

Projet Pour le titre professionnel de
Développeur Web full stack

Projet EcoRide

Application de covoiturage Eco-friendly

Lucas Cherbuin

Sommaire

Introduction	2
Compétences attendues	2
Développement Front end	2
A. Maquetter une application web	2
B. Réaliser une interface utilisateur web statique	2
C. Développer une interface utilisateur web statique	2
Développement Back end	2
A. Maquetter une application web	2
B. Réaliser une interface utilisateur web statique	3
C. Développer une interface utilisateur web statique	3
Résumé	3
Cahier des charges	3
Besoin	3
Objectif	3
Cibles	4
User Story	4
Fonctionnalité détaillé	5
Maquettage	8
Wireframe	8
Charte Graphique et visuel	10
Spécifications techniques	11
Création base de données	11
Diagramme de classes	11
Dictionnaire	12
Base de donnée Non relationnelle	14
Routing	18
Environnement de travail	20
Configuration Repository	20
Front end	21
Middleware	27
Back end	28
Utilisation du sql	30
Fonction Crud	34
Création d'un compte administrateur sécurisé	38
Veille Technologique	43
Veille de sécurité	43
Docker	40
Déploiement	47
Conclusion	49
Annexe	49

Introduction

Ayant commencé mon parcours professionnel dans le domaine de la logistique afin de rentrer dans la vie professionnelle.

Je décide de changer de voie afin de travailler dans un métier lié à l'informatique qui me correspond plus et que je voulais faire depuis l'école.

Travaillant dans une entreprise de grande distribution en tant que logisticien en suisse, je trouvais ça perspicace de réaliser des cours en ligne de développeur web afin de pouvoir travailler dans un établissement où j'ai pu prouver ma valeur en tant qu'employé.

Afin de finaliser ces études, je réalise un projet d'application de manière autonome pour valider mon diplôme.

Le projet consiste en une application de covoiturage ayant des utilisateurs consommateurs de notre application et des utilisateurs côté modérations et d'administration de l'application.

Ce projet « ecoride » m'a permis d'appliquer les compétences apprises durant les cours ainsi que les leçons tirées de l'évaluation des capacités fonctionnelles d'un autre projet auparavant réalisé.

Compétences attendues

Développement Front-end

A . Maquetter une application web

Avant de réaliser la partie code, J'ai réalisé les users stories représentant les utilisateurs ainsi que les différents wireframes d'interfaces du projet en version mobile et desktop pour imaginer la structure de l'application.

Enfin une charte graphique et une maquette pour visualiser le rendu final du projet.

B. Réaliser une interface utilisateur web statique

Etant donné qu'il s'agit d'une application de covoiturage, avoir une interface orientée mobile a été réfléchi car son utilisation se fera principalement par smartphone, cependant, il faut rendre l'application responsive pour les desktops afin que les utilisateurs gérant la modération ainsi que ceux créant des annonces puissent le faire plus « confortablement ».

Une interface simple et accessible sera réalisée afin que n'importe quelle personne puisse utiliser l'application selon son niveau en informatique.

C. Développer une interface utilisateur web dynamique

Pour la gestion des annonces, l'utilisation d'une interface dynamique est cruciale pour la confirmation d'une course, son statut (en prévision, annulée...), l'évolution des informations.

Développement Back-end

A. Création d'une base de données

Comme pour le côté-front end, une maquette pour l'utilisation de bases de données a été réalisée.

Un Schéma de classes pour les différentes tables et leurs relations.

J'ai utilisé Mysql pour la gestion des données

B. Développement des composants d'accès aux données

Pour la gestion des accès aux données, je me suis tourné vers laravel qui propose la migration et mises à jour des données intégré dans le framework,

C. Développer la partie Back-end d'une application web ou web mobile

Pour couvrir le coté back-end de l'application, j'ai utilisé les autres les outils proposé par laravel en créant le routing des pages liées à des controllers ainsi que de middleware pour l'authentification et validation de données.

Résumé

La startup EcoRide a pour but de réduire l'empreinte environnemental en encourageant le covoiturage et à commissionner un développeur de chez Studi pour réaliser l'application.

Le programme servira de services pour les consommateurs avec un rôle « Conducteur » où ils pourront créer des annonces de covoitages avec les informations comme les itinéraires, le prix du voyage ses préférences pour accepter ou non de fumer ou d'avoir un animal durant le voyage, le véhicule utilisé et annoncé le début et fin d'une course.

Cahier Des charges

Besoin

L'application aura besoin d'une interface visiteur ainsi que des espaces pour répondre aux besoins des utilisateurs.

Interface

La demande est que les couleurs et le thème sont orienté coté écologique

Objectif

Rôles et actions

1. **Conducteur**
 - Organise les courses.
 - Gère le statut des courses.
2. **Passager**
 - Choisit une course.
 - Consulte le statut des courses.
 - Laisse un avis.
3. **Employé**
 - Gère les avis des passagers.
 - Contacte les chauffeurs en cas de litige.
4. **Administrateur**
 - Consulte différents graphiques sur l'évolution de la plateforme.
 - Gère les utilisateurs de la plateforme.

Cibles

Un persona est établi pour avoir une idée des profils des différents Acteur :

<p>Passager :</p> <p>Nom : Emilien</p> <p>Age : 19 ans</p> <p>Profession : Etudiant</p> <p>Technologie : Smartphone (iPhone 11, Samsung s 5)</p> <p>Attente : Trouver un moyen pour voyager à moindre frais et faire attention à son empreinte carbone</p> <p>Besoin : Avoir une interface simple et avoir les informations directement</p>
<p>Conducteur :</p> <p>Nom : José</p> <p>Age : 68 ans</p> <p>Profession : Retraité</p> <p>Technologie : Smartphone (Samsung s 5), ordinateur fixe</p> <p>Attente : Pouvoir rentabiliser ses voyages</p> <p>Besoin : Pouvoir remplir les formulaire sur son ordinateur et pouvoir gérer les voyages depuis son smartphone</p>
<p>Employé :</p> <p>Nom : Claudine</p> <p>Age : 33 ans</p> <p>Profession : Référent EcoRide</p> <p>Technologie : Smartphone (Samsung s 5), tablette</p> <p>Attente : Avoir un espace ou les informations sont claires et précise pour faciliter les démarches</p> <p>Besoin : Pouvoir effectuer son travail avec un outil responsive sur 2 machines</p>
<p>Employé :</p> <p>Nom : Gérard</p> <p>Age : 45 ans</p> <p>Profession : Cadre supérieur Ecoride</p> <p>Technologie : Smartphone (iPhone 11), ordinateur fixe</p> <p>Attente : Pouvoir consulté son espace en déplacement facilement</p> <p>Besoin : le site doit pouvoir afficher des graphiques réactif</p>

User Stories

Voici les actions et résultats des utilisateurs

En Tant que	Je souhaite	Afin de
Visiteur	Me connecter	Accéder à mon espace
Visiteur	Me déconnecter	Laisser la place à un autre
Utilisateur sans rôles	Choisir mes rôles	Avoir accès aux menus
Passager	Filtrer les covoiturages	Choisir un covoiturage adapté à mes besoin
Passager	Laisser un avis positif	Recommandé le conducteur

Passager	Annulé un covoiturage	Me délester d'un voyage
Conducteur	Créer un covoiturage	Réaliser un trajet tarifié
Conducteur	Annulé un covoiturage	Ne pas réaliser un voyage
Conducteur	Démarrer et terminer un covoiturage	De recevoir le montant du voyage
Employée	Approuvé un avis	Permettre aux conducteurs d'avoir une meilleur réputation
Employée	Refuser un avis	Supprimer le commentaire s'il n'est pas pertinent ou calomnieux
Employée	Prendre contact avec un conducteur blâmé	Discuter pour avoir sa version des faits.
Administrateur	Consulté les Dashboard	Me faire une idée sur l'évolution de la startup
Administrateur	Créer un user employée	Permettre à un employé de travailler
Administrateur	Modifier un user employée	Modifier les informations comme le nom ou mot de passe
Administrateur	Supprimer un utilisateur	Supprimer un compte pour mauvais comportement, inactivité ou départ d'un employé d'EcoRIde.

Fonctionnalités détaillées

1) Pages Visiteurs

1. Page d'accueil

- Présentation de l'entreprise avec des images.
- Barre de recherche avec possibilité de saisir un itinéraire (ville de départ et d'arrivée).
- Pied de page incluant l'e-mail de l'entreprise et les mentions légales.

2. Menu de navigation

- Lien vers la page d'accueil.
- Lien vers la page des annonces de covoiturage.
- Lien de connexion (ou accès à l'espace utilisateur).
- Lien vers un formulaire de contact.

3. Page des covoiturages

- Liste des différentes annonces de voyages disponibles.
- Formulaire (similaire à celui de l'accueil) pour rechercher un covoiturage (départ et arrivée).
- Chaque annonce présente les informations suivantes :
 - Profil du chauffeur (pseudo, photo, note).
 - Nombre de places restantes.
 - Prix.
 - Date et heure de départ et d'arrivée.
 - Mention indiquant si le véhicule est à énergie propre.
 - Détails supplémentaires sur le chauffeur.
- Si aucun voyage n'est trouvé, le visiteur est invité à modifier ses critères de recherche pour un itinéraire plus proche.

4. Filtre de covoiturage

- Plusieurs filtres peuvent être appliqués pour affiner la recherche :
 - Utilisation d'un véhicule à énergie propre (électrique).
 - Prix minimum.
 - Durée du voyage.
 - Note minimale du chauffeur.

5. Détail d'un covoiturage

- Informations complémentaires sur l'annonce :

- Avis laissés par les passagers précédents.
 - Véhicule utilisé et ses caractéristiques.
 - Préférences du conducteur (musique, animaux, fumeur ou non, etc.).
- 6. Participation à un covoiturage**
- L'utilisateur peut réserver s'il remplit les conditions suivantes :
 - Posséder un compte utilisateur avec le rôle « passager ».
 - Disposer de crédits suffisants.
 - Avoir des places encore disponibles sur le trajet.
 - Le visiteur doit se connecter ou créer un compte pour valider sa participation.
 - Si toutes les conditions sont remplies, un message de confirmation s'affiche. Le passager est alors débité, et le nombre de places disponibles est mis à jour.
- 7. Connexion / Inscription**
- Page permettant à différents utilisateurs de s'identifier (identifiant/mot de passe).
 - Bouton d'inscription redirigeant le visiteur vers un formulaire où il renseigne un pseudonyme, un e-mail et un mot de passe sécurisé.

2) Espace Utilisateurs (Consommateurs)

- 1. Espace Utilisateur**
- À la création du compte, l'utilisateur n'a aucun rôle particulier. Il peut ensuite choisir son rôle : passager, conducteur ou les deux.
 - S'il choisit le rôle de chauffeur, il doit renseigner :
 - Plaque d'immatriculation.
 - Date de la première immatriculation.
 - Informations sur le véhicule (marque, modèle, couleur).
 - Nombre de places disponibles.
 - Préférences pendant le voyage (fumeur ou non, animaux acceptés ou non, etc.).
- 2. Création d'un voyage**
- Le chauffeur peut créer une annonce contenant :
 - Ville de départ et d'arrivée.
 - Prix du trajet.
 - Sélection du véhicule (avec possibilité d'en ajouter un depuis le formulaire).
 - Une taxe de 2 crédits est prélevée pour le fonctionnement de l'application.
- 3. Historique des voyages**
- Permet de visualiser, pour les deux rôles (passager ou conducteur), l'état des covoitages :
 - En prévision.
 - En cours.
 - Terminé.
 - Annulé.
 - Les utilisateurs peuvent annuler leurs voyages, ce qui envoie un e-mail aux participants/conducteurs et rembourse automatiquement les passagers concernés.
- 4. Commencer et terminer un covoiturage**
- Le chauffeur peut démarrer un covoiturage (statut « en prévision ») pour le passer à « en cours ».
 - À la fin du trajet, il clique sur « arrêter » pour le basculer en statut « terminé ».
 - Une fois le voyage terminé, un e-mail est envoyé aux passagers pour leur demander de laisser un avis.
 - Les avis sont ensuite validés par les employés. En cas de problème, un commentaire négatif peut être associé au trajet.

