NOM	ROBERT
Prénom	Yvan
Date de naissance	10/03/1999

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/navy4D3/Ecoride.git
Lien de l'outil de gestion de projet : https://www.meistertask.com/projects/tgrhejz40s/join/
Lien du déploiement : ecoride-yvan-robert-ee1f7a4d7a19.herokuapp.com
Login et mot de passe administrateur : admin@ecoride.com – Admin.1234

SANS CES ELEMENTS, VOTRE COPIE SERA REJETEE

Partie 1 : Analyse des besoins

1. **Effectuez un résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots**

Dans le cadre de ma formation, j'ai réalisé Ecoride, une application web de covoiturage développée avec Symfony. L'objectif est de mettre en relation des passagers avec des chauffeurs pour organiser des trajets partagés. Chaque utilisateur inscrit est par défaut passager, mais peut devenir conducteur à condition d'avoir plus de 18 ans et de remplir un formulaire dédié.

L'application permet de rechercher des trajets, d'en proposer, de gérer son profil, d'ajouter un véhicule, et de noter les trajets terminés. Les trajets proposés tiennent compte des étapes intermédiaires grâce à l'API Google Maps Directions, tandis que la saisie des adresses est facilitée par Google Autocomplete. Le système de notation des trajets permet d'avoir une gestion du paiement des crédits en phase avec la satisfaction des passagers.
Un système de rôles (utilisateur, chauffeur, employé, administrateur) est en place, chaque rôle ayant des accès dédiés. Les employés ont accès à un portail permettant de gérer les avis. Les administrateurs disposent d'un tableau de bord avec des statistiques et ont la possibilité de gérer les différents compte, et notamment de créer les comptes employés.

Ce projet m'a permis de me familiariser avec le Framework Symfony et de mobiliser mes compétences en développement backend, en intégration frontend, en gestion de base de données, en sécurité, et en intégration d'APIs externes.

2. Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet

Le projet Ecoride est une application web de covoiturage à vocation écologique, développée pour répondre aux besoins d'une jeune startup française souhaitant réduire l'impact environnemental des trajets en voiture. L'objectif est de proposer une plateforme intuitive, responsive et sécurisée, centrée sur les fonctionnalités essentielles d'un service de covoiturage moderne.

Objectifs fonctionnels - Le produit doit permettre :

- À un **visiteur** de consulter les trajets disponibles via un moteur de recherche (ville de départ, d'arrivée et date), avec des **filtres et tris** (voiture électrique, prix maximum, note minimum, durée).
- D'afficher des trajets contenant : chauffeur, note, nombre de places restantes, prix, durée, mention écologique, bouton de détail.
- À un **utilisateur** de s'inscrire, de se connecter, de **réserver une place** (avec système de crédits) ou de **proposer un trajet** (s'il devient chauffeur).
- À un utilisateur de **devenir chauffeur** avec ses caractéristiques/préférences (description, fumeur, animal…) et d'enregistrer un ou plusieurs véhicules.
- D'**afficher l'historique des trajets**, de pouvoir les **annuler**, **démarrer** et **clôturer** un trajet avec envoi automatique de mails de suivi.
- D'**évaluer les trajets terminés** (note, avis), soumis à modération/gestion par un employé.
- À un **employé** de modérer les avis et de gérer les trajets mal évalués.
- À un **administrateur** de gérer les utilisateurs/employés, de visualiser des **statistiques** (graphique des trajets/jour et revenus en crédits), de suspendre des comptes, et de créer les profils employés.

Contraintes techniques et graphiques :

- La charte graphique doit évoquer l'écologie (vert, nature, sobriété).
- L'interface doit être fluide, responsive et intuitive.
- L'application devra intégrer **les APIs de Google Maps** pour être fonctionnel et proposer des trajets pertinent en France.
- L'application devra être accessible depuis les différents navigateurs web.

Partie 2 : Spécifications technique

1. **Spécifiez les technologies que vous avez utilisé en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments**

- **Symfony** : framework PHP robuste, sécurisé et modulaire. Idéal pour structurer une application métier complexe.
- **Docker** : pour configurer un environnement de travail modulable et reproductible.
- **Twig** : moteur de template intégré à Symfony, facilite la séparation logique/design.
- **Doctrine ORM** : pour la gestion simplifiée des entités et de la base de données relationnelle.
- **Webpack**: pour dynamiser le frontend, pour compiler les assets et faciliter l'intégration des fichiers JS.
- **UX Symfony :** Pour faciliter l'intégration d'icône.
- **SCSS + Bootstrap** : pour un design responsive et moderne.
- **Google Maps API (Directions + Autocomplete)** : permet de suggérer des trajets pertinents et de simplifier la saisie des adresses.
- **Symfony Mailer + Mailtrap** : pour l'envoi de mails (vérification, notifications), en environnement de développement.
- **Heroku** : pour un déploiement flexible et simplifié
- **Recaptcha v3 (Victor PRDH)** : pour lutter contre les robots et sécuriser les formulaires publics.

