Compte-rendu d'exercice de développement

Département de la Marne

16 janvier 2025

VERPOORTE William

Sommaire

1 Objectif

L'objectif de ce projet est de développer une application web qui repose sur une architecture moderne combinant un backend en Symfony avec API Platform et un frontend en React. Cette application doit permettre au frontend de consommer les ressources fournies par l'API backend.

L'application consiste en une interface météo simple avec les fonctionnalités suivantes :

- **Affichage de la météo actuelle pour Reims** : Accessible à tous les utilisateurs dès la page d'accueil.
- **Inscription et connexion des utilisateurs** :
  - Les utilisateurs authentifiés peuvent ajouter des adresses personnalisées.
  - Ils peuvent consulter les prévisions météo sur 7 jours pour ces adresses.
- **Utilisation d'API externes** :
  - `OpenMeteo` pour récupérer les données météorologiques.
  - `adresse.data.gouv.fr` pour trouver et valider les adresses saisies.
- **Gestion administrateur** :
  - Un compte administrateur est préconfiguré dans la base de données.
  - L'administrateur dispose d'un tableau de bord lui permettant de :
    - Ajouter, modifier ou supprimer un utilisateur.
    - Ajouter ou supprimer une adresse associée à un utilisateur.
  - En dehors de ces fonctions spécifiques, l'administrateur possède les mêmes fonctionnalités qu'un utilisateur standard.

2 Use Case

Le diagramme ci-dessous illustre les cas d'utilisation de l'application, mettant en évidence les actions accessibles à chaque type d'utilisateur en fonction de son rôle (utilisateur standard ou administrateur).

