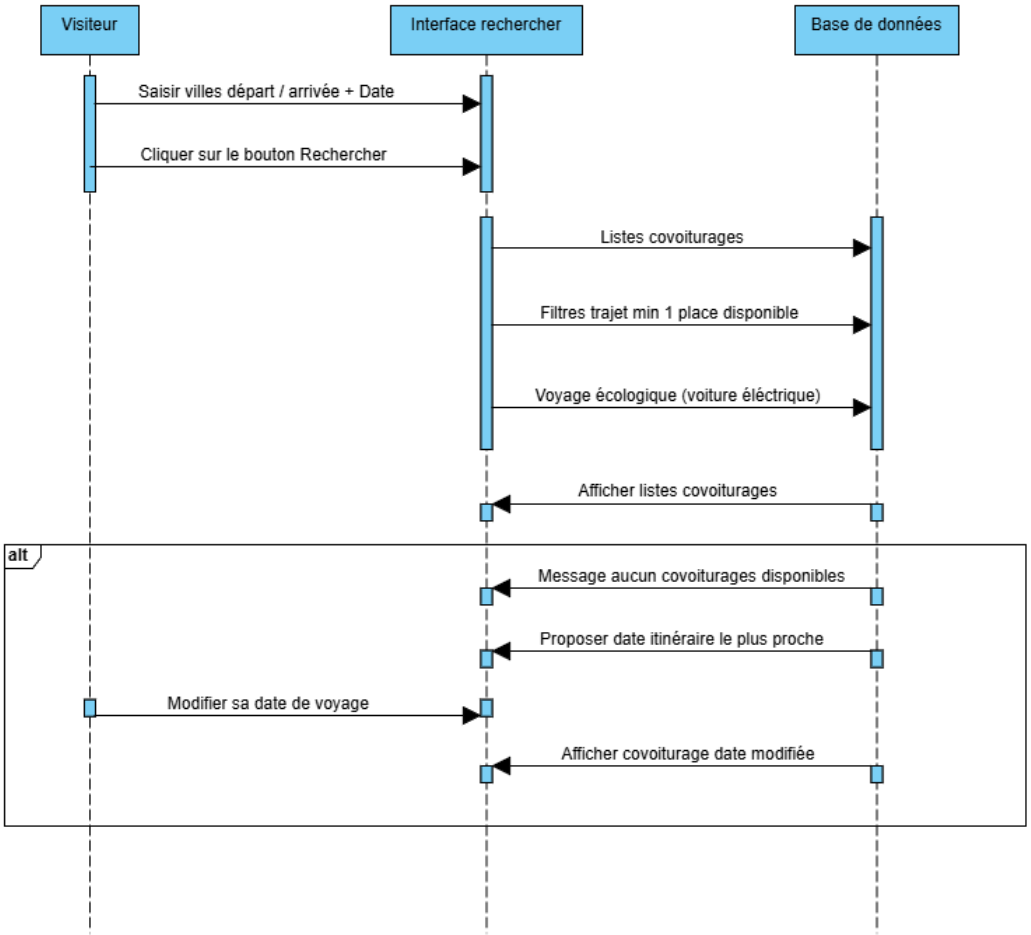
S’inscrire



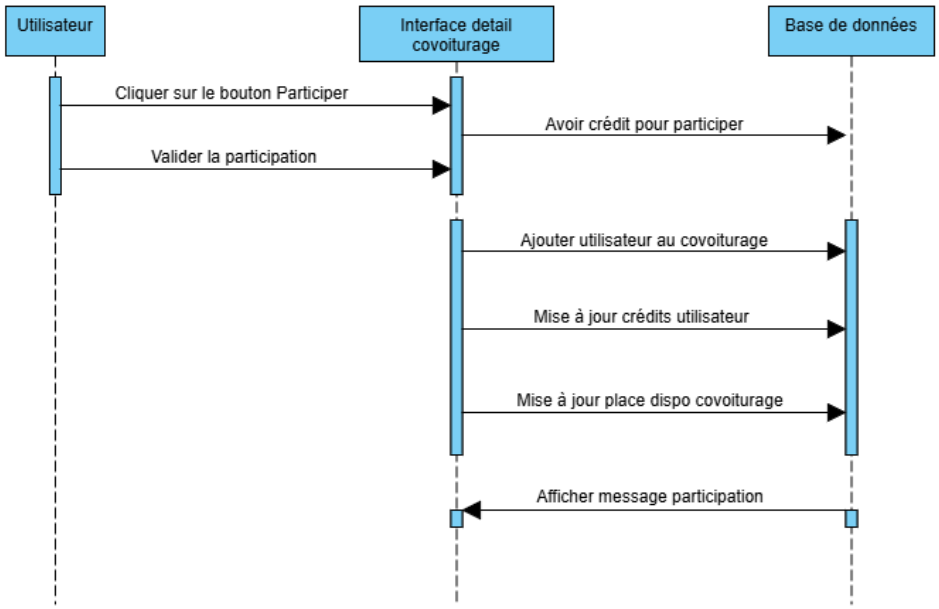
Se connecter



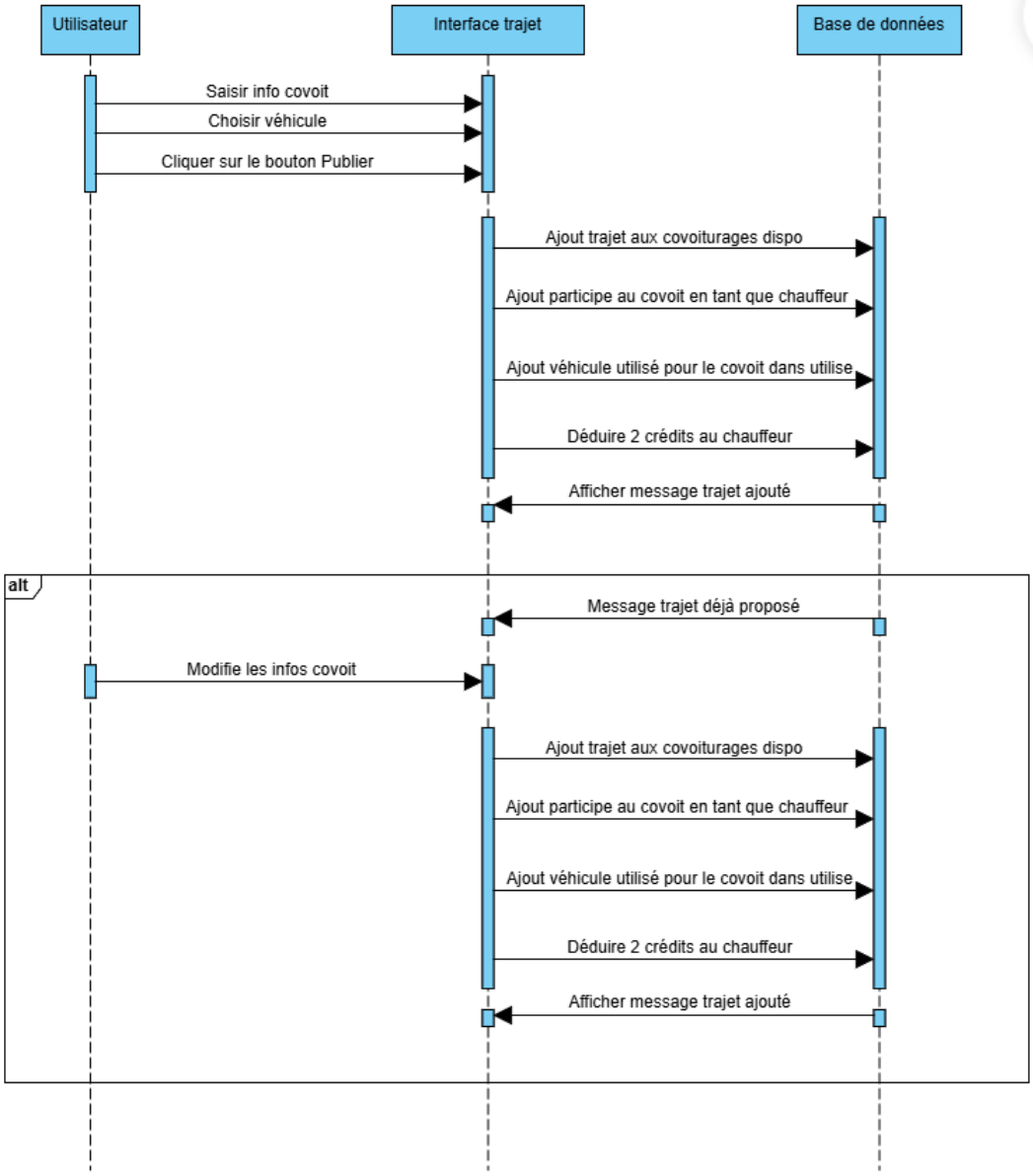
Afficher covoiturages



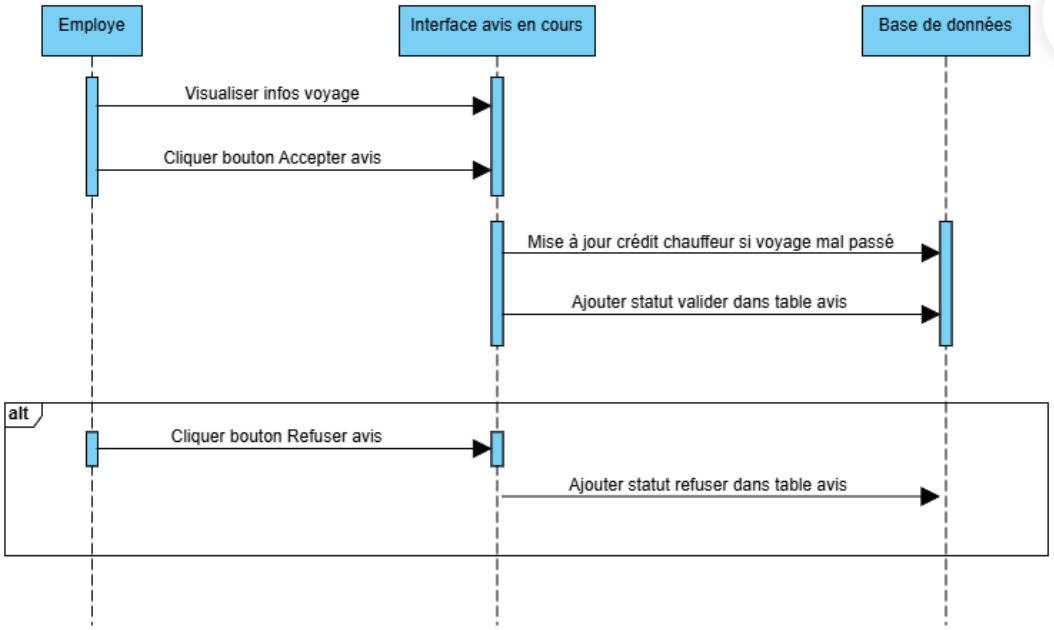
Participer covoiturage



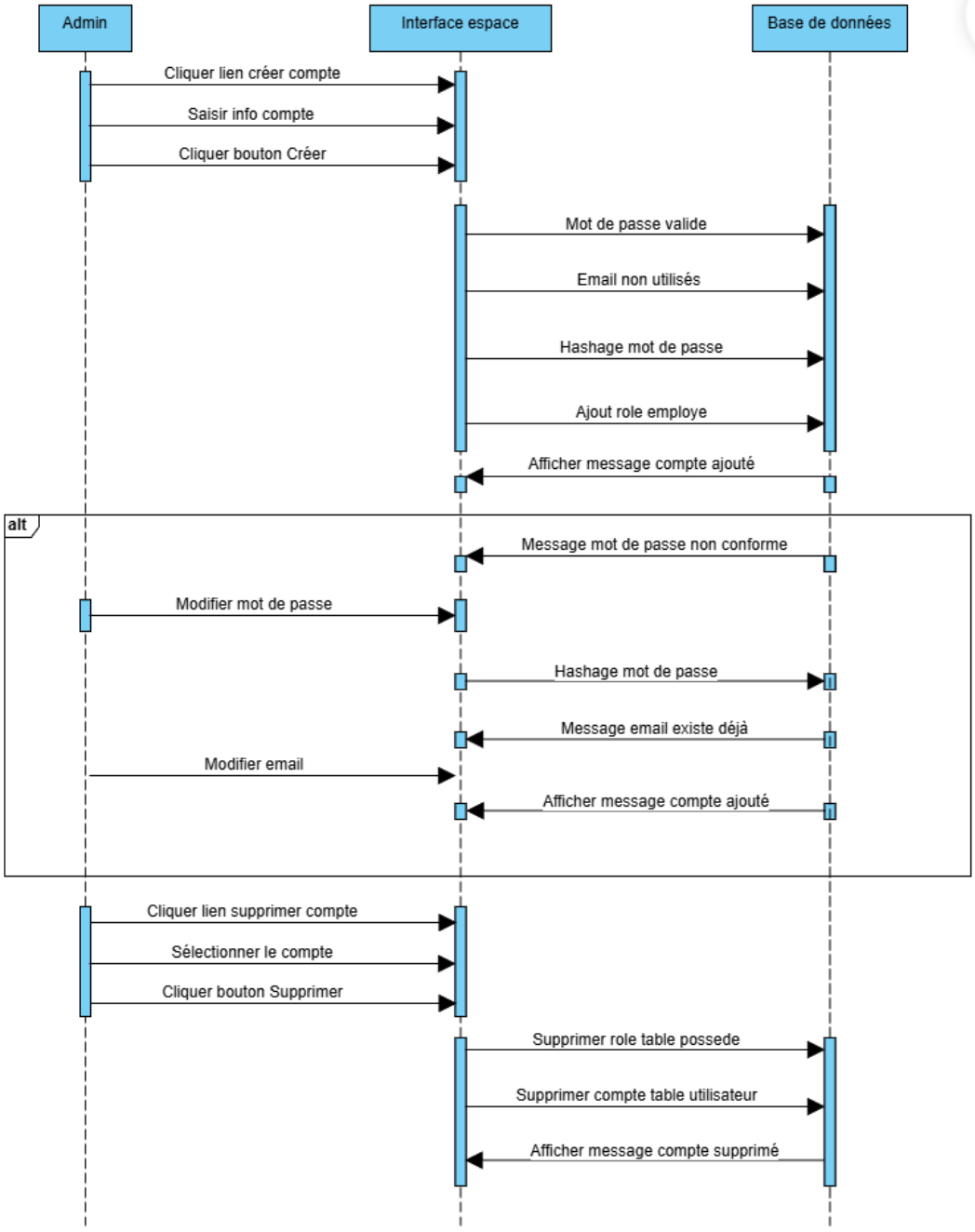
Proposer covoiturage



Valider / Refuser avis



Gérer compte



5. Déploiement de l'application

En local