3) Espace Employé

- 1. Gestion des avis**
- L'employé valide ou invalide les avis.
 - Chaque avis validé modifie la note globale du chauffeur et apparaît sur ses annonces.

- Un onglet permet aussi de gérer les litiges liés à un covoiturage négatif. L'employé peut consulter les informations relatives au voyage, aux participants et contacter le chauffeur par e-mail pour obtenir des explications.

4) Espace Administrateur

1. Graphiques

- L'administrateur peut consulter l'évolution de l'utilisation du service :
 - Nombre de covoiturages validés par jour (graphique).
 - Crédits gagnés en fonction des jours (graphique).
- Le total de crédits accumulés doit également être affiché.

2. Gestion des comptes utilisateurs

- Possibilité de créer, modifier et supprimer un compte.
- Les comptes clients peuvent être supprimés uniquement par l'administrateur.

Maquettage

Afin de rendre le visuel de mon projet, je vais utiliser l'application Figma pour son côté accessible, accessible par n'importe qui pour des avis ainsi que la communauté proposant des assets pour sa création.

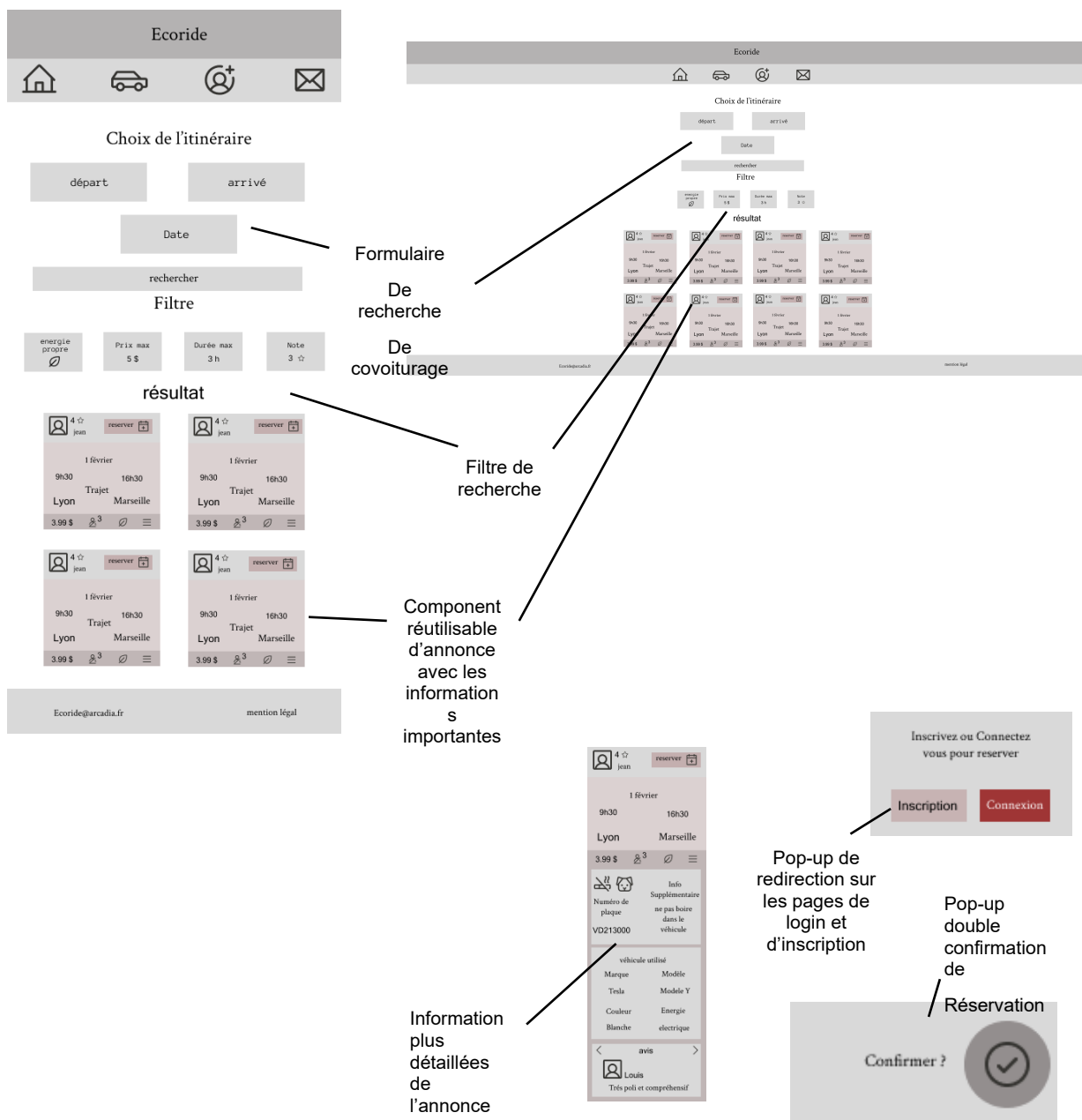
WireFrames

J'ai réalisé des wireframe mobile et desktop étant les 2 plateformes susceptibles d'être utilisés

Pour une meilleure lisibilité, vous pouvez consulter le Figma

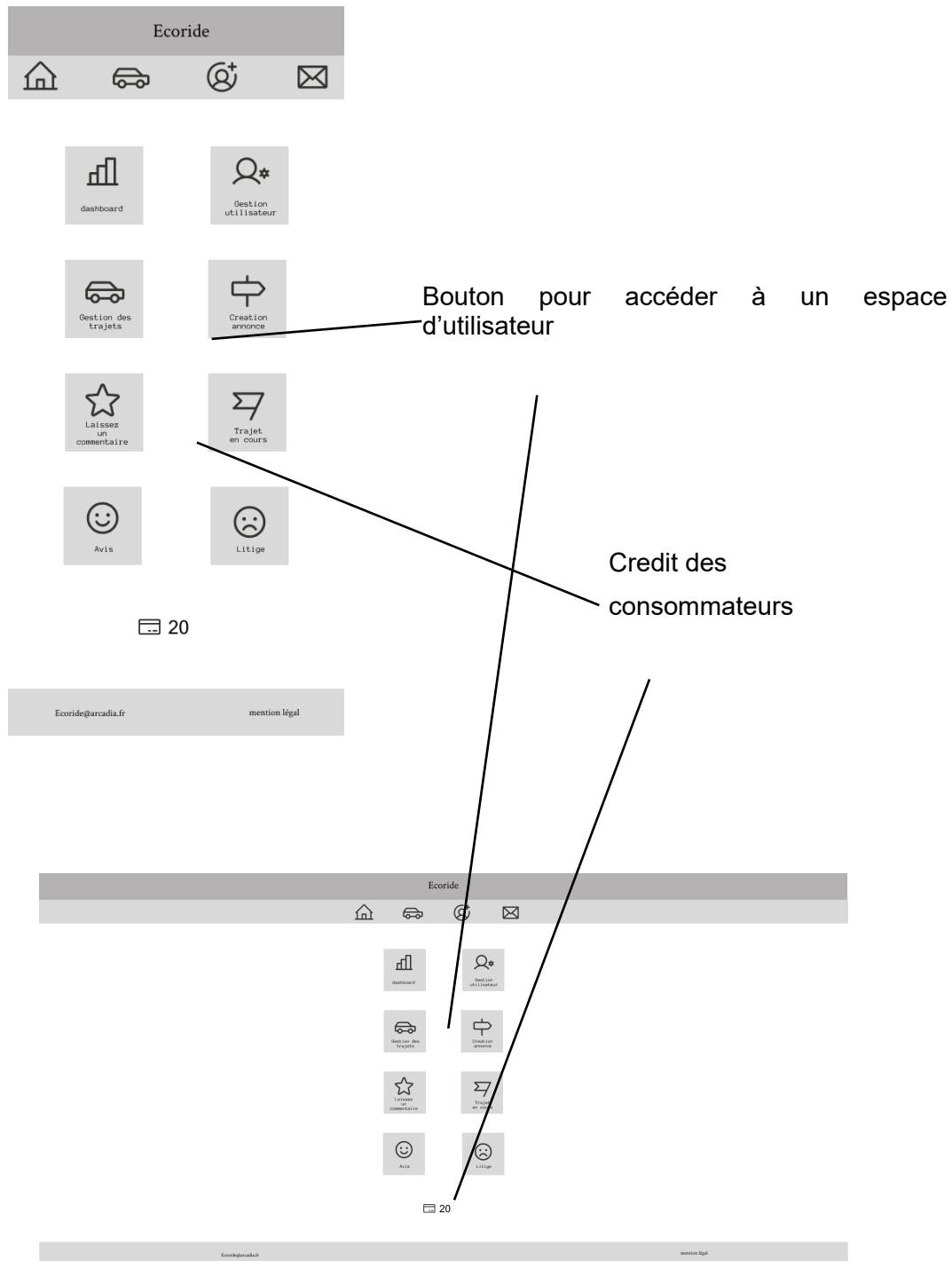
<https://www.figma.com/design/Z4mvqQUXw8fv5kKWirAY3y/Untitled?node-id=366-485&p=f&t=4Sa81dkrC7IKybSH-0>

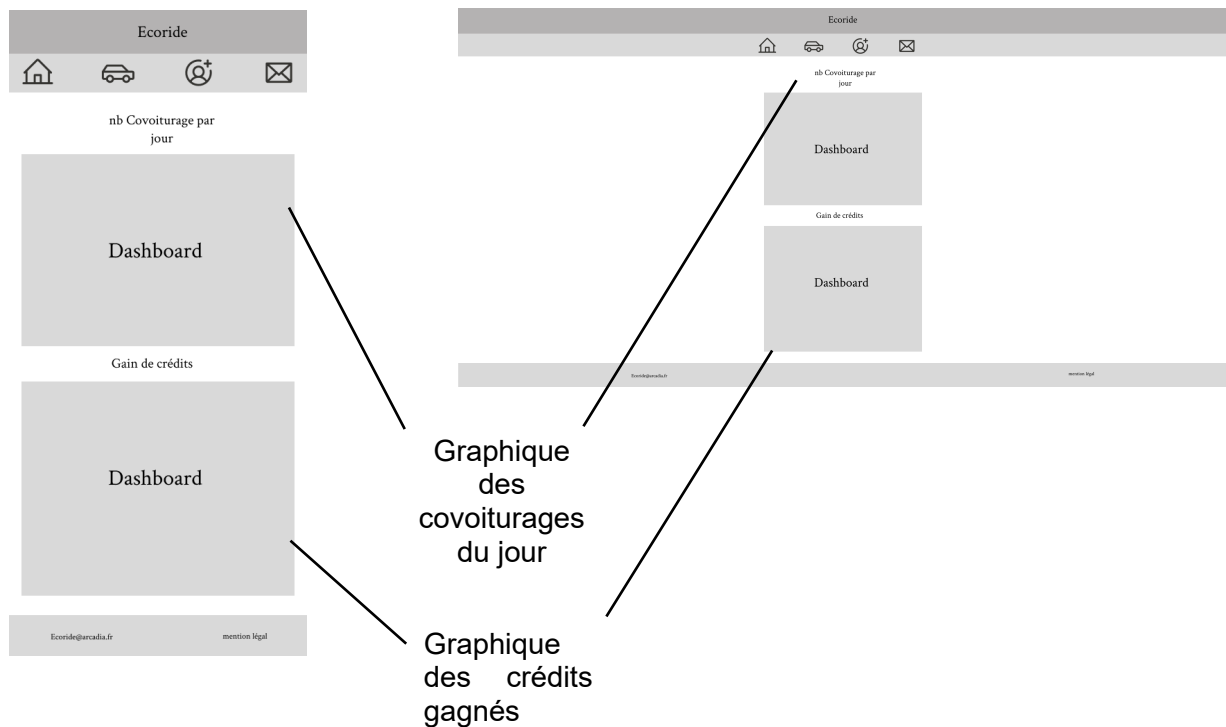
Pages covoiturages



Menu utilisateur

Un menu de navigation servira pour la navigation des différents utilisateurs





Dans les wireframes et les maquettes, j'utilise la bibliothèque « phosphor icon » pour construire des visuels et pourra être facilement importable sur mon projet via le système CDN, de plus les fonctionnalités de Figma me permettent de créer des composants qui sont des éléments réutilisables pour les réutiliser et modifier selon les besoins.

