

Réflexions technologiques

Pour mon application de covoiturage, j'ai choisi de travailler avec PHP et Symfony pour le back-end :

- J'ai fait le choix d'utiliser PHP, car c'est un langage très répandu, et très utilisé pour le développement web. Il est particulièrement adapté pour mon application, qui a besoin d'interagir énormément avec les bases de données, et d'effectuer des traitements côté serveur. PHP possède également un large choix de bibliothèque (comme PHPMailer) et de framework (comme Symfony ou Laravel). C'est un langage avec une grande communauté et de ressources en ligne, et en plus il est simple à apprendre.

M'étant orienté vers PHP, j'ai donc choisi un framework associé : Symfony.

- Symfony est un framework plus que pratique, il est très puissant et inclus beaucoup de « magie » comme l'authentification. Il possède également des composants de sécurité inclus, ce qui m'a été d'une grande aide. Symfony permet de bien structurer son application, avec les différents dossiers, et séparation de fichier, comme il est normal de faire en programmation, ce qui m'a permis d'utiliser les bonnes pratiques facilement, même en étant débutante.

Le serveur intégré de Symfony est également d'une grande aide pour aider à debugger, avec toutes les étapes d'indiquées, son profiler et sa gestion des erreurs. Etant plus que novice dans le débogage, cela m'a été d'une grande aide.

Pour mes bases de données, j'ai choisi MySQL en relationnelle, MongoDB en non relationnelle :

- MySQL est une base de données relationnelle, ayant pas mal de table et de relation dans l'application web, comme les utilisateurs, les covoiturages, les voitures, les rôles métier, etc., il me fallait quelque chose de haute performance. Etant la 1^{ère} fois que j'utilisais une base de données, j'en ai choisi une ayant une forte compatibilité avec PHP.
- MongoDB est utilisé ici pour gérer les données de préférences des utilisateurs 'chauffeur'. Il permet de stocker les données sous forme de document, et est un logiciel très souple. Ici, j'utilise MongoDB Atlas, pour une version en ligne et une liaison plus simple lors du déploiement.

Travaillant avec PHP, il était judicieux de travailler également avec Doctrine, qui est un ORM (Object-Relational Mapping) pour PHP. Il m'a permis de travailler avec des objets au lieu de réaliser des requêtes SQL. Pour cela, j'ai créé des entités représentant les tables de ma base de données, avec des propriétés qui correspondent aux colonnes de mes tables. Pour lier représenter les relations entre mes tables, j'ai dû réaliser un mapping. Doctrine permet aussi de gérer le CRUD (Create, Read, Update, Delete) plus facilement, en utilisant l'EntityManager, persist, flush, remove.

Pour ce qui est de la partie front-end de l'application, j'ai choisi les langages HTML, CSS et JavaScript :

- HTML est un langage utilisé pour structurer sa page web. Je l'ai choisi car c'est le langage numéro un pour le front-end, pour voir du texte s'affiche sur un navigateur. Également, PHP peut générer du HTML, cela me paraissait donc logique de l'utiliser.
- Le CSS sert à définir des règles de style pour les différents éléments HTML. Il permet de réaliser des mises en pages originales quand on le maîtrise bien. Il permet également en utilisant les médias queries, de s'adapter à différentes tailles d'écran. Tout comme le HTML, il s'agit du langage numéro un, pour le style et la mise en page, il est donc logique de l'utiliser dans mon application web de covoiturage.
- JavaScript est un langage qui est utilisé côté front-end et côté back-end. Ici, je l'utilise côté front-end, en réalisant des fenêtres pop-up, qui rendent l'application plus interactive et dynamique. Pour moi, il est également le langage numéro un pour rendre une page dynamique, il était donc logique de l'utiliser.

Pour le côté front-end, j'utilise également, le moteur de template Twig. Je l'ai découvert en même temps que Symfony car il est utilisé par défaut dans Symfony. Je l'ai trouvé très pratique pour structurer ma page avec le HTML, et également me permettre de réaliser des blocs réutilisables dans plusieurs pages, comme le header ou le footer. Également, Twig permet d'insérer des variables dynamiques venant du back-end, comme ici dans l'application des données venant de la base de données (pseudo utilisateur, photo de profil, liste de véhicule, historique de covoiturage). Il m'a permis aussi de réaliser des conditions sans se mélanger avec le code PHP, comme n'afficher un lien ou bouton que si l'utilisateur a le rôle métier adéquate. Twig m'a aussi surtout permis de rendre mes formulaires plus dynamiques, et de leur amener de la sécurité (comme les injections XSS).