Pour le déploiement de mon application en local, j'ai d'abord créé mes conteneurs en installant les dépendances via mon fichier YAML :

*Extrait du fichier **docker-compose.yml** :*

```yaml
services:
  # Service PHP
  php:
    build: .
    container_name: ecoride-php
    ports:
      - "8000:80"
    restart: unless-stopped
    depends_on:
      - db
      - mongo
    env_file:
      - ./.env
    volumes:
      - ./src:/var/www/html/src
      - ./public:/var/www/html/public

  # Database MySQL
  db:
    image: mysql:8.0
    container_name: ecoride-db
    restart: always
    ports:
      - "3307:3306"
    env_file:
      - ./.env
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      MYSQL_DATABASE: ${DB_NAME}
      MYSQL_USER: ${DB_USER}
      MYSQL_PASSWORD: ${DB_PASS}
      TZ: Europe/Paris
    volumes:
      - db_data:/var/lib/mysql
      - ./database/ecoride.sql:/docker-entrypoint-initdb.d/01-ecoride.sql:ro

```

Ce fichier définit les principaux services nécessaires à l'application :

- Le service **PHP** exécute l'application avec Apache et mappe les dossiers src et public dans le conteneur.
- Le service **MySQL** configure la base de données avec ses identifiants, le port exposé et le script d'initialisation.
- Les conteneurs sont liés via **depends_on** pour garantir le démarrage dans le bon ordre.

Cet extrait montre la configuration principale permettant le déploiement local de l'application via Docker.