Charte Graphique et visuel

Afin de convenir aux attentes graphiques, une couleur verte claire a été utilisée pour souligner le côté écologique que l'application souhaite montrer.

(Retrouvez les maquettes en annexe)

Spécification techniques

Pour réaliser ce projet, je me suis tourné vers des technologies adaptées pour mon projet, les technologies et outils utilisés devront faire en sorte que mon application puisse être utilisée sur les navigateurs communs.

Technologies utilisées

- **Front-end**
 - Bootstrap
 - Laravel (Routing)
- **Back-end**
 - Laravel
 - MySQL
 - MongoDB
- **Déploiement**
 - Heroku
- **Autres outils**
 - Visual Studio Code
 - Git et GitHub
 - PhpMyAdmin
 - XAMPP
 - Docker

Création base de données

Base de données relationnelles

Avis [Id, note, commentaire, valid, user_id, covoiturage_id]

Conducteur [Id, immatriculation, date immatriculation, modele_id, user_id]

Covoiturage [Id, départ, arrive, date, heure départ, heure arrive, Energie verte, user_id, conducteur_id, status_id, prix]

Préférence [Id, fumeur, animal, détail, user_id]

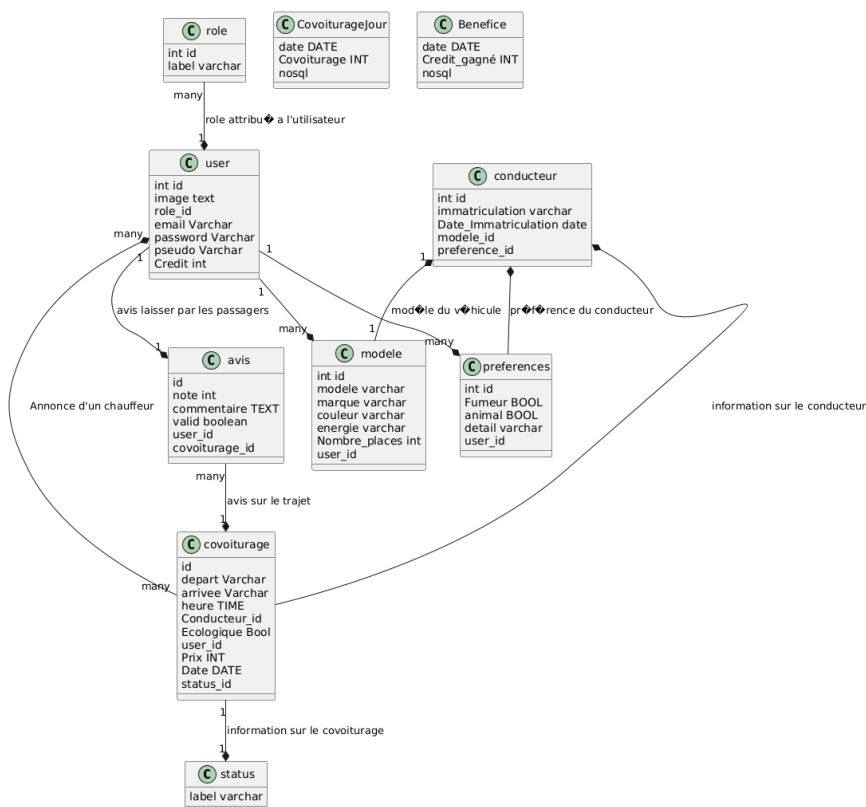
Modèles [Id, modèle, marque, couleur, Energie, nombre places]

Rôle [Id, label]

Statu [Id, label]

Users [Id, pseudo, image, email, password, crédit, role_id]

Diagramme de classe



Dictionnaire

Nom de la donnée	But	Type	Contrainte
Avis id	Code numérique d'un avis	Int	Primary key
Note	Évaluation d'un trajet	Int	Not null
Commentaire	Détail sur un trajet	Text	Not null
Valid	Confirmation si l'avis est affiché sur une annonce	Boolean	Not null
User_id	Relation avec la table user	Int	Foreign key
Covoiturage_id	Relation avec la table covoiturage	Int	Foreign key
Conducteur id	Code numérique d'un conducteur	Int	Primary key
Immatriculation	Numéro en exadécimal de la plaque d'immatriculation	Text	Not null
Date_immatriculation	Date d'immatriculation d'un véhicule	Date	Not null
Modele_id	Relation avec la table model	Int	Foreing key
Preference_id	Relation avec la table preference	Int	Foreing key
Covoiturage_id	Relation avec la table covoiturage	int	Foreing Key
Covoiturage id	Code numérique d'un covoiturage	Int	Primary key

Depart	Lieu de départ	Text	Not null
Arrive	Lieu d'arrivé	Text	Not null
Heure	Heure de départ	Time	Not null
Ecologique	Indique si le véhicule à une Energie propre	Boolean	Not null
User_id	Relation avec la table utilisateur	Int	Foreign key
Status_id	Relation avec la table statuts	Int	Foreign key
Prix	Montant du voyage	Int	Not null
Préférence id	Code numérique d'une préférence	Int	Primary key
Fumeur	Si le voyage est fumeur	Boolean	Not null
Animal	Si le conducteur accepte les animaux	Boolean	Not null
detail	Information complémentaire	Text	null
User_id	Relation avec la table user	Int	Foreign key
Modelé id	Code numérique d'un modèle de voiture	Int	Primary Key
Modèle	Modèle de la marque	Text	Not null
Marque	Marque du véhicule	Text	Not null
Couleur	Couleur du véhicule	Text	Not null
Energie	Type de carburant du véhicule	Text	Not null
Nombres places	Nombre de places passagers disponible	Int	Not null
Rôle id	Code numérique d'un rôle	Int	Primary Key
Label	Définition du rôle attribué	Text	Not null
Status id	Code numérique d'un statuts	Int	Primary Key
Label	Etat d'un statuts de covoiturage	Text	Not null
User id	Code numérique d'un user	Int	Primary Key
Pseudo	Pseudonyme d'un utilisateur	Text	Not null
Image	Photo de profil	Text	
Email	Adresse mail de l'utilisateur	Text	Not null
Password	Mot de passe hacher de l'utilisateur	Text	Not null
Crédit	Montant du crédit d'un utilisateur	Int	Not null
Role_id	Relation avec la table rôle	Text	Foreing key

Base de données non relationnelle

Comme indiqué sur le diagramme de classes, 2 tables n'ont aucune relation entre elle et servira pour DES données non relationnels.

Il s'agit de données qui vont s'agréments pour les statistiques qui se retrouveront dans la section « Dashboard » de l'administrateur.

Comme base NoSql, j'ai initié mongoDb qui va gérer les incrémentation d'addition et de soustraction des données pour le total de covoiturage par jour et credits gagnés.

Configuration

J'ai du installer le package laravel *composer require jenssegers/mongodb* me permettant d'avoir toute les fonctionnalités liées a Mango.

Ensuite, il faut configurer la connexion de la base de donnée dans le fichier des variables d'environnement et ajouter un tableau de la configuration dans le fichier *database.php*

.Env

database.php

```
DB_CONNECTION=mongodb
DB_HOST=127.0.0.1
DB_PORT=27017
DB_DATABASE=EcorideNoSQL
DB_USERNAME=
DB_PASSWORD=
```

```
'mongodb' => [
    'driver' => 'mongodb',
    'host' => env('DB_HOST', default: '127.0.0.1'),
    'port' => env('DB_PORT', default: 27017),
    'database' => env('DB_DATABASE'),
    'username' => env('DB_USERNAME'),
    'password' => env('DB_PASSWORD'),
],
```

Modele

Comme pour la base de donnée relationnel, j'ai créer des modèles avec des attributs contenant les informations des Dashboard

Covoiturages effécutés

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

4 references | 0 implementations
class CovoiturageDonnees extends Model
{
    0 references
    protected $connection = 'mongodb';
    0 references
    protected $collection = 'covoitruage_donnees';

    0 references
    protected $fillable = ['date', 'total'];
}
```

Credits gagné

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

2 references | 0 implementations
class CreditGagneDonnees extends Model
{
    0 references
    protected $connection = 'mongodb';
    0 references
    protected $collection = 'creditGagne_donnees';

    0 references
    protected $fillable = ['date', 'total'];
}
```

Dans le modeles, je spécifie avec `$connection` d'utiliser la base de donnée nosql « mangoDb » au lieu de MySql

Collection Permet de spécifier la collection MongoDB avec un nom sous forme de snake_case pour la compatibilité avec Laravel dans le controller.

Fillable seront les champs qui vont être inséré

Controller

Je configure les controller pour récupérer le nombre de covoiturage dans la journées, les credits gagnés ou remboursés de la journée ainsi que le rendu de la vue du dash board.

Nombres de voyages

```
class DashboardAdminController extends Controller
{
    // Incrémenter le nombre de covoiturations par jour (NoSQL)
    0 references | 0 overrides
    public function covoiturage(Request $request): void
    {
        $date = now()->toDateString();

        $stat = CovoiturageDonnees::firstOrCreate(
            attributes: ['date' => $date],
            values: ['nb_covoiturations' => 0]
        );

        $stat->increment(column: 'nb_covoiturations');
    }
}
```


Dans la fonction *covoiturage* \$date va récupérer la date du jour et \$stat va récupérer la variable \$date et le nombre de covoiturage.

Le second \$stat incrémentera le nombre decovoiturage réalisée.

Credits gagnés et perdu

```
0 references | 0 overrides
public function covoiturageCreation(Request $request): void
{
    $user = $request->user();

    if ($user->credit >= 2) {
        $user->decrement('credit', 2);

        Covoiturage::create(attributes: [
            'status_id' => 'en prévision', // statut par défaut
        ]);
    }
}

//Annulation + remboursement

0 references | 0 overrides
public function covoiturageAnnulation(Covoiturage $covoiturage, Request $request): void
{
    $user = $request->user();

    if ($covoiturage->status->label != 'annulé') {
        $user->increment('credit', 2);

        $covoiturage->update(attributes: [
            'status_id' => 'annulé', // statut "annulé"
        ]);
    }
}

/**
 * + Mise à jour des crédits gagnés quotidiennement (NoSQL)
 */
0 references | 0 overrides
public function credit(Request $request): void
{
    $date = now()->toDateString();

    $gain = CreditGagneDonnees::firstOrCreate(
        attributes: ['date' => $date],
        values: ['creditGagne_donnees' => 0]
    );

    $gain->increment(column: 'creditGagne_donnees');
}
```

La fonction *covoiturageCreation* va prélever une taxe de 2 euros quand un Conducteur va créer une nouvelle annonce se basant sur le status « en prévision » qui est celui par défaut lors de la création d'un covoiturage.

La 2ème fonctions *covoiturageAnnulation* va quant a elle remboursé le conducteur si il a annulé son voyage, il se base sur le status « annulé ».