Configuration de mon environnement de travail

Pour mon environnement de travail, je possédais déjà Visual Studio Code sur mon PC. Avec la formation Studi, j'ai installé PHP Storm, comme indiqué dans le cours, mais je ne m'en sers actuellement pas. Je suis beaucoup plus à l'aise avec VS Code.

Comme serveur local, j'utilise WAMP et le serveur de Symfony. J'ai fait le choix de travailler sur les deux en parallèle car le serveur de Symfony apporte une grande aide au niveau back-end, pour tout ce qui est débogage, avec l'indication des erreurs, des mappings. Et le serveur de WAMP est l'exemple de ce que je vais avoir une fois mon application déployée. Il me permet d'avoir le bon rendu visuel et d'ajuster en conséquence (comme des images qui ne s'affichent pas).

- **Pour WAMP** : Comprenant PHP, Apache et MySQL

Tout d'abord il faut aller sur le site de <https://www.wampserver.com/>, télécharger WAMP et se laisser guider par la guide d'installation.

Une fois WAMP installé, il a fallu mettre PHP dans mes variables d'environnement. Pour cela, dans ma barre de recherche de Windows, j'ai tapé 'SystemPropertiesAdvanced.exe', j'ai cliqué sur exécuter. Sur la fenêtre qui s'ouvre, il faut cliquer sur 'variables d'environnement', sous 'variable système' j'ai sélectionné path, et cliquer sur 'modifier'. Ensuite j'ai cliqué sur 'nouveau' puis 'parcourir' et j'ai recherché le chemin de mon fichier PHP, dans mon dossier WAMP/bin. Après cela, j'ai cliqué sur 'OK', jusqu'à la disparition de la fenêtre.

Pour Apache, il m'a fallu aller dans le fichier WAMP/bin/Apache/conf/httpd.conf. Après cela, j'ai suivi le cours de Studi m'indiquant comment configurer Apache

Pour permettre à Apache de lire vos fichiers PHP, vous devez faire un peu de configuration !

- Accédez au fichier « C:\Apache24\conf\httpd.conf ».
- Ajoutez le code suivant tout en bas du fichier :

```
1 PHPIniDir "C:/php"
2 LoadModule php_module "C:/php/php8apache2_4.dll"
3 AddType application/x-httpd-php .php
```

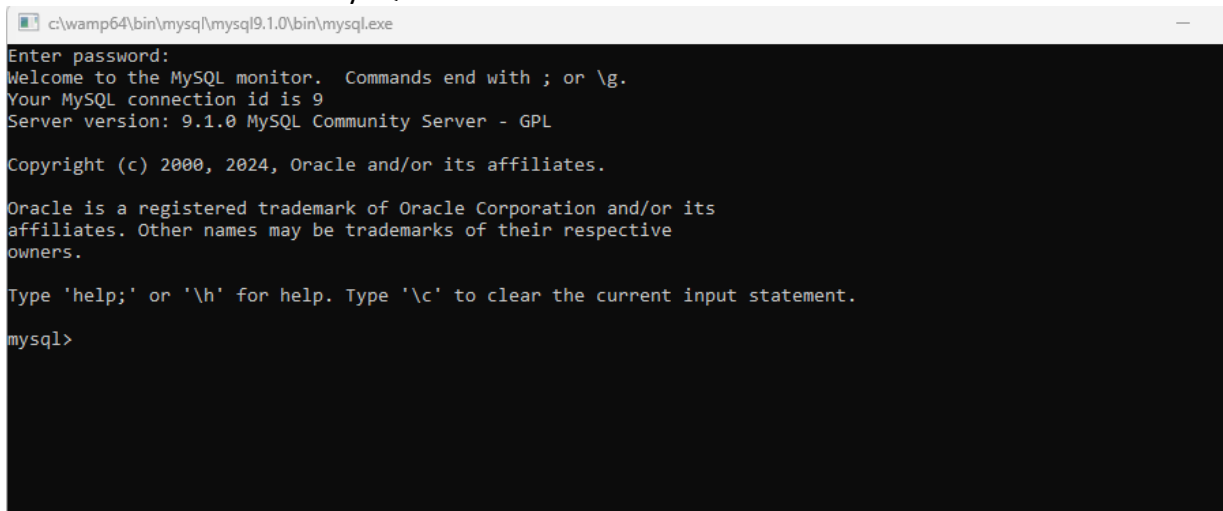
Également :

