Les choix technologiques

Objectif : décrire et expliquer les choix technologiques permettant de réaliser une plateforme de covoiturage intégrant une approche écologique.

1. Les besoins fonctionnels

- Accéder et consulter les pages de la plateforme.
- Stocker et gérer les données.
- Gérer les accès sécurisés et différents rôles (utilisateur, employé et administrateur) avec des permissions spécifiques pour chacun d'eux.
- Effectuer des opérations CRUD (créer, lire, modifier, supprimer) sur les objets utilisateurs, véhicules, trajets, avis, ...
- Rechercher et filtrer les trajets.
- Permettre aux utilisateurs de contacter les équipes d'EcoRide.
- Notifier les utilisateurs en temps réel et par email.
- Afficher des graphiques pour l'administrateur.

2. Les contraintes

- Utiliser une base de données relationnelle - SQL.
- Utiliser une base de données non relationnelle - NoSQL.
- Rendre l'application fonctionnelle avant mi-février 2025.

3. Les bonnes pratiques

- Sécuriser les données.
- Créer une application responsive.
- Respecter les principes d'accessibilité.
- Respecter les principes d'éco-conception.
- Intégrer les futurs besoins de scalabilité.
- Fournir un code testé, commenté et maintenable.
- Assurer un versionnage et une sauvegarde durable du code.
- Fournir une documentation de l'application.

4. Le back-end

a. PHP 8.3

Les possibilités sont très nombreuses : parmi les options les plus courantes, on trouve Java, Python, Go, JavaScript/Node.js, PHP, Rust, Swift, C#, Ruby, Elixir, Kotlin, ...
Chaque langage a ses spécificités : même si certains ont pu être écartés - car moins performants pour le web (Ruby, Python), trop lourds et/ou surdimensionnés pour les besoins du projet (C#, Java, Swift) ou pas assez fournis en documentation et communauté (Go, Rust) - le choix final reste tout de même un peu arbitraire.

PHP a été retenu pour :

- Sa spécificité pour le développement web,
- Sa fiabilité éprouvée (langage mature et stable),
- Ses frameworks robustes (notamment Symfony et Laravel),
- Son excellent support des bases de données (SQL et NoSQL),
- Sa disponibilité chez les hébergeurs à des prix très abordables,
- Sa très large documentation et sa communauté active,
- Ses performances largement accrues depuis les versions 7 et 8.

Sa version la plus récente sera utilisée pour des raisons de sécurité et de performance.

b. Symfony 6.4 (LTS)

Pour optimiser le développement et respecter la deadline de rendu, un framework sera utilisé : Symfony, une solution dont les avantages sont reconnus :

- Ses fonctionnalités de sécurité robustes pour l'authentification, la gestion des utilisateurs et la protection contre les attaques courantes.
- Son architecture MVC et sa structure modulaire qui le rendent flexible,
- Ses performances optimisées grâce à son système de cache efficace,
- Sa documentation très complète et sa communauté active,
- Son écosystème très riche en composants et bundles parmi lesquels :
    - Doctrine qui propose un ORM (Object-Relational Mapper) et un ODM (Object-Document Mapper) pour gérer les interactions avec les bases de données qu'elles soient relationnelles ou non,
    - Twig qui est un moteur de template qui intègre le PHP au front avec une syntaxe légère et fonctionnelle,
    - Symfony UX dont les bibliothèques servent d'interfaces pour accélérer les interactions JavaScript et dynamiser le front,
    - Validator qui permet de valider des objets en définissant des contraintes,
    - Mailer qui facilite l'envoi d'emails.

C'est la version LTS (Long-Term Support) - pour laquelle des correctifs de sécurité seront édités jusqu'en novembre 2027 – qui sera utilisée.

5. Les bases de données

- Relationnelle : MySQL - est choisi pour sa simplicité d'utilisation, ses performances élevées en lecture, et son intégration facile avec les services d'hébergement. Sa vaste communauté et sa documentation abondante garantissent excellent support, et sa fiabilité en production en fait un choix éprouvé. PostgreSQL aurait été plus adapté si les données à gérer avaient été plus complexes. MySQL sera utilisé pour gérer les objets Utilisateur et Véhicule, et leurs données connexes (besoin de structure et de relations).
- Non relationnelle : MongoDB - est choisi en raison de sa popularité (c'est le plus répandu) et de sa bonne intégration avec le back-end. Il offre également une documentation solide et bénéficie d'une grande communauté. MongoDB répondra aux besoins de scalabilité (en termes de structure et surtout de volume) des objets Trajet et Avis.