La dernière fonction *credit* va récupéré la date du jour ainsi que les différents calcul réalisé dans la journée.

dashboard

```
// Dashboard Vue

1 reference | 0 overrides
public function dashboard(): JsonResponse|mixed
{
    $view1 = view(view: 'admin.dashboard.get-covoiturage-stats')->render();
    $view2 = view(view: 'admin.dashboard.get-credit-stats')->render();

    return response()->json(data: [
        'covoiturage' => $view1,
        'credit' => $view2,
    ]);
}
```

La fonction dashboard va récupérer les résultats des statistiques des données et le rendre un visuel pour l'utilisateur (\$view1 pour le nombre de covoiturage et \$view2 pour les crédits remportés

Vue

Pour rendre un affichage de statistique, j'ai implémenter des dashboards proposé par Bootstrap customizable en y ajoutant les données et en changeant les couleurs.

Covoiturages

```

You, 1 second ago | 1 author (You)
<script>
  document.addEventListener("DOMContentLoaded", function() {
    .then(response => response.json())
    .then(data => {
      const labels = data.map(item => item.date);
      const values = data.map(item => item.nb_covoiturages);

      const ctx = document.getElementById('CovoiturageChart').getContext('2d');
      new Chart(ctx, {
        type: 'bar',
        data: {
          labels: labels,
          datasets: [{
            label: 'Nombre de covoiturages',
            data: values,
            backgroundColor: 'rgba(255, 255, 255, 0.6)',
            borderColor: 'rgba(163, 230, 53, 1)',
            borderWidth: 1
          }]
        },
        options: {
          responsive: true,
          scales: {
            y: {
              beginAtZero: true,
              ticks: {
                stepSize: 1
              }
            }
          }
        }
      });
    });
  });
</script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

```

Credits gagnés

```

You, 1 second ago | 1 author (You)
<script>
  document.addEventListener("DOMContentLoaded", function() {
    .then(response => response.json())
    .then(data => {
      const labels = data.map(item => item.date);
      const values = data.map(item => item.creditGagne_donnees);

      const ctx = document.getElementById('CreditChart').getContext('2d');
      new Chart(ctx, {
        type: 'bar',
        data: {
          labels: labels,
          datasets: [{
            label: 'Nombre de crédits',
            data: values,
            backgroundColor: 'rgba(255, 255, 255, 0.6)',
            borderColor: 'rgba(163, 230, 53, 1)',
            borderWidth: 1
          }]
        },
        options: {
          responsive: true,
          scales: {
            y: {
              beginAtZero: true,
              ticks: {
                stepSize: 1
              }
            }
          }
        }
      });
    });
  });
</script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

```

On configure les données des variables \$stat des fonctions « Covoiturages » et « credit » pour rendre le résultat.

Routing des différentes pages

Avec Laravel, j'ai accès à un routing par fichier, cela permet une meilleur organisation selon les différents acteurs du site.

	ENDPOINT	OUTPUT	CONTROLLER	AUTH
		VISITEUR		
GET	/	Page d'accueil		Aucune connexion
GET	/covoiturage	Page des annonces	RechercheCovoiturageController	Passager
GET	/contact	Demande de contact avec l'entreprise	ContactController	Aucune connexion
GET	/login	Page de Connexion	AuthenticatedSessionController	Aucune connexion
POST	/login	Utilisateur connecter, redirection sur son espace	AuthenticatedSessionController	User, admin, employé, conducteur, passager
GET	/register	Pages d'inscription	RegisterUserController	Aucune connexion
POST	/register	Création d'un compte avec le role « Utilisateur »	RegisterUserController	Utilisateur
POST	/upload	Enregistre l'image de profil de l'utilisateur sur le serveur	RegisterUserController	Aucune connexion
POST	/logout	Déconnexion de l'utilisateur	AuthenticatedSessionController	Aucune connexion
		UTILISATEUR		
GET	/user/menu-utilisateur	Accès aux menus pour les consommateur		User, conducteur, passager
GET	/user/role-choice	Page de choix des roles	RoleChoiceController	User
POST	/user/role-choice	Modification du rôle selon les choix de l'utilisateur	RoleChoiceController	Conducteur, passager ou les 2 en même temps
		CONDUCTEUR		
GET	/user/covoiturage/create	Formulaire de création d'une annonce	CovoiturageController	Conducteur
POST	/user/covoiturage/store	Soumission d'une annonce	CovoiturageController	Conducteur

GET	/user/historique	Afficher des différentes annonces créées	StatusController	Conducteur
PATCH	/user/historique/{id}/delete	Annulation d'un covoiturage	StatusController	Conducteur
GET	/user/covoiturageDemmarage	Page pour démarrer et terminer un voyage	StartRideController	Conducteur
PUT	/user/covoiturageDemmarage/{id}/changer-status	Commencer ou terminer une course	StartRideController	Conducteur
		PASSAGER		
GET	/user/historique	Afficher des différentes annonces créées	StatusController	Passager
PATCH	/user/historique/{id}/delete	Annulation de sa venue à un covoiturage	StatusController	Passager
GET	/user/avis	Page pour donner un avis	AvisController	Passager
POST	/user/avis/{id}/send	Soumission d'un avis pour validation	AvisController	Passager
		EMPLOYE		
GET	/employe/validation	Page pour les validations d'un avis	ValidationController	Employe
POST	/employe/validation{id}/validateAvis	Validation d'un avis pour l'afficher dans une annonce	ValidationController	Employe
POST	/employe/validation/{id}/rejectAvis	Refus d'un avis en le supprimant	ValidationController	Employe
GET	/employe/Litige	Page pour les avis négatifs	ValidationController	Employe
POST	/employe/litige/{id}/validateAvis	Validation d'un avis pour l'afficher dans une annonce	ValidationController	Employe
POST	/employe/litige/{id}/rejectAvis	Refus d'un avis en le supprimant	ValidationController	Employe
		ADMIN		
GET	/admin/userCreation/	Tableaux des utilisateurs de l'application	ProfilController	Admin
GET	/admin/userCreation/create	Formulaire de création d'un employé	RegisteredController	Admin
POST	/admin/userCreation/create	Ajout d'un nouvel employé	RegisteredController	Admin

GET	/admin/userCreation/{id}/edit	Formulaire pour modifier des données d'un employée	ProfilController	Admin
DELETE	/admin/userCreation/{id}	Suppression d'un utilisateur	ProfilController	Admin
GET	/admin/Dashboard	Page Pour les Dashboard	DashboardAdminController	Admin
GET	/admin/Dashboard/get-covoiturage-stats	Dashboard du nombre de covoiturage par jour	DashboardAdminController	Admin
GET	/admin/Dashboard/get-credit-stats	Dashboard des gains et perte des crédit ainsi que son montant total	DashboardAdminController	Admin

Environnement de travail

Pour ce projet, j'ai utilisé plusieurs outils pour mon travail

J'utilise Visual code studio comme éditeur avec des extensions facilitant mon travail

j'ai installé XAMPP afin de disposer d'une version compatible de PHP, ainsi que Composer pour exécuter des commandes spécifiques. J'ai ensuite lancé la commande :

Composer create-project laravel/laravel Ecoride

directement dans l'éditeur de code, ce qui m'a permis d'obtenir la structure de base de l'application.

Dans le fichier des variables d'environnement, j'ai configuré plusieurs valeurs :

- La clé d'application.
- La base de données SQL.
- La base de données NoSQL.
- Le système de messagerie (mailer).

Pour garantir un environnement développement reproductible, j'ai intégré Docker qui me permet de faciliter le déploiement sur une autre machine et simplifier la mise en production.

Enfin, j'ai vérifié le bon fonctionnement du projet en local grâce à la commande :
Php artisan serve

Configuration Repository

Afin de versionner et de stocker mon code, j'ai opté pour Git et GitHub. Voici les étapes suivies :

- 1. Installation et configuration de Git**
 - Installation de Git.
 - Configuration de mes informations personnelles (nom, e-mail) pour le projet.
- 2. Création du dépôt GitHub**
 - Configuration d'un nouveau dépôt sur GitHub.
 - Récupération de l'URL du dépôt pour le lier à mon dépôt local.
- 3. Configuration du .gitignore**
 - Ajout du fichier des variables d'environnement (et autres fichiers sensibles) dans .gitignore.

- Préparation d'un fichier séparé spécialement conçu pour l'hébergeur en production.

4. Commandes Git

- Git add . : ajout de tous les fichiers pour le prochain commit.
- Git commit -m "Initial commit" : premier commit dans le projet.
- Git branch -M main : création (ou renommage) de la branche principale du dépôt.
- Git push -u origin main : envoi du code vers GitHub.

Enfin, pour travailler efficacement sur différentes fonctionnalités, je crée plusieurs branches indépendantes. Cela évite d'impacter l'intégralité du projet et garantit une meilleure lisibilité du suivi.

Develop	Branch pour le développement principal, c'est ici que le ou les branches si dessous seront merge après test et seront corrigée. Develop est une sous-branche de Main qui servira, dès que l'application est fonctionnelle, à être merge sur Main pour la production.
Sous branche de Develop	
Features/LoginRegister	Code relatif pour configurer l'authentification, la connexion, les différents rôles pour un utilisateur
Features/adminUser	Création d'un compte administrateur sécurisée
Features/Visiteurs	Fonctionnalités relatifs aux page visiteurs
Features/PagesAdmin	Fonctionnalités relatifs aux page administrateur
Features/pagesVisiteurs	Fonctionnalités relatifs aux page visiteurs
Features/CustomerUser	Fonctionnalités relatifs aux page des consommateurs
Features/EmployerUser	Fonctionnalités relatifs aux page des employés

Front end

Blade

Laravel offre un moteur de template Blade avec une Syntaxe pour créer des vues dynamiques en garantant un code structuré.

Pour montrer le potentiel de Blade, j'ai réaliser la maquette de la page de covoiturage

J'ai produit les annonces sous forme de composants réutilisable car je vais l'afficher sur plusieurs pages de différentes manières

```

<div class="annonce">
  <button type="button">Réserver <i class="ph ph-calendar-plus"></i></button>

  <!-- Affichage des informations de covoiturage -->
  <div class="principal">
    <p>{{ $date ?? 'Date non disponible' }} à {{ $heure ?? 'Heure non disponible' }}</p>
    <p>Trajet : {{ $depart ?? 'Départ inconnu' }} + {{ $arrive ?? 'Arrivée inconnue' }}</p>
  </div>

  <!-- Affichage des détails du modèle -->
  <div class="detail">
    <p>{{ $prix ?? 'Prix inconnu' }} € - {{ $nombrePlaces ?? '0' }} places - Énergie : {{ $energie ?? 'Non spécifiée' }}</p>
  </div>
  <div class="accordion-detail">
    <!-- Plus de détails ici -->
  </div>

  <!-- Section des avis -->
  <div class="avis">
    <div class="Carousel-controls">
      <a class="carousel-control-prev" href="#avisCarousel" role="button" data-bs-slide="prev">
        <i class="ph ph-caret-left"></i>
      </a>
      <p>Avis</p>
      <a class="carousel-control-next" href="#avisCarousel" role="button" data-bs-slide="next">
        <i class="ph ph-caret-right"></i>
      </a>
    </div>

    <!-- Affichage des avis validés -->
    @if (!empty($avis))
      @foreach ($avis as $avisItem)
        @if ($avisItem['valid'])
          <div class="card">
            <div class="card-body">
              <p>{{ $pseudo ?? 'Utilisateur inconnu' }}</p>
              <p>{{ $avisItem['commentaire'] }}</p>
            </div>
          </div>
        @endif
      @endforeach
    @else
      <p>Aucun avis disponible.</p>
    @endif
  </div>
</div>

```

Blade me permet d'intégrer la structure if...else facilement comme pour vérifier si des avis d'un conducteur est disponible

JavaScript

Sur la pages de covoiturage, quand un utilisateur va réserver un covoiturage, un pop up va apparaître pour confirmer la réservation et une autre apparaîtra invitant à s'identifier ou se connecter si l'utilisateur n'a pas un rôle de passager.