Ensuite, j'ai construit l'image à partir de mon fichier Dockerfile :

*Fichier **dockerfile** :*

```
# Image de base PHP avec Apache
FROM php:8.2-apache

# Dépendances système
RUN apt-get update && apt-get install -y \
    libicu-dev \
    libssl-dev \
    unzip \
    git \
    curl \
    && docker-php-ext-install intl \
    && pecl install mongodb \
    && docker-php-ext-enable mongodb

# Installer PDO MySQL
RUN docker-php-ext-install pdo pdo_mysql

# Installer Composer
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

# Activer mod_rewrite
RUN a2enmod rewrite

# Copier le projet
COPY . /var/www/html/

# Installer les dépendances PHP
WORKDIR /var/www/html
RUN composer install --no-dev --optimize-autoloader

# Apache : DocumentRoot = public
RUN sed -i 's|/var/www/html|/var/www/html/public|g' \
    /etc/apache2/sites-available/000-default.conf \
    /etc/apache2/apache2.conf

# Autoriser .htaccess dans public
RUN printf '<Directory /var/www/html/public>\n\
    AllowOverride All\n\
    Require all granted\n\
</Directory>\n' >> /etc/apache2/apache2.conf

# Permissions Apache
RUN chown -R www-data:www-data /var/www/html \
    && chmod -R 755 /var/www/html
```

Ce **Dockerfile** définit l'environnement nécessaire à l'exécution de l'application PHP :

- **Image PHP + Apache** : le conteneur exécute le code PHP avec un serveur Apache prêt à l'emploi.
- **Extensions et dépendances** : les extensions PHP nécessaires (**intl**, **pdo**, **pdo_mysql**, **mongodb**) sont installées, ainsi que les outils système (**git**, **curl**, **unzip**) pour gérer le projet et ses dépendances.

- **Composer** : le gestionnaire de dépendances PHP est installé depuis l'image officielle pour gérer automatiquement les bibliothèques utilisées par l'application.
- **Configuration Apache** : le **DocumentRoot** est défini sur le dossier public et **.htaccess** est autorisé pour permettre la réécriture d'URL avec **mod_rewrite**.
- **Copie du projet et installation des dépendances** : le code source est copié dans le conteneur et les dépendances PHP sont installées **avec composer install --no-dev --optimize-autoloader**.
- **Permissions** : le conteneur définit les droits d'accès corrects pour l'utilisateur Apache (**www-data**) afin d'éviter les problèmes de lecture ou d'écriture.

Cet extrait montre la configuration principale permettant le **déploiement local de l'application PHP via Docker**, avec un environnement isolé, reproductible et prêt à l'exécution.

Ensuite pour lancer mon application et accéder à mes bases de données, j'ai utilisé **VS Code** avec le terminal intégré. Voici les étapes réalisées :

1. **Construction de l'image Docker**
   J'ai exécuté le build de l'image Docker de mon projet afin de créer un environnement isolé pour l'application et ses services associés.
   Commandes utilisées :

```bash
docker-compose build
docker-compose up -d
```

- **docker-compose build** : permet de construire toutes les images nécessaires à partir du fichier docker-compose.yml.
- **docker-compose up -d** : démarre les conteneurs en arrière-plan, ce qui permet de lancer l'application et les services associés sans bloquer le terminal.

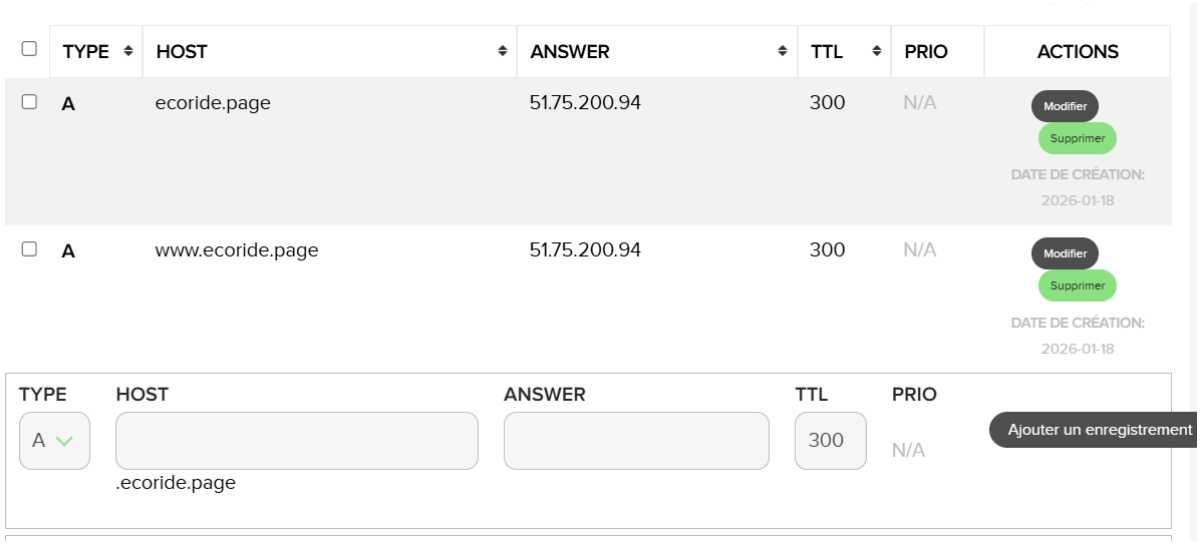
2. **Démarrage des conteneurs**
   Une fois l'image construite, j'ai lancé les conteneurs Docker, permettant de :
   - Exécuter l'application **PHP** et la visualiser dans le navigateur.
   - Accéder à la base de données **MySQL** via **Adminer**, ce qui m'a permis de consulter et gérer les données facilement.
   - Accéder à la base de données **MongoDB** via **Mongo Express** pour gérer les données NoSQL.