De manière générale, j'ai utilisé les technologies enseignées au cours de ma formation avec Studi, notamment Symfony, Twig, Doctrine, Bootstrap et Docker qui forment une base solide pour le développement web moderne. Afin de répondre pleinement aux exigences fonctionnelles du projet, j'ai complété cette stack par d'autres solutions pertinentes pour aboutir à une application de covoiturage complète et fonctionnelle.

L'usage de Docker s'est imposé comme un choix pertinent car il permet de garantir un environnement de développement homogène, reproductible et indépendant de la configuration locale de chaque machine. Grâce aux conteneurs, l'application, la base de données SQL, la base NoSQL ainsi que les services annexes (comme Mailtrap ou Mongo Express) sont isolés et peuvent être démarrés en un seul clic. Cela simplifie grandement la collaboration, le déploiement et la maintenance du projet, tout en évitant les problèmes classiques liés aux différences de versions ou de dépendances entre environnements.

J'ai intégré SCSS afin d'améliorer l'efficacité du développement front-end. Grâce à ses fonctionnalités comme les variables, les mixins ou encore l'encapsulation des classes, la structure du code CSS est plus claire, plus modulaire et plus facile à maintenir sur le long terme.

Pour la gestion des trajets et des adresses, j'ai opté pour les API de Google Maps (Directions et Autocomplete). Ce choix s'explique par leur large adoption, leur documentation exhaustive et leur simplicité d'intégration, notamment avec Symfony. Elles m'ont permis de proposer une recherche intelligente et fluide des itinéraires.

Concernant l'envoi des emails (notifications, validation, etc.), j'ai utilisé Mailtrap, une solution recommandée dans la documentation officielle de Symfony. Elle permet de simuler l'envoi d'emails en environnement de développement, gratuitement et de manière sécurisée, ce qui est parfaitement adapté à une application pédagogique.

Enfin, pour le déploiement de l'application, j'ai choisi Heroku. Parmi les différentes solutions présentées durant la formation, c'est celle que j'ai trouvé la plus simple à prendre en main. Elle permet un déploiement rapide, une intégration directe avec GitHub, et une gestion facile des environnements et des bases de données. Pour finir, le recaptcha de google a été intégré à mon application grâce à une intégration facilité de Victor PRDH pour une application symfony.

2. **Comment avez-vous mis en place votre environnement de travail ? Justifiez vos choix. (README.md)**

Afin de développer mon application Symfony dans de bonnes conditions, j'ai commencé par mettre en place un environnement de développement local complet, en installant les outils nécessaires à la fois pour le backend, le frontend, et la base de données.

Outils installés :

- **VisualStudioCode** : éditeur de code très largement utilisé pour le développement.
- **Docker** : Utilisé pour héberger localement la base de données MySQL, le serveur Nginx et les services annexes (MongoDB, phpMyAdmin, etc.).
- **PHP** : installation de PHP 8.2, version compatible avec Symfony 7, indispensable pour exécuter l'application.
- **Composer** : le gestionnaire de dépendances PHP m'a permis d'installer Symfony ainsi que les bundles tiers nécessaires au projet (Doctrine, Mailer, Security, API Google, etc.).
- **Symfony CLI** : outil de ligne de commande spécifique à Symfony, utilisé pour démarrer le serveur local, créer les entités, les contrôleurs, les formulaires, et exécuter les migrations.
- **npm** : utilisé pour gérer les dépendances frontend (Bootstrap, SCSS, Stimulus) et compiler les assets via Webpack Encore.
- **phpMyAdmin** : Outil web, également exécuté dans un conteneur Docker, pour interagir visuellement avec la base de données MySQL. Il m'a permis de vérifier et manipuler les données pendant le développement.
- **MonGoDB Compass** : Application locale permettant de visualiser les base de données NoSQL MonGoDB.

Après duplication du fichier .env en .env.local, j'y ai configuré la variable DATABASE_URL selon les paramètres du service MySQL défini dans Docker, afin d'établir la connexion entre Symfony et MySQL. La base de données a alors été créée avec **Doctrine**.

Une connexion équivalente a été établie avec la base de données MonGoDB en configurant les variables d'environnement MONGODB_URL et MONGODB_DB.