Le 1^{er} script va recevoir en paramètre `PlacesRestantes` en paramètre, ensuite exécuté la const `confirmation` affichant le pop up pour confirmer la réservation.

Si il est confirmé, cela va déclenché la const `fetch` qui va renvoyé une requête AJAX vers la route `/annonce/reservation/{nombres_places}` pour decrementer le champ `nombres_place` de la table `modele` en relation avec le covoiturage en répondant avec une réponse JSON via la const `data`.

Le bloc `if data.success` va envoyer un message si la réservation est faite ou annoncé une erreur en cas d'échec.

Si l'instruction `try` laisse passer une exeception, le bloc `Catch` renverra une erreur Ajax

```

</form>

<script>
  async function confirmReservation(nombres_places) {
    const confirmation = await new Promise(resolve => {
      const result = confirm('Confirmer la réservation ?');
      resolve(result);
    });

    if (confirmation) {
      try {
        const response = await fetch(`/annonce/reservation/${nombres_places}`, {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
            'X-CSRF-TOKEN': "{{ csrf_token() }}"
          },
          body: JSON.stringify({})
        });

        const data = await response.json();

        if (data.success) {
          alert('Réservation confirmée !');

          window.location.href = data.redirect;
        } else {
          alert('Une erreur est survenue.');
```

Script quand l'utilisateur n'a pas un rôle de passager

```

2  @else
3  <script>
4    document.addEventListener("DOMContentLoaded", () => {
5      const choice = confirm("Vous devez être connecté pour réserver ?");
6
7      window.location.href = choice
8      ? "{{ route(name: 'login') }}"
9      : "{{ route(name: 'register') }}";
10     fetch("{{ route(name: 'reservation.choice') }}", {
11       method: "POST",
12       headers: {
13         "Content-Type": "application/json",
14         "X-CSRF-TOKEN": "{{ csrf_token() }}"
15       },
16       body: JSON.stringify({ login: choice })
17     })
18     .then(response => response.json())
19     .then(data => {
20       if (data.redirect) {
21         window.location.href = data.redirect;
22       }
23     })
24     .catch(error => {
25       console.error("Erreur AJAX :", error);
26     });
27   });
28 </script>

```

Si L'utilisateur n'a pas un rôle de passager, on lui affichera le second script, la ligne *document* s'affiche si la page est entièrement chargée et lance la const que nous devons être connecter pour réservé.

La ligne *windows* redirige à la page de connexion ou d'inscription.

Le bloc fetch envoie une requête sécurisée avec un jeton CSRF pour éviter.

Avec *.then* si le back end renvoie une url, l'utilisateur est redirigé.

La ligne catch affiche une erreur si il y une anomalie.

Css et Responsive

Pour rendre ajouter du design sur ma page, je vais créer Plusieurs fichier CSS réutilisable et l'importer sur ma page

```
<link href="{{ asset(path: 'assets/css/form.css') }}" rel="stylesheet">
<link href="{{ asset(path: 'assets/css/app.css') }}" rel="stylesheet">
<link href="{{ asset(path: 'assets/css/annonce.css') }}" rel="stylesheet">
```

J'ai ensuite construis le desing par rapport à la maquette

Form.css

```
@import url('https://fonts.googleapis.com/css2?family=Anonymous+Pro:ital,wght@0,400;0,700;1,400;1,700&display=swap');

.custom-form {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  gap: 10px; /* Espacement entre les éléments */
  width: 500px;
  max-width: 400px;
  margin: 20px auto; /* Centrer le formulaire */
  padding: 20px;
  border-radius: 50px;
}

/* Style des champs input */
.form-input {
  width: 100%;
  padding: 10px;
  background-color: transparent !important;
  font-family: 'Anonymous Pro', monospace;
  border-color: #4CAF50;
  color: black;
  border-radius: 60px;
  font-size: 15px;
  outline: none;
}

.form-button-ms-4 {
  width: 95%;
  font-family: 'Anonymous Pro', monospace;
  padding: 12px;
  background: white;
  font-size: 15px;
  font-weight: bold;
  border: #4CAF50;
  border-radius: 60px;
  cursor: pointer;
  transition: 0.3s;
}
```

App.css

```
/* Importation de la police */
@import url('https://fonts.googleapis.com/css2?family=Anonymous+Pro:ital,wght@0,400;0,700;1,400;1,700&display=swap');

/* Appliquer la police globale */
body {
  font-family: 'Anonymous Pro', monospace;
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  display: flex;
  flex-direction: column;
  min-height: 100vh; /* Assurer que le contenu prend toute la hauteur de l'écran */
}

/* HEADER */
.title {
  background-color: #68A72A;
  text-align: center;
  padding: 15px 0;
}

nav {
  background-color: #C1F27D;
  text-align: center;
  gap: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 10px 0;
}

/* Liens du menu */
.nav-item {
  font-size: 44px;
  display: inline-block;
  margin: 0 15px;
}

/* FOOTER */
footer {
  background-color: #68A72A;
  text-align: center;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
  padding: 20px;
  margin-top: auto; /* Fait en sorte que le footer reste en bas */
}

/* S'assurer que le contenu principal prend tout l'espace */
main, h1 {
  flex: 1;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}
```

Annonce.css

```
You, last month | 1 author (You)
.annonce {
  background-color: #a3e635;
  border-radius: 10px;
  padding: 10px;
  width: 300px;
  font-family: Arial, sans-serif;
}

.profile {
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.profile img {
  width: 40px;
  height: 40px;
  border-radius: 50%;
}

button {
  background-color: #000;
  color: #fff;
  border: none;
  padding: 5px 10px;
  border-radius: 5px;
  cursor: pointer;
}

.principal, .detail, .preferences, .Modele, .avis {
  background-color: #d9f99d;
  border-radius: 8px;
  padding: 8px;
  margin-top: 10px;
  text-align: center;
}

.accordion-button {
  background: none;
  border: none;
  font-size: 18px;
  cursor: pointer;
}

.plaque p, .detail p, .Modele p {
  font-weight: bold;
}

.informations p {
  margin: 5px 0;
}

You, last month + ajout des fonctionnalités, accueil, contact ave...
.Carousel-controls {
  display: flex;
  align-items: center;
  justify-content: space-between;
}
```

Les différentes valeurs me permettent de configurer les couleurs, la position sur l'écran, l'espacement entre les différents éléments et la taille des éléments.

Pour rendre les CSS responsive, j'ai intégré des media queries pour que l'adaptation sur Smartphone et tablette

```
@media (min-width: 768px) {
  .container {
    font-size: 18px;
    max-width: 720px;
    margin: 0 auto;
  }
}

/* Desktop (min-width: 1024px) */
@media (min-width: 1024px) {
  .container {
    font-size: 20px;
    max-width: 960px;
  }
}

You, 1 second ago • Uncomm
```

Le résultat depuis l'application se trouve en annexe.

Middleware

Afin de sécuriser les envois entre le Controller et les requêtes http, j'ai implémenté plusieurs middlewares relatifs aux utilisateurs

ImageUploadMiddleware

Comme indique dans mon dictionnaire pour les bases de données, une image peut être enregistrée comme photo de profil, le but du Middleware sera de filtrer les formats d'images pour

Prévenir l'envoi de fichier malveillant.

```
You, last week | 1 author (You)
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

You, last week | 1 author (You) | 4 references | 0 implementations
class ImageUploadMiddleware
{
    0 references | 0 overrides
    public function handle(Request $request, Closure $next): Response
    {
        // Vérifie si une image est bien envoyée
        if ($request->hasFile(key: 'image')) {
            $file = $request->file(key: 'image');
            $extension = strtolower(string: $file->getClientOriginalExtension());

            // Vérifie si l'extension est valide
            if (!in_array(needle: $extension, haystack: ['jpg', 'jpeg', 'png'])) {
                return response()->json(data: ['error' => 'Image non supportée'], status: 400);
            }
        }

        return $next($request);
    }
}
```

RoleMiddleware

Les chemins aux pages utilisateurs doivent être limités au rôle dédié, le middleware renverra une interdiction d'accès si un rôle non-autorisé essaye de s'y rendre.

```

<?php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Symfony\Component\HttpFoundation\Response;

You, 7 hours ago | 1 author (You) | 4 references | 0 implementations
class RoleMiddleware
{
    0 references | 0 overrides
    public function handle($request, Closure $next, ...$roles): mixed
    {
        $user = Auth::user();

        if (!$user || !in_array(needle: $user->role->label ?? '', haystack: $roles)) {
            abort(code: 403, message: 'Accès interdit');
        }

        return $next($request);
    }
}

```

Back end

Implémentation base de données

Dans le projet, je crée les différentes tables sous forme de modèle pour gérer les données.

```

You, 4 weeks ago | 1 author (You)
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

You, 4 weeks ago | 1 author (You) | 12 references | 0 implementations
class Modele extends Model
{
    use HasFactory;

    0 references
    protected $fillable = [
        'id',
        'modele',
        'marque',
        'couleur',
        'nombres_places',
        'energie'
    ];

    0 references | 0 overrides
    public function users(): HasMany
    {
        return $this->hasMany(related: User::class);
    }
}
You, 2 months ago • initiation du projet ecoride sous laravel

```

Le protected \$fillable seront les champs utilisable en rapport avec le modèle. La fonction HasMany sert avec d'autres tables en relation.

Pour remplir la base de données avec les tables requis je doit générer une migration avec une commande *artisan* et je remplis le fichiers avec les informations.

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        // Création des tables indépendantes en premier
        Schema::create(table: 'role', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'label')->unique();
            $table->timestamps();
        });

        Schema::create(table: 'status', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'label')->unique();
            $table->timestamps();
        });

        Schema::create(table: 'modele', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'modele');
            $table->string(column: 'marque');
            $table->string(column: 'couleur');
            $table->integer(column: 'nombres_places')->default(value: 0);
            $table->string(column: 'energie');
            $table->timestamps();
        });
    }
};

```

Je précise le type de données correspondant des différentes tables et j'exécute *php artisan migrate* pour remplir ma base de donnée.

En allant sur PhpMyAdmin, on peut voir que les données sont remplit

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> avis	★	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> conducteur	★	1	InnoDB	utf8mb4_unicode_ci	80,0 kio	-
<input type="checkbox"/> covoiturage	★	1	InnoDB	utf8mb4_unicode_ci	80,0 kio	-
<input type="checkbox"/> migrations	★	0	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> modele	★	1	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> permissions	★	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> preferences	★	1	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> role	★	6	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> session	★	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> status	★	4	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> user	★	2	InnoDB	utf8mb4_unicode_ci	16,0 kio	-

Grâce a l'option d'exproation des tables, on peut télécharger un fichier avec les différents script SQL réaliser dans la base de données avec la migration depuis Laravel

```

Run | Select
CREATE TABLE `avis` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `note` int(11) NOT NULL DEFAULT 18,
  `commentaire` text NOT NULL,
  `valid` tinyint(1) NOT NULL DEFAULT 0,
  `user_id` bigint(20) UNSIGNED NOT NULL,
  `covoiturage_id` bigint(20) UNSIGNED NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

```

-----

--
-- Structure de la table `role`
--