6. Le front-end

- HTML 5 pour structurer les pages web de manière sémantique et accessible, et garantir une compatibilité avec tous les navigateurs modernes,
- CSS 3 pour styliser les pages web et assurer un design réactif et accessible sur tous les supports (desktop, tablette et mobile),
- JavaScript pour rendre les pages web interactives et dynamiques, en particulier pour des fonctionnalités comme le filtrage des trajets en temps réel ou l'actualisation de contenu sans rechargement de page (c'est la seule technologie permettant de réaliser cela). Étant donné les besoins techniques de l'application, l'utilisation d'un framework complet n'est pas justifiée, JavaScript sera donc utilisé en version vanilla pour préserver la légèreté du code transmis aux clients. Toutefois, quelques bibliothèques spécifiques seront intégrées via Symfony UX :
    - Stimulus et Turbo : utilisées par le Live Component, elles sont essentielles pour dynamiser les formulaires, gérer la validation en direct et exécuter des requêtes asynchrones, notamment pour mettre en place un système de filtres dynamiques (filtrage des trajets en temps réel).
    - Chart.js : utilisée pour l'espace administrateur, cette bibliothèque permettra la génération de graphiques clairs et interactifs.
- Bootstrap 5 (framework CSS) pour accélérer le développement d'interfaces responsives et esthétiques. Son intégration modulaire garantit que le projet ne sera pas alourdi par des fonctionnalités inutilisées.

7. Le développement

Pour le respect des bonnes pratiques et dans un souci d'efficacité, les outils, services et bibliothèques les plus populaires seront utilisés :

- Git et GitHub - un système de contrôle de version distribué et une plateforme de développement collaboratif offrant des possibilités de CI/CD.
- Composer - un gestionnaire de dépendances pour PHP, le plus réputé.
- PHPStan - un outil d'analyse statique du code PHP. Il sera intégré au projet via Composer et configuré par un fichier "phpstan.neon.dist".
- PHP CS Fixer - un outil d'automatisation pour corriger et formater le code PHP. Il sera intégré au projet via Composer et configuré (fichier ".php-cs-fixer.dist.php") pour respecter les standards définis par Symfony qui incluent ceux des PSR-12 (Extended Coding Style) et PSR-4 (sur le chargement automatique des classes).
- PHPUnit - un framework de test unitaire pour PHP. Grâce à son intégration dans Symfony, il permet également de réaliser des tests fonctionnels.
- Docker et Docker-Compose - un outil de containerisation et son outil dédié pour la configuration. Ils seront utilisés pour la gestion des bases de données en environnement de développement.
- Sass - un préprocesseur CSS utilisé pour améliorer la lisibilité et la maintenabilité du code, permettant des styles organisés et modulaires. Il est indispensable pour personnaliser Bootstrap.
- Webpack Encore – plutôt que l'Asset Mapper de Symfony pour un contrôle plus fin des builds CSS et JavaScript, afin d'y intégrer notamment postCSS et purgeCSS (postprocesseur et plugin optimisant les fichiers CSS).
- Node.js et npm - nécessaires pour le fonctionnement de Webpack Encore : Node.js fournit l'environnement d'exécution JavaScript, npm gère les dépendances et permet d'installer des bibliothèques JavaScript.
- ESLint - un outil d'analyse statique du code JavaScript. Il sera intégré au projet via npm et configuré par un fichier ".eslintrc".

8. Le déploiement

LWS (Ligne Web Services) est choisi comme hébergeur web pour l'application, ses serveurs sont situés en France et leur service client est très réactif. LWS met également en œuvre des mesures de sécurité conformes aux standards de l'industrie, garantissant la protection des données sensibles hébergées. Cette option offre une administration accessible grâce à l'interface CPanel, qui rend la gestion des services PHP et MySQL intuitive.
Cependant, il est à noter que LWS ne prend pas en charge MongoDB directement. Pour pallier cette limite, MongoDB Atlas sera intégré en complément.
Note : les builds CSS et JavaScript seront générés en local avant déploiement.

9. La stack technologique finale