J'ai également configuré **Webpack Encore** afin de compiler automatiquement les fichiers **SCSS et JavaScript**. J'ai complété l'installation de base de webpack afin de bénéficier d'un bénéficiant du rafraîchissement automatique du navigateur à chaque modification.

Cet environnement m'a offert une solide pour le développement et les tests de l'application.

3. **Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.**

Sécurité des formulaires :

- Protection **CSRF** activée dans tous les formulaires Symfony.
- Validation stricte côté serveur avec les **Constraints Symfony** (âge minimum, email valide, type de données, etc.).
- Intégration de **Google reCAPTCHA v3** sur les formulaires publics (inscription, réinitilisation de mot de passe).

Back-end :

- Gestion des rôles et droits d'accès via security.yaml et annotations @IsGranted ou conditions dans les contrôleurs.
- Routes sensibles protégées (ex : /admin, /devenir-chauffeur, etc.).
- Accès aux trajets, notes ou véhicules restreint à leur propriétaire ou à des rôles supérieurs.

Front-end :

- Les erreurs sont limités à travers différentes contraintes gérés dynamiquement en JS.
- Les actions dangereuses sont protégées (confirmation avant suppression, messages d'erreur utilisateur-friendly).
- Les données sensibles ne sont jamais exposées dans le DOM ou les JS non sécurisés.

4. **Décrivez une veille technologique que vous avez effectuée, sur les vulnérabilités de sécurité.**

Dans le cadre du projet et de ma formation, j'ai pu développer mes connaissances sur les vulnérabilités de sécurité suivante :

- **Injection SQL** : évitée grâce à la limitations des libertés sur les input, l'utilisation de Doctrine et des requêtes préparées.
- **Cross-Site Scripting (XSS)** : évité grâce à l'échappement automatique des variables dans Twig et à la limitations des libertés sur les input.
- **Cross-Site Request Forgery (CSRF)** : protection activée sur tous les formulaires.
- **Attaque par force brute** : via **Google reCAPTCHA v3** sur les formulaires publics.

Partie 3 : Recherche

1. **Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source**

Différentes situations rencontrées au cours du développement m'ont amené à effectuer des recherches pour identifier des solutions adaptées ou corriger certains problèmes. Par exemple, lors de l'intégration de **Google reCAPTCHA** dans mes formulaires publics, j'ai exploré plusieurs ressources afin de sécuriser efficacement l'inscription et la prise de contact. C'est ainsi que j'ai découvert le **bundle de Victor PRDH**, une solution open source bien documentée, spécifiquement conçue pour simplifier l'intégration de reCAPTCHA avec **Symfony**.

2. **Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.**

Maîtrisant l'anglais, je n'ai pas eu à traduire ce site. J'ai pu exploiter directement les instructions.
https://dev.to/victorprdh/adding-recaptha-to-symfony-2m60

Partie 4 : Informations complémentaire

1. Autres ressources

Lien vers le fichier FIGMA ayant permis de concevoir le site :

https://www.figma.com/design/e5YHUhWvpAO58U5JPIxw37/EcoRide?node-id=0-1&t=dsyno2BDvfsVIEKw-1

2. Informations complémentaires

L'application développé ne sera exploité que dans le cadre de cette formation, et ne sera pas maintenu une fois validé. Par conséquent, certaines fonctionnalités sont limités :

- Envoie des mails avec Mailtrap : L'envoie réel des mails de validation/information n'est pas mise en place car une évolution vers un compte payant est demandé. Le bon envoie des mails à cependant bien été vérifié dans la zone de test.
- Aucune solution de paiement réel (Stripe, mangopay…) n'a été intégré car un ensemble de justification sur le projet (informations sur l'entreprise et le développeur, finalité de l'application…) est demandé pour avoir une solution fonctionnel. Les crédits sont ajoutés virtuellement au travers d'un formulaire dédié.