```

```

Run | Select
CREATE TABLE `role` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `label` varchar(255) NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

Utilisation du SQL

Malgrès que j'essaie de faire en sorte que l'interaction des utilisateur entre mon application et la base de donnée soit configurer de manières à éviter les problèmes, le risque zero n'exite pas.

C'est pour ça que, si un problème survient, je puisse corriger le problème malgrès tout

Modifier une données

Imaginons qu'un employé c'est tromper dans la validation d'un avis et qu'il a approuver trop vite un avis sans l'avoir consulté complètement.

L'action fera passer, dans la table avis, le champ *valid* de type boolean de « 0 » à « 1 » qui sera considérer comme étant afficher dans les covoiturations.

Voici comment corriger dans la base de donnée avec une commande sql

```

Run | + Tab | JSON
SELECT * FROM avis;
Run
UPDATE avis SET valid = '0' WHERE avis.id = 1;

```

SELECT * FROM avis va selectionner la table avis

Update avis va mettre à jour la table

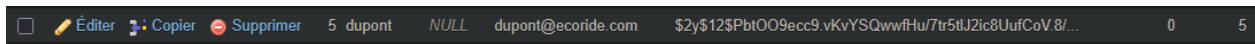
Set modifie en sorte que le champ *valid* doit passer à la valeur 0

Where indique quel quel identifiant je veux modifier (ici avis.id 1)

Supprimer

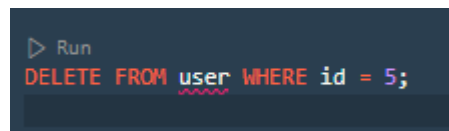
La situation indique que l'administrateur n'arrivant pas supprimer l'utilisateur Dupont depuis son espace.

En regardant dans la base de donnée je remarque que l'id de l'utilisateur Dupont est le numéro 5



J'exécute par précaution **SELECT * FROM user WHERE id = 5;** pour vérifier qu'il s'agit bien du bon compte

Voici comment supprimer le compte



DELET FROM user indique que je voudrais supprimer une ligne de la table user

WHERE user id = 5 s'agit de la condition que cela doit être l'id 5 qui doit être supprimer

Données de test

Pour vérifier si ma base de données est fonctionnelle, je réalise des tests en remplissant les tables avec des données factisment réalistes.

Avant tout, je configure une autre tables de données SQL spécialement pour les tests, ce qui permet de ne fausser la table principal lors de la migration.

Laravel propose de réaliser des tests des différents modeles des tables avec des classes Factory.

On génères les différent fichiers avec `php artisan make:factory + nom_du_modelFactory`


```

*/
0 references | 0 implementations
class CovoiturageFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'id' => fake()->numberBetween(),
            'depart' => fake()->text(),
            'arrivee' => fake()->text(),
            'date' => fake()->date(),
            'heure_depart' => fake()->time(),
            'heure_arrivee' => fake()->time(),
            'prix' => fake()->numberBetween(int1: 5,int2: 50),
            'ecologique' => fake()->boolean(),
            'user_id' => function (): mixed {
                return User::inRandomOrder()->first()->id ?? User::factory()->create()->id;
            },
            'conducteur_id' => function (): mixed {
                return Conducteur::inRandomOrder()->first()->id ?? Conducteur::factory()->create()->id;
            },
            'status_id' => function (): mixed {
                return Status::inRandomOrder()->first()->id ?? Status::factory()->create()->id;
            },
        ];
    }
}

```

Dans cette classe, j'ai configuré les différents champs avec la fonction `fake()` → `type()` qui les remplira selon le type, la particularité est que je peux préciser le *type* pour générer une donnée crédible (exemple `fake()` → `name()`, qui générera un nom de famille)

J'ai ajouté une fonction dans les champs relationnel qui permet d'éviter que la génération échoue si une des autres tables sont vides.

Pour remplir la base de données. Je génère un fichier Seeder qui me permettra avec la commande `php artisan db:seed --FactorySeeder` de remplir les tables

```
<?php

namespace Database\Seeders;

use App\Models\Avis;
use App\Models\Conducteur;
use App\Models\Covoiturage;
use App\Models\Modele;
use App\Models\User;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

0 references | 0 implementations
class FactorySeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    public function run(): void
    {
        Avis::factory()->count(10)->create();
        Conducteur::factory()->count(10)->create();
        Covoiturage::factory()->count(10)->create();
        Modele::factory()->count(10)->create();
        User::factory()->count(count: 10)->create();
    }
}
```

Voici la table Covoiturage après migration

Le fichier d'exportation de PHPmyAdmin montre les scripts SQL réalisés

```
-- Structure de la table 'covoiturages'
--
-- Run | Select
CREATE TABLE `covoiturages` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `depart` varchar(255) NOT NULL,
  `arrivee` varchar(255) NOT NULL,
  `date` date NOT NULL,
  `heure_depart` time NOT NULL,
  `heure_arrivee` time NOT NULL,
  `prix` int(11) NOT NULL,
  `ecologique` tinyint(1) NOT NULL DEFAULT 0,
  `user_id` bigint(20) UNSIGNED NOT NULL,
  `conducteur_id` bigint(20) UNSIGNED NOT NULL,
  `status_id` bigint(20) UNSIGNED NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Déchargement des données de la table 'covoiturages'
--
-- Run | Select
INSERT INTO `covoiturages` (`id`,`depart`,`arrivee`,`date`,`heure_depart`,`heure_arrivee`,`prix`,`ecologique`,`user_id`,`conducteur_id`,`status_id`,`created_at`,`updated_at`) VALUES
(25168849, 'Que quia sit ut error fuga odit. Neque possim quam omnis et. Vitae nobis velit reprehenderit. Nam perspicatis natus ab magni hic at. Odit est veritatis facilis quam eius in asperiores voluptas.', 'Et et distinctio qui dolore aut esse. Accus (29286889, 'Sint eaque in eaque nihil consequatur qui. Odit suscipit officii culpa ut harum non. Atque fugit ut commodi aliquid.', 'Quia perferendis velit adipisci tempore molestias. Non deserunt et qui placeat corporis. Modi laborum qui quia iusto. Optio (42225449, 'Temporibus asperiores perspicatis rerum esse voluptatem eveniet voluptas. Commodi doloreque in fugiat distinctio. Minus optio eos consequatur.', 'Sunt et incident voluptates. Consequatur ex voluptas doloribus esse perspicatis doloreque et. (61373089, 'Accusamus ut dolore temporibus quam atque beatae. Ab provident et eos carum repudiandae consequatur. Est quod maiores excepturi vel nobis voluptates unde vitae.', 'Ante voluptas quisquam culpa et. Voluptates et alias deserunt non facere. Cupidi (414737456, 'Maxime quo voluptas quam eum ab ipsa qui. Sint facere et soluta. Et est animi at atque quia architecto. Voluptatem sunt neque dolorum non maxime dolor unde. Dolorem sit delectus et itaque.', 'Aut ut animi corrupti cupiditate corporis expedita. (591812202, 'Enim et ducimus quae delentii sunt. Voluptatem qui asperiores sit eius voluptas quos. Optio iusto harum magni tenetur magnae iure et. Ut ut et modi culpa quo et nihil.', 'Voluptate perferendis placeat quasi voluptatibus sit eius. Ut tenetur tem (608819302, 'Dolore repudiandae voluptas omnis at ipsum tempore molestias. Qui magni dolor expedita laudantium. Qui aut aut quos rerum pariatur excepturi qui.', 'Delentii soluta omnis incident quae soluta ducimus quibusdam sunt. Velit asperiores a nostrum et (6118466594, 'Est fugit nisi ducimus consectetur. Dolores a quae consequatur voluptas soluta totae iste. Quae non aut quam unde laborum recusandae. Dicta quisque esse aut perferendis sit distinctio.', 'Et corrupti incident assumenda architecto. Harum quae (613562112, 'Praesentium reprehenderit ea sint possimus. Aut quae quis quo dolore illo. Aperiam provident cumque suscipit temporibus.', 'Neque quo laqueid voluptatem unde non consequatur dolores. Anet doloreque animi illum consequatur delectus. Sed aut in (843993130, 'Numquam blanditiis atque magni voluptate. Natus sunt aut rerum nemo dolores rerum.', 'Sunt et ea corrupti dolores. Ex qui facilis aperiam omnis mollitia. Sunt culpa voluptatem molestias perferendis ab fuga blanditiis praesentium. Earum ut venia (914337093, 'Nisi ut provident aut molestias doloreque officia sed. Queraat culpa ducimus velit doloribus dolor quo pariatur modi.', 'Dolor quae sit illum. Queraat fugiat saepe necessitatibus alias. Animi sed velit laborum aut sed. Qui modi in cum cupidita (1435242463, 'Vel nobis nihil dolores ab eveniet doloribus consequatur. Qui quia non est alias est quae dolor expedita. Odio nulla aut qui ab est eum.', 'Ut amet porro quidem. Enia occaecati porro iste eos culpa. Accusamus aliquam voluptates sed quo. Praese (1455868249, 'Unde qui odio at et quo. Totae deserunt odio voluptatem aut illo quia. Ut reiciendis fugit dolores quaerat. Est omnia vel aliquam aperiam.', 'Sintillique eveniet iure voluptates quae nesciunt. Asperiores asperiores fuga rerum aliquid consequatur (1639744463, 'Doloribus enim unde mollitia est debitis. Nobis illum nam ab tempora hic. Cumque est ex est natus. Assumenda illo rem non in.', 'Est aut voluptatum totam incident beatae autem. Maxime quae sequi non ducimus earum in. Nemo quidem quia ut quasi. (1722591621, 'Nam nobis dolores animi nobis repellendus. Aut recusandae beatae minus accusamus. Sunt debitis consequatur molestiae.', 'Quod vitae occaecati officii ut ut quia. Perferendis id sequi aut voluptatem. Vel id esse asperiores nihil omnis autem.', (1755781206, 'Facilis dignissimos error explicabo iusto iusto sit et nulla. Nisi est illo non quia. Voluptatum nobis accusamus sint fugit commodi voluptatem minima.', 'Pariatur nemo corrupti hic sunt optio molestiae. Alias maiores veniam doloribus consequat (1774638239, 'Consequatur tempora qui iusto ipsum quos repellat. Ante eum facere rerum expedita. Sit vel necessitatibus asperiores. Ex aut ab quo tenetur corporis aperiam tenetur.', 'Sit tempore aut aut dolores. Et rerum quod aut earum eos provident.', 'A (1789418136, 'Qui quo est aliquam debitis maiores quia. Vero consequatur sed quia officia esse. Et sunt natus iusto quia.', 'Nam suscipit quae earum quo est aut autem. Dolores consequatur qui nati quia voluptate explicabo. Magnas omnis ipsum eligendi qui et (1918579288, 'Voluptatibus unde inventore eum ratione praesentium deserunt eos. Facilis molestiae cupiditate placeat magnam. Illum qui minus fugit voluptates. Est nisi rerum molestiae excepturi.', 'Sed dicta possimus est eum occaecati eum. Delentii assumend (1977595922, 'Assumenda eos et voluptas at. Quis explicabo architecto aut voluptatibus debitis. Debitis alias autem iste dolores ut. In ut optio qui esse et non.', 'Molestiae aliquid eaque nulla aliquid consequatur. Laborum nihil iusto non vel. Dignissimos (2146368365, 'Quia nemo quis velit et odio. Architecto magni et ea voluptatem. Nemo aut sapiente voluptatem odio.', 'Cumque vitae omnis blanditiis sequi. A in magnas rerum molestiae. Omnis minus ut excepturi.', '1975-10-27', '01:04:04', '11:56:35', 25, 1, 9
```

Fonction Crud

Pour que l'utilisateur puisse interagir avec la base de données , il faut intégrer un controller avec les méthodes d'opération de CRUD.

La tabel des annonces étant parfaite pour montrer le concept, je commence par utiliser la commande php artisan make:controller CovoiturageController --resource me générant plusieurs méthodes dont celle qui m'intéresse (read, create, update, delete)