- Cherchez la ligne <IfModule dir_module> et ajoutez le mot « *index.php* » entre « *Directoryindex* » et « *index.html* ».
- Sauvegardez le fichier et redémarrez Apache2.

```
1 <IfModule dir_module>
2     DirectoryIndex index.php index.html
3 </IfModule>
```

Extrait du cours « Préparer l'environnement et la création d'un projet Symfony » de Sudi

L'installation de WAMP m'a permis de pouvoir travailler avec MySQL, pour créer ma base de données relationnelle. Pour cela, en cliquant sur le bouton de WAMP, je peux sélectionner l'invit de commande de MySQL.

A screenshot of a Windows command prompt window titled "c:\wamp64\bin\mysql\mysql9.1.0\bin\mysql.exe". The terminal shows the MySQL login sequence: "Enter password:", "Welcome to the MySQL monitor. Commands end with ; or \g.", "Your MySQL connection id is 9", "Server version: 9.1.0 MySQL Community Server - GPL", "Copyright (c) 2000, 2024, Oracle and/or its affiliates.", "Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.", "Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.", and finally the prompt "mysql>".

```
c:\wamp64\bin\mysql\mysql9.1.0\bin\mysql.exe
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

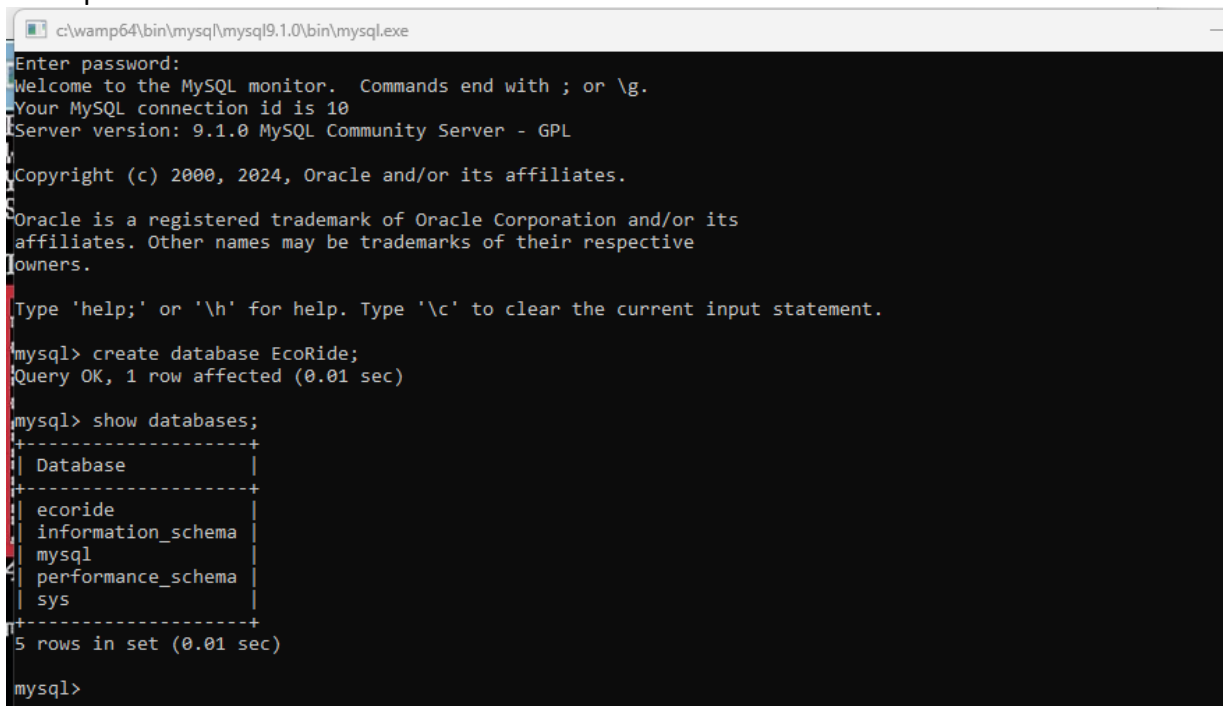
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Capture d'écran du terminal de MySQL

Une fois ouvert, j'ai rentré mes différentes commandes SQL pour créer ma base de données, ainsi que mes tables et leurs colonnes associées.

A screenshot of the same MySQL command prompt window. It shows the execution of two SQL commands. First, "mysql> create database EcoRide;" followed by "Query OK, 1 row affected (0.01 sec)". Then, "mysql> show databases;" followed by a table listing the databases: "Database", "ecoride", "information_schema", "mysql", "performance_schema", and "sys". The output ends with "5 rows in set (0.01 sec)" and the prompt "mysql>".