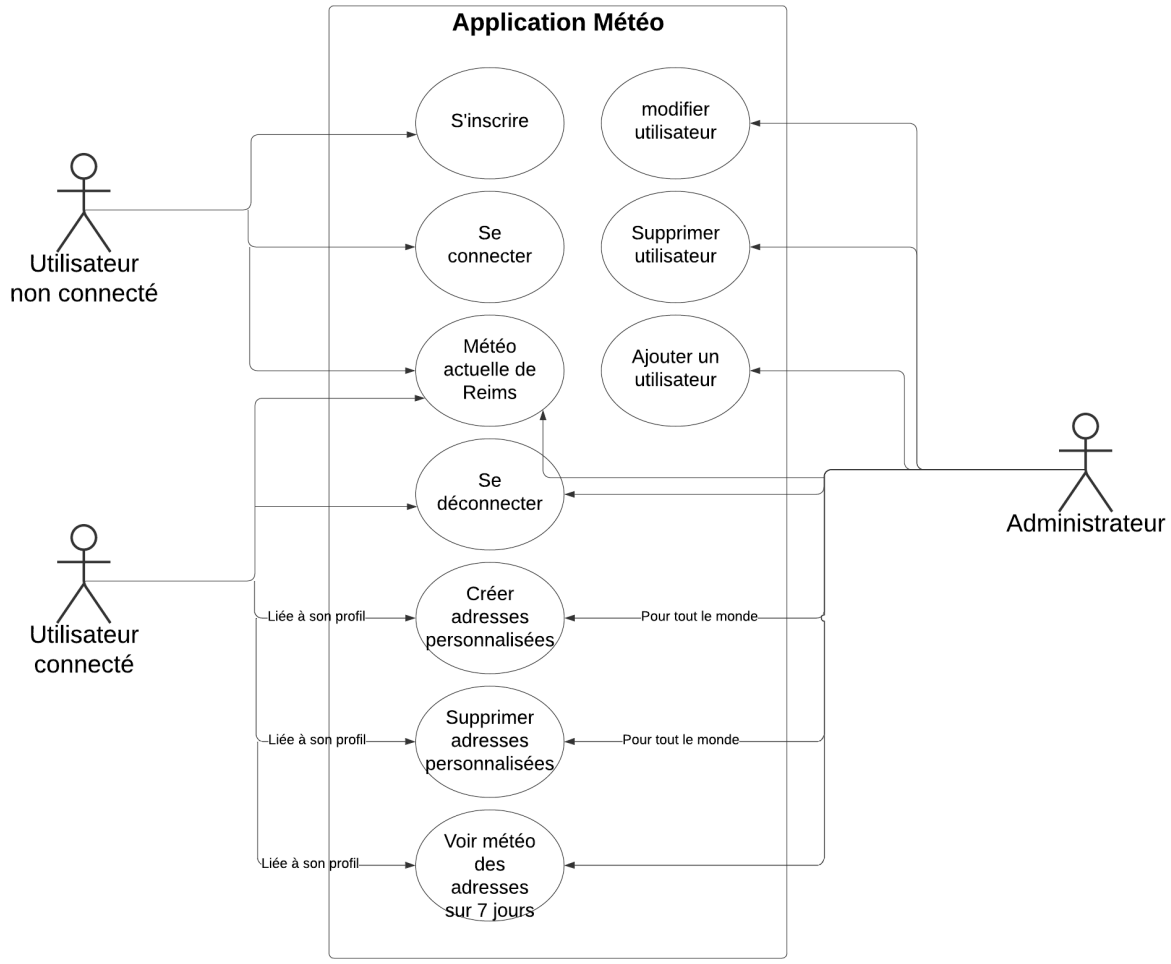

Figure 1: Diagramme des cas d'utilisation de l'application

3 MCD

Le diagramme ci-dessous présente le Modèle Conceptuel de Données (MCD) de l'application. Il décrit les relations entre les différentes entités utilisées dans le système.

Description des entités et relations

- **Weather** : Cette entité est utilisée pour stocker les informations météorologiques affichées par défaut dans l'application, comme la météo actuelle de Reims.
- **Utilisateur - Adresse** :
  - Un utilisateur est associé à une liste d'adresses. La relation est de type $0, n$, indiquant qu'un utilisateur peut ne pas avoir d'adresse ou en avoir plusieurs.

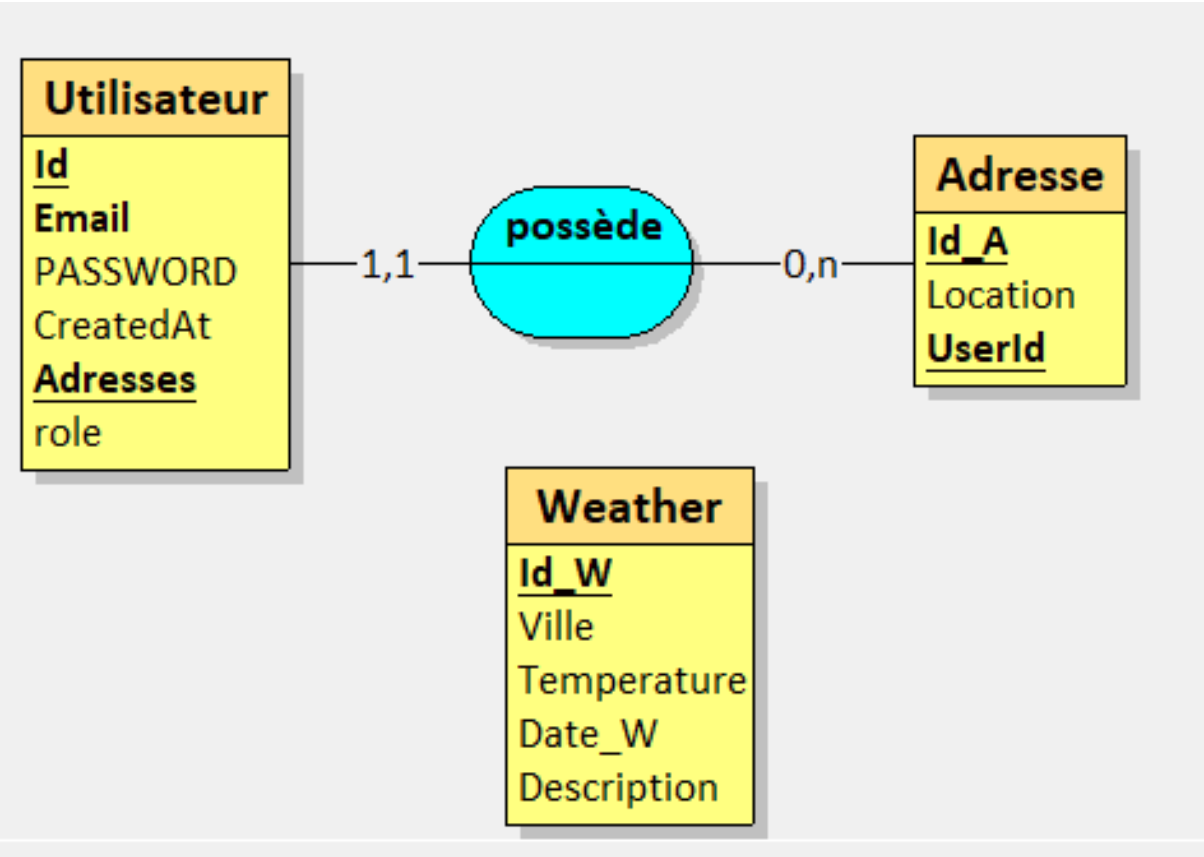

Figure 2: Modèle Conceptuel de Données (MCD)

– Une adresse est associée à un seul utilisateur. La relation est de type $1,1$, garantissant qu'une adresse appartient exclusivement à un utilisateur.