```
You, last week | 1 author (You)
<?php

namespace App\Http\Controllers;

use App\Http\Requests\CovoiturageRequest;
use App\Models\Covoiturage;
use App\Models\Modele;
use App\Models\Avis;
use Illuminate\Http\Request;

12 references | 0 implementations | You, last week | 1 author (You)
class CovoiturageController extends Controller
{
    0 references | 0 overrides
    public function index(): View
    {
        // Récupération de tous les covoitages avec leurs modèles associés
        $covoitages = Covoiturage::with(relations: 'modeles')->get();
        return view('covoiturage.index', data: compact('covoitages'));
    }

    2 references | 0 overrides
    public function store(CovoiturageRequest $request): RedirectResponse
    {
        // Création du covoiturage
        $preparedcovoiturage = Covoiturage::create(attributes: [
            'depart' => $request->depart,
            'arrive' => $request->arrive,
            'heure' => $request->heure,
            'EnergieVerte' => $request->EnergieVerte ?? false,
        ]);

        // Vérifier si un véhicule doit être ajouté
        if ($request->filled(key: ['modele', 'marque', 'couleur', 'nombre_places', 'energie'])) {
            $preparedmodele = Modele::create(attributes: [
                'modele' => $request->modele,
                'marque' => $request->marque,
                'couleur' => $request->couleur,
                'nombre_places' => $request->nombre_places,
                'energie' => $request->energie,
            ]);

            // Associer le véhicule au covoiturage si une relation existe
            if (method_exists(object_or_class: $preparedcovoiturage, method: 'modeles')) {
                $preparedcovoiturage->modeles()->attach($preparedmodele->id);
            }

            You, last month • ajout des fonctionnalités pour les utilisateurs
            return redirect()->route(route: 'covoiturage.index')->with(key: 'success', value: 'Covoiturage enregistré avec succès.');
```

```

1 reference | 0 overrides
public function show($id): JsonResponse|mixed|View
{
    $covoiturage = Covoiturage::with(relations: 'modeles')->find(id: $id);

    if (!$covoiturage) {
        return response()->json(data: ['message' => 'Covoiturage non trouvé'], status: 404);
    }

    $avis = Avis::where(column: 'covoiturations', operator: $id)->get();
    $modeles = $covoiturage->modeles ?? collect();

    return view(view: 'components.annonce', data: compact(var_name: 'covoiturage', var_names: 'avis', 'modeles'));
}

0 references | 0 overrides
public function update(Request $request, $id): JsonResponse|mixed
{
    $covoiturage = Covoiturage::find(id: $id);

    if (!$covoiturage) {
        return response()->json(data: ['message' => 'Covoiturage non trouvé'], status: 404);
    }

    $validatedData = $request->validate(rules: [
        'depart' => 'sometimes|string',
        'arrive' => 'sometimes|string',
        'heure' => 'sometimes',
        'EnergieVerte' => 'sometimes|boolean',
    ]);

    $covoiturage->update($validatedData);

    return response()->json(data: $covoiturage);
}

0 references | 0 overrides
public function destroy($id): JsonResponse|mixed
{
    $covoiturage = Covoiturage::find(id: $id);

    if (!$covoiturage) {
        return response()->json(data: ['message' => 'Covoiturage non trouvé'], status: 404);
    }

    $covoiturage->delete();

    return response()->json(data: ['message' => 'Covoiturage supprimé avec succès']);
}

```

Index : Page pour la création d'un covoiturage

Read : me permet de lire les informations d'une donnée.

Create : Formulaire avec les champs à remplir

Store : créer une nouvelle données en remplissant les champs demandés

Update : Mettre à jour une donnée

Delete : Supprimer une donnée de la base de données

On ajoute les différents fonctions dans le fichier de routing avec le controlleur de covoiturage, je rajoute la route middleware pour que cela ne soit que des roles de conducteurs qui puissent utiliser La fonctionnalité Crud des covoiturations.

```

Route::middleware(middleware: ['auth', 'role:ROLE_CONDUCTEUR|ROLE_CHAUFFEURPASSAGER'])->group(callback: function (): void {
    Route::resource(name: 'covoiturage', controller: CovoiturageController::class)->only(methods: [
        'index', 'store', 'show', 'update', 'destroy'
    ]);
});

```

J'ai pu trouver dans la documentation comment déclarer les routes en une ligne au lieu d'en faire des routes individuels.

Vues

Index

```
@extends (view: 'base')
<link rel="stylesheet" href="{{ asset(path: 'assets/css/menu.css') }}">
@section(section: 'content')
<a href="{{ route(name: 'menuAdmin') }}" class="back-button">
  <i class="ph ph-arrow-left"></i>
</a>

<table>
  <thead>
    <tr>
      <th>mes Covoiturages</th>
    </tr>
  </thead>
  <tbody id="userTableBody">
    @foreach ($Covoiturages as $covoiturage)
      <tr data-role="{{ $covoiturage->id }}">
        @include(view: 'components.annonce')
        <td>
          <a href="{{ route(name: 'conducteur.annonceCreation.edit', parameters: ['id' => $covoiturage->id]) }}"><i class="ph ph-note-pencil"></i></a>
          <form method="post" action="{{ route(name: 'admin.annonceCreation.destroy', parameters: ['id' => $covoiturage->id]) }}" onsubmit="return confirm('Êtes-vous sûr ?');">
            @csrf
            @method('DELETE')
            <button type="submit"><i class="ph ph-trash"></i></button>
          </form>
        </td>
      </tr>
    @endforeach
  </tbody>
</table>

<div class="button-group">
  <a href="{{ route(name: 'conducteur.annonceCreation.create') }}" class="button">
    <i class="ph ph-plus-circle"></i>
  </a>
</div>
@endsession
```

L'index est ici ou va se retrouver les information de covoiturages de l'utilisateur avec les différentes fonctionnalités Crud que l'on va consutlé

Create et edit

```
{{-- Voyage --}}
<div class="mb-3">
  <label for="depart" class="form-label">Départ :</label>
  <input type="text" name="depart" id="depart" class="form-control" value="{{ old(key: 'depart') }}" required>
</div>

<div class="mb-3">
  <label for="arrive" class="form-label">Arrivée :</label>
  <input type="text" name="arrive" id="arrive" class="form-control" value="{{ old(key: 'arrive') }}" required>
</div>

<div class="mb-3">
  <label for="heure" class="form-label">Heure :</label>
  <input type="time" name="heure" id="heure" class="form-control" value="{{ old(key: 'heure') }}" required>
</div>

<div class="mb-3">
  <label for="date" class="form-label">date :</label>
  <input type="date" name="date" id="date" class="form-control" value="{{ old(key: 'date') }}" required>
</div>

<div class="mb-3">
  <label for="prix" class="form-label">Prix :</label>
  <input type="int" name="prix" id="prix" class="form-control" value="{{ old(key: 'heure') }}" required>
</div>

<div class="form-check mb-3">
  <input type="checkbox" name="Ecologique" id="Ecologique" class="form-check-input" {{ old(key: 'Ecologique') ? 'checked' : '' }}>
  <label class="form-check-label" for="Ecologique">Ecologique ?</label>
</div>
```

```

{{-- Ajout d'un véhicule (optionnel) --}}
<h4>Ajouter un véhicule</h4>

<div class="mb-3">
  <label for="modele" class="form-label">Modèle :</label>
  <input type="text" name="modele" id="modele" class="form-control" value="{{ old(key: 'modele') }}">
</div>

<div class="mb-3">
  <label for="marque" class="form-label">Marque :</label>
  <input type="text" name="marque" id="marque" class="form-control" value="{{ old(key: 'marque') }}">
</div>

<div class="mb-3">
  <label for="couleur" class="form-label">Couleur :</label>
  <input type="text" name="couleur" id="couleur" class="form-control" value="{{ old(key: 'couleur') }}">
</div>

<div class="mb-3">
  <label for="nombre_places" class="form-label">Nombre de places :</label>
  <input type="number" name="nombre_places" id="nombre_places" class="form-control" value="{{ old(key: 'nombre_places') }}">
</div>

<div class="mb-3">
  <label for="energie" class="form-label">Énergie :</label>
  <input type="text" name="energie" id="energie" class="form-control" value="{{ old(key: 'energie') }}">
</div>

<button type="submit" class="btn btn-primary">Enregistrer</button>
</form>
</div>

```

La page va permettre d'inscrire les informations nécessaires pour créer ou modifier une annonce et proposera de sélectionner un véhicule de la table « modele » ou de pouvoir en inscrire un nouveau.

Après avoir rempli le formulaire et confirmé la base de données stockera l'annonce remplie par le conducteur

	id	depart	arrivee	date	heure	prix	ecologique	user_id	conducteur_id	status_id
<input type="checkbox"/> Éditer <input type="button" value="Copier"/> <input type="button" value="Supprimer"/>	1	Bordeau	Marseille	2025-04-21	09:45:00	3	0	3	1	1

Pour des questions de sécurité,

Delete

Contrairement aux autres opérateurs, delete est une méthode sous forme de bouton dans la page index qui va supprimer l'annonce.

L'utilisation de formulaires est indispensable pour que l'utilisateur puisse intégrer des données dans les champs de la base de données

Show

La vue Show est le composant que j'ai présenté dans la section Front end en voulant montrer l'utilisation du potentiel de Blade

Pour finir, je vais générer un fichier *request* qui gère me permettant de configurer la validation des données acceptés, ce qui garantit les entrées des données et évite d'injecter du code malveillant par les formulaires.

```
0 references | 0 overrides
public function rules(): array
{
    return [
        //voyage
        'depart' => 'required|string',
        'arrive' => 'required|string',
        'heure' => 'required|time',
        'Ecologique' => 'nullable|boolean',
        'prix' => 'required|int|min:1',
        'date' => 'required|date',
        //choix du véhicule
        'modeles' => 'required|collection',
        //ajout d'un véhicule
        'modele' => 'nullable|string|max:20|min:5',
        'marque' => 'nullable|string|max:20|min:5',
        'couleur' => 'nullable|string|max:20|min:5',
        'nombre_places' => 'nullable|string|max:20|min:5',
        'energie' => 'nullable|string|max:20|min:5'
    ];
}

0 references | 0 overrides
public function messages(): array
{
    return [
        'depart.required' => 'Indiquez un départ',
        'arrive.required' => 'Indiquez une arrive',
        'heure.required' => 'indiquez une heure',
        'prix.required' => 'Indiquez un prix',
        'date.required' => 'indiquez une date pour la course',
        'modeles' => 'veuillez ajouter un véhicule pour le voyage'
    ];
}
```

La fonction *configure* l'autorisation des requêtes

rules de paramétrer les conditions des données à entrer

Et *messages* pour afficher un text si une condition n'est pas respecté

Création d'un compte Administrateur sécurisé

Une des volonté du Client été que le compte Administrateur ne soit pas un compte utilisateur modifiable.

Afin de créer un compte avec un mot de passe haché et un role d'administrateur j'ai utilisé les classes Seeder proposé par Laravel qui me permettent de remplir la base de données avec des valeurs par défauts

Avec les commandes artisan *Php artisan make:seeder {nom du fichier}* je crée donc 2 fichier Seeder

RolesSeeder

Cette classe insérera les rôles nécessaire de mon application