Grâce à cette configuration, j’ai pu tester l’application et vérifier le fonctionnement de toutes les bases de données dans un environnement stable et contrôlé.

En production

Pour le déploiement de l'application EcoRide, j'ai dans un premier temps obtenu un nom de domaine gratuit via le site *name.com*, grâce au **GitHub Student Developer Pack**. Ce nom de domaine permet d'accéder à l'application de manière publique et professionnelle.

Ensuite, j'ai **associé ce nom de domaine à l'adresse IP de mon serveur VPS** en ajoutant deux enregistrements de type **A** dans les paramètres DNS de *name.com*, comme illustré ci-dessous :



Exemple des enregistrements A pour ecoride.page et www.ecoride.page pointant vers l'IP de mon VPS (51.75.200.94).

1. **Installation et configuration du serveur Nginx :**

Sur le VPS, j'ai installé **Nginx** pour gérer les requêtes web vers mon application. Les commandes utilisées sont les suivantes :

```bash
sudo apt update
sudo apt install -y nginx
```

Ensuite, j'ai créé un fichier de configuration spécifique pour le site EcoRide :

```bash
sudo nano /etc/nginx/sites-available/ecoride
```

Le fichier contient les directives suivantes :

- Proxy vers l'application Docker sur le port 8000.
- Gestion des headers pour conserver l'IP réelle du client.

- Redirection automatique HTTP → HTTPS.
- Activation SSL via Certbot pour sécuriser la connexion.

```
server {

    server_name ecoride.page www.ecoride.page;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/ecoride.page/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/ecoride.page/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}

server {
    if ($host = www.ecoride.page) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = ecoride.page) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    listen [::]:80;

    server_name ecoride.page www.ecoride.page;
    return 404; # managed by Certbot




}
```

2. Activation et test de la configuration

Pour activer le site :

bash Copier le code

```
sudo ln -s /etc/nginx/sites-available/ecoride /etc/nginx/sites-enabled/
```

Puis relancer Nginx pour appliquer les changements :

bash Copier le code

```
sudo nginx -t
sudo systemctl reload nginx
```

3. **Résultat de la redirection**

- La requête sur http://51.75.200.94 redirige automatiquement vers le nom de domaine.
- Docker continue de tourner sans conflit sur le port 8000.
- Ports ouverts :
    - **Nginx → 80**
    - **Apache/Docker → 8000**

4. **Installation d'un certificat SSL**

Pour sécuriser la connexion HTTPS, j'ai installé Certbot :

```bash
sudo apt update
sudo apt install -y certbot python3-certbot-nginx
```

Puis, génération du certificat pour les deux enregistrements A du domaine :

```bash
sudo certbot --nginx -d ecoride.page -d www.ecoride.page
```

5. **Application fonctionnelle et sécurisée**

L'application EcoRide est maintenant :

- Accessible via HTTPS.
- Sécurisée grâce au certificat SSL et à la redirection HTTP → HTTPS.
- Hébergée sur un VPS distinct pour séparer **environnement de développement local** et **production**.
- Testée pour vérifier stabilité, accessibilité et fonctionnement des fonctionnalités principales.