4 Documentation

4.1 Installation de l'application

Prérequis :

- PHP 8.2
- Composer installé
- Node.js et npm installés
- MySQL installé et configuré

Étapes d'installation :

1. Clonez le projet ou extrayez le package :

```
git clone https://github.com/WilliamVPT/appli-meteo
cd appli_meteo
```

ou dézippez l'archive.

2. Installez les dépendances backend avec Composer :

```
backend/composer install
```

3. Installez les dépendances frontend avec npm :

```
frontend/npm install
```

4. Configurez le fichier .env du backend :

```
DATABASE_URL="mysql://[user]:[password]@[host]:3306/
[database_name]"
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/
private.pem
JWT_PASSPHRASE=your_passphrase_here
```

5. Importez la base de données :

```
mysql -u [utilisateur] -p [nom_base] < dump.sql
```

6. Lancez le serveur Symfony :

```
backend/symfony serve
```

7. Compilez et lancez le frontend :

```
frontend/npm start
```

Notes :

- Remplacez `[user]`, `[password]`, `[host]` et `[database_name]` par vos informations MySQL.
- Assurez-vous que les clés JWT (`private.pem` et `public.pem`) sont générées et placées dans le dossier `config/jwt`.

4.2 Connexion

L'administrateur par défaut peut se connecter avec les identifiants suivants :

- **Email** : `admin@admin`
- **Mot de passe** : `admin`

Pour tous les utilisateurs déjà présents dans la base de données, les identifiants suivent la structure suivante :

- **Email** : `[nom]@[nom]` (parfois suivi de `.com`)
- **Mot de passe** : `[nom]`

4.3 Technologies utilisées

Comme mentionné précédemment, l'application utilise React pour le frontend et Symfony pour le backend, avec un environnement de développement configuré sur Visual Studio Code. Pour la gestion de la base de données, j'ai utilisé MySQL, tandis que Composer et npm ont été employés pour la gestion des dépendances. Mon expérience préalable avec npm a facilité cette partie du projet.

Concernant la gestion de l'état côté frontend, j'ai choisi d'utiliser `localStorage` au lieu de React Context, comme suggéré initialement. Cette solution offre une meilleure persistance des données lors de la navigation ou du rechargement de la page, ce qui améliore selon moi l'expérience utilisateur en la rendant plus fluide et cohérente.

Pour l'esthétique et l'ergonomie de l'application, j'ai utilisé Bootstrap, ce qui m'a permis de concevoir une interface utilisateur agréable et fonctionnelle. Vers la fin du projet, j'ai commencé à m'intéresser à Flowbite pour des composants plus personnalisés, mais par manque de temps, je n'ai pas pu approfondir cette partie.

Les échanges de requêtes et réponses avec le backend sont gérés à l'aide d'`axios`. Pour simplifier son utilisation dans les différentes pages, j'ai configuré une instance centralisée d'`axios`, que vous pouvez retrouver dans le fichier `axiosConfig.js`.

Côté backend, j'ai utilisé le package `jwt` pour gérer les informations des utilisateurs connectés et permettre au frontend d'y accéder. Deux fichiers, contenant une clé privée et une clé publique générées avec OpenSSL, sont présents dans le dossier `config/jwt`. Ces clés fonctionnent en lien avec les configurations de `jwt` définies dans le fichier `.env`.

4.4 Difficultés rencontrées

La principale difficulté de cet exercice a été d'apprendre à utiliser Symfony depuis zéro. Cela comprenait notamment la configuration des routes et leur bonne utilisation pour structurer l'API de manière optimale.

Un autre défi a été de déterminer la meilleure architecture de projet pour combiner efficacement Symfony et React. Ce processus m'a demandé du temps et des ajustements pour assurer une intégration correcte entre le frontend et le backend.

Enfin, une part significative du temps a été consacrée au débogage des erreurs de serveur lors des requêtes et des réponses entre le backend et le frontend. Ces erreurs impactaient l'échange correct des informations et la mise à jour de la base de données. Cela m'a permis de mieux comprendre les interactions entre les deux parties de l'application et de garantir un fonctionnement fluide.

5 Conclusion

Ce projet m'a offert l'opportunité de découvrir le développement avec Symfony tout en approfondissant mes connaissances en React et PHP. Il m'a également permis de revisiter et consolider certaines notions de développement que j'avais pu oublier par le passé, enrichissant ainsi mes compétences globales en programmation.

Je vous remercie pour le temps consacré à la lecture de ce document et pour l'attention portée à ma candidature. J'espère avoir l'opportunité de travailler avec votre équipe prochainement.