Annexe – Diagrammes et Modèles

Modèle conceptuel de données

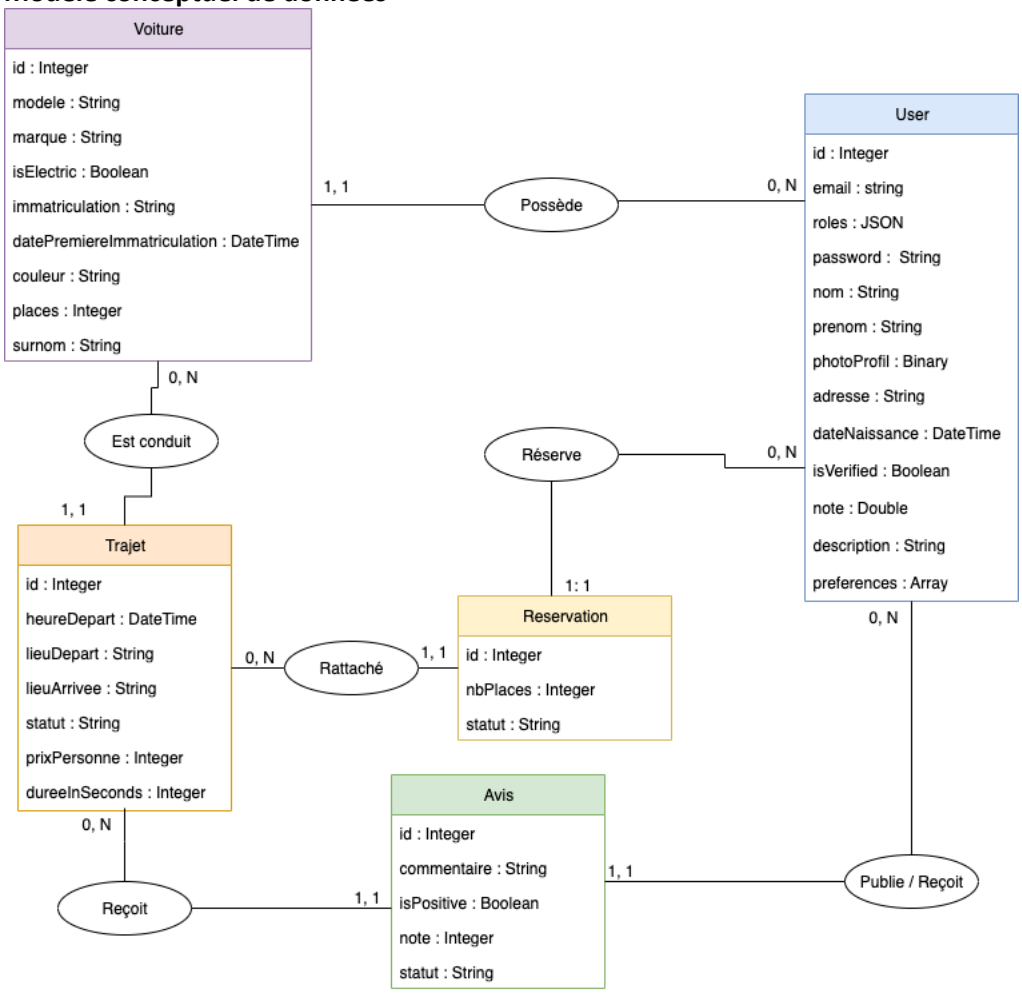
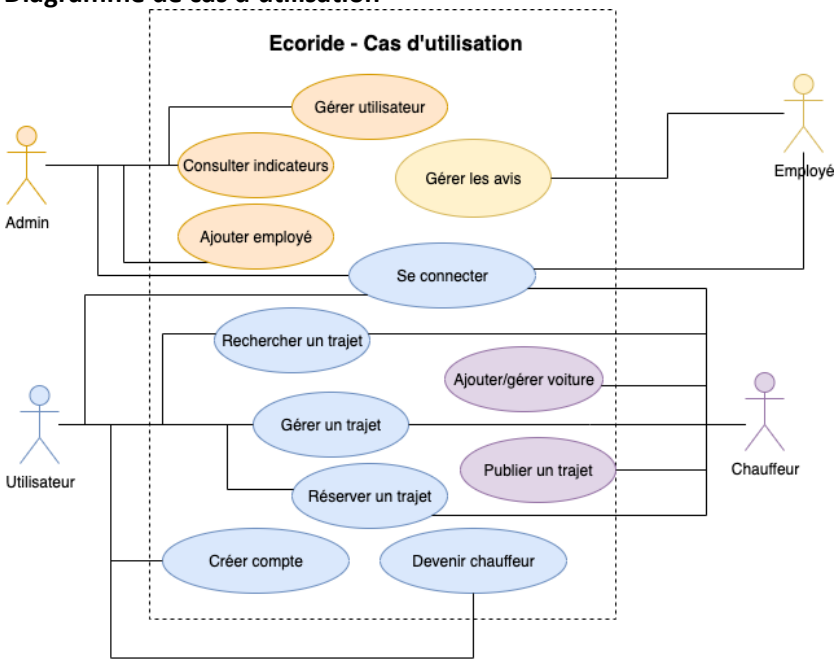


Diagramme de cas d'utilisation



Diagrammes de séquence