```
c:\wamp64\bin\mysql\mysql9.1.0\bin\mysql.exe
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database EcoRide;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| ecoride  |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.01 sec)

mysql>
```

Capture d'écran du terminal MySQL après la création de la base de données EcoRide

Une fois la base de données créée entièrement en SQL, j'ai utilisé PhpMyAdmin, compris dans WAMP. Pour visualiser plus facilement ma base de données lors des essais sur l'application, comme lors d'une inscription, pour vérifier que les données étaient bien présentes dans ma base après validation du formulaire.

- **Pour le serveur de Symfony :**

Pour installer le serveur de Symfony, j'ai d'abord installé Symfony CLI et Composer (voir plus bas, l'installation de Composer). Pour installer Symfony CLI, j'ai tapé la commande suivante dans mon invit de commande

```
composer global require symfony/cli
```

Ensuite le serveur était disponible, il me suffit de le lancer

```
PS C:\Users\samvh\Desktop\formation\evals\ECF ete 25\projet\dossier projet\MonProjet-EcoRide> .\symfony server:start
```

Pour l'arrêter, j'utilise la commande `.\symfony server:stop`

Composer, va me permettre d'installer le serveur de Symfony et également de créer mon projet Symfony. Il s'agit d'un gestionnaire de dépendances PHP, et me permettre d'installer celles dont j'ai besoin pour le projet. Pour cela, il faut vérifier que PHP est bien dans mes variables d'environnement et télécharger la dernière version de Composer. Ensuite, il suffit de lancer l'assistant d'installation et de se laisser guider par les instructions. J'ai également, ajouté Composer à mes variables d'environnement, comme avec PHP.

Pour le framework **Symfony**, je n'ai pas fait d'installation particulière, à l'aide de Composer j'ai créé le projet directement. Dans mon invit de commande, je me suis mise dans le dossier où j'ai choisi de créer mon projet. Puis, j'ai tapé

```
composer create-project symfony/skeleton MonProjet-EcoRide
```

Si besoin j'ai rajouté des dépendances PHP, comme Doctrine ou Twig

```
composer require symfony/orm-pack
composer require symfony/twig-pack
```

Ici, j'ai fait le choix de mettre mon projet dans le dossier "C:\Users\samvh\Desktop\formation\evals\ECF ete 25\projet\dossier projet" et non dans le dossier 'www' de WAMP, car je voulais que tous ce qui concerne le projet soit au même endroit, et j'avais peur que cela n'interfère avec de futurs projets. Pour que WAMP prenne bien en compte le projet, j'ai configuré un VirtualHost, directement sur le serveur de WAMP. Aujourd'hui, je suis consciente, que si j'avais mis le projet dans le dossier 'www' cela n'aurait en rien perturbé les projets futurs, et je mettrais donc mes prochains projets dans le dossier 'www' sans aucun souci.

Une fois mon projet créée, c'était parti pour l'aventure, avec la création de mon premier controller, grâce au cours de Studi.

```
php bin/console make:controller
```

Avec ma première vue Twig, et la création manuelle de mon fichier CSS associé.

J'ai dû paramétrer ma base de donnée avec mon projet, en allant dans le fichier .env, et en remplissant correctement le DATABASE_URL, pour qu'il pointe vers ma base de données MySQL.

```
DATABASE_URL="mysql://root@127.0.0.1:3306/ecoride?serverVersion=16&charset=utf8"
```

En parallèle, j'ai créé mes entités représentant les tables de ma base de données.

```
php bin/console make:entity
```

Puis tout le reste de mon projet, en m'aidant des cours de Studi et d'internet.