```
<?php

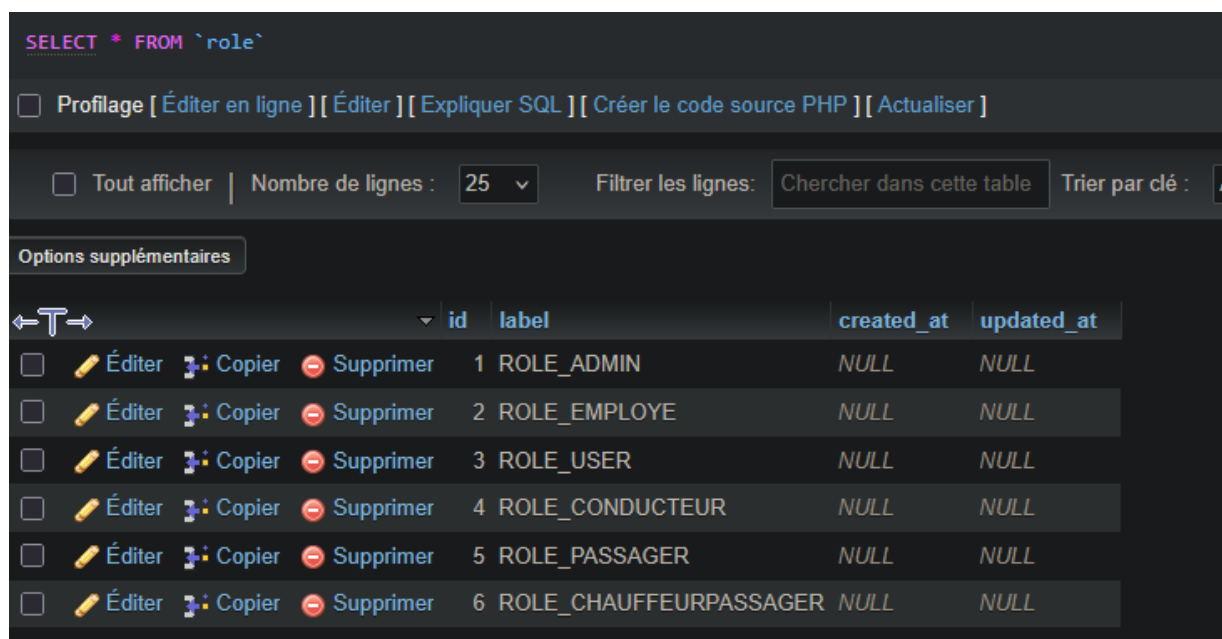
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

0 references | 0 implementations | You, 4 days ago | 1 author (You)
class RolesSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        // Insère des rôles dans la table 'roles'
        DB::table('role')->insert(values: [
            ['label' => 'ROLE_ADMIN'],
            ['label' => 'ROLE_EMPLOYE'],
            ['label' => 'ROLE_USER'],
            ['label' => 'ROLE_CONDUCTEUR'],
            ['label' => 'ROLE_PASSAGER'],
            ['label' => 'ROLE_CHAUFFEURPASSAGER'],
        ]);
    }
}
```

Ma fonction cherchera la table et remplira les colonnes « label »
qui sont les définitions des rôles.

Ensuite, j'exécute la commande `php artisan db:seed --class=RolesSeeder` pour remplir la table
En regardant sur PhpmyAdmin, l'insertion est réussie



The screenshot shows the phpMyAdmin interface for a database. At the top, the SQL query `SELECT * FROM `role`` is entered. Below the query, there are buttons for 'Profilage', 'Éditer en ligne', 'Éditer', 'Expliquer SQL', 'Créer le code source PHP', and 'Actualiser'. A toolbar shows 'Tout afficher', 'Nombre de lignes: 25', 'Filtrer les lignes: Chercher dans cette table', and 'Trier par clé: A'. An 'Options supplémentaires' button is also present. The main table displays the data for the 'role' table, with columns for 'id', 'label', 'created_at', and 'updated_at'. Each row includes action buttons for 'Éditer', 'Copier', and 'Supprimer'.

		id	label	created_at	updated_at
<input type="checkbox"/>	Éditer Copier Supprimer	1	ROLE_ADMIN	NULL	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	2	ROLE_EMPLOYE	NULL	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	3	ROLE_USER	NULL	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	4	ROLE_CONDUCTEUR	NULL	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	5	ROLE_PASSAGER	NULL	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	6	ROLE_CHAUFFEURPASSAGER	NULL	NULL

AdminUserSeeder

Maintenant que j'ai des rôles définis, on peut créer l'utilisateur Administrateur

Je refais les mêmes commandes et je construis le seeders comme ceci


```

<?php

namespace Database\Seeders;

use App\Models\User;
use App\Models\Role;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

0 references | 0 implementations | ...
class AdminUserSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $adminRole = Role::where('label', 'ROLE_ADMIN')->first();

        User::updateOrCreate([
            'email' => 'admin@ecoride.com',
        ], [
            'pseudo' => 'Admin',
            'password' => Hash::make(value: 'admin'),
            'role_id' => $adminRole->id,
        ]);
    }
}

```

En plus d'inscrire les valeurs des colonnes je crée la ligne *\$adminRole*

Afin qu'il récupère la valeur spécifique dans la table Role ainsi que la méthode *Hash::make()* pour crypter le mot de passe.

Après la commande d'insertion, on remarque que la table récupère bien la table role avec l'identifiant *ROLE_ADMIN* avec le mot de passe haché

	id	pseudo	image	email	password	credit	role_id	created_at	updated_at
<div><div><div></div><div></div><div></div></div><div><div>Éditer</div><div>Copier</div><div>Supprimer</div></div></div>	2	Admin	NULL	admin@ecoride.com	\$2y\$12\$IKsh1bUd6nSKP7Uc6ZNa.nvJfG4PrtmBJFSazhl6YE...	0	1	2025-04-13 08:56:18	2025-04-13 08:56:18

Docker

L'implémentation de Docker permet la configuration optimal et rapide de l'environnement de travail.

Cela garantit que si un nouveau développeur ou que je doivent changer de machine, de pouvoir travailler sur le même standard de développement.

Configuration

Voici les fichier nécessaire pour l'utilisation de Docker

Dockerfile

```
# Utiliser une image officielle PHP avec Apache
FROM php:8.2-apache

# Installer les extensions PHP nécessaires
RUN apt-get update && apt-get install -y \
    libicu-dev libpq-dev libzip-dev unzip git \
    && docker-php-ext-install intl pdo pdo_mysql zip

# Activer mod_rewrite pour Laravel
RUN a2enmod rewrite

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

# Définir le répertoire de travail
WORKDIR /var/www/html

# Copier les fichiers du projet
COPY . .

# Installer les dépendances PHP
RUN composer install --no-dev --no-interaction --optimize-autoloader

# Définir les permissions pour Laravel
RUN chown -R www-data:www-data /var/www/html/storage /var/www/html/bootstrap/cache

# Exposer le port 80
EXPOSE 80

CMD ["apache2-foreground"]
```

Ce fichier va installer les dépendances nécessaires du projet
Docker-composer.yml

```

docker-compose.yml The Compose specification establishes a standard for the definition
version: '3.8'

Run All Services
services:
  Run Service
  app:
    build: .
    container_name: laravel_app
    restart: always
    volumes:
      - ../var/www/html
    ports:
      - "8080:80"
    depends_on:
      - mysql
      - mongodb
    environment:
      APP_ENV: local
      DB_CONNECTION: mysql
      DB_HOST: mysql
      DB_PORT: 3306
      DB_DATABASE: laravel_db
      DB_USERNAME: root
      DB_PASSWORD: root
      MONGO_URL: mongodb://mongodb:27017/laravel_mongo

  Run Service
  mysql:
    image: mysql:8
    container_name: laravel_mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: laravel_db
    ports:
      - "3307:3306"
    volumes:
      - mysql_data:/var/lib/mysql

```

You, 2 months ago • configuration de docker et ajouts d

```

Run Service
mongodb:
  image: mongo:6
  container_name: laravel_mongo
  restart: always
  ports:
    - "27018:27017"
  volumes:
    - mongo_data:/data/db

volumes:
  mysql_data:
  mongo_data:

```

Le fichier me permet d'exécuté l'ensemble des services nécessaires comme les bases de données comme ici les bases de données en local.

Veille technologique

Pour garantir le bon fonctionnement de mon application, j'ai consulté plusieurs ressources dans le but :

- d'assurer la sécurité en identifiant les éventuelles vulnérabilités,
- d'optimiser l'application en mettant régulièrement Laravel à jour,
- de suivre l'évolution des différents outils.

Parmi ces ressources :

- La documentation de Laravel (<https://laravel.com/docs/11.x>)
- Le serveur Discord de Laravel, qui offre l'aide de la communauté

Grâce à ces informations, je peux mieux anticiper et prévenir les vulnérabilités potentielles.

Veille de sécurité

Outil Laravel

Heureusement, Laravel offre des outils pour garder une application sécurisée.

Injection Sql

Injection SQL

L'injection SQL est une attaque qui permet de détourner certaines fonctionnalités du site afin d'insérer du code SQL malveillant. Cette technique peut, par exemple, permettre de modifier des données sensibles de la base de données, comme les mots de passe des utilisateurs.

Solution

Pour prévenir ce risque, il est indispensable d'éviter l'écriture d'injections brutes dans le code. Dans ce projet, j'utilise deux outils pour la gestion de la base de données, afin de rendre les requêtes plus sécurisées :

1. Eloquent

Eloquent est adapté pour gérer les relations entre entités et propose une structure prédéfinie. Par exemple, pour créer le modèle User, on utilise la commande :

```
Php artisan make:model User
```

Dans ce modèle, la propriété \$fillable détermine quels champs peuvent être massivement assignés (mass assignent), évitant ainsi qu'un utilisateur malveillant ne modifie des données non autorisées.

```

<?php
namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Spatie\Permission\Traits\HasRoles;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use App\Models\Role;

class User extends Authenticatable
{
    /** @use HasFactory<Database\Factories\UserFactory> */
    use HasFactory, Notifiable, HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'id',
        'image',
        'pseudo',
        'email',
        'password',
        'credit'
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
}

```

2. Query Builder

Pour exécuter des requêtes, l'utilisation du Query Builder permet d'insérer et de manipuler des données à l'aide d'une syntaxe fluide et expressive, sans recourir directement au SQL brut.

Exemple

Dans un seeder, on peut injecter les noms des rôles dans la table correspondante. On importe la façade DB pour exécuter les requêtes en toute sécurité.

```

<?php
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class RolesSeeder extends Seeder
{
    public function run(): void
    {
        // Insère des rôles dans la table 'roles'
        DB::table('role')->insert([
            ['label' => 'ROLE_ADMIN'],
            ['label' => 'ROLE_EMPLOYE'],
            ['label' => 'ROLE_USER'],
            ['label' => 'ROLE_CONDUCTEUR'],
            ['label' => 'ROLE_PASSAGER'],
            ['label' => 'ROLE_CHAUFFEURPASSAGER'],
        ]);
    }
}

```

Conclusion :

Ces deux outils (Eloquent et Query Builder) s'utilisent conjointement, chacun ayant ses avantages pour garantir des requêtes sécurisées et limiter au maximum les risques d'injection SQL. [Cross-site Request Forgery \(CSRF\)](#)

Le CSRF (Cross-Site Request Forgery) est une attaque dans laquelle un attaquant peut usurper l'identité d'un utilisateur authentifié pour envoyer des requêtes malveillantes.

Solution

Pour se protéger de ce type d'attaque, il faut utiliser le système CSRF intégré à Laravel :

1. Activation du middleware

- Laravel dispose d'un middleware dédié au CSRF, situé dans app/Http/Kernel.php.
- Le groupe de middleware correspondant assure automatiquement la protection CSRF pour toutes les requêtes définies.

2. Protection des formulaires

```
6      You, last month * ajout des fonctionnalités, accueil, contact ave...
7      <form class="custom-form" action="{{ route('contact.send') }}" method="POST">
8          @csrf
9          <input class="form-input" type="email" name="email" placeholder="email" value="{{ old('name') }}">
10         @error('email') <p style="color: red; "{{ $message }}"</p> @enderror
11         @csrf
12         <input class="form-input" type="text" name="titre" placeholder="titre" value="{{ old('titre') }}">
13         @error('titre') <p style="color: red; "{{ $message }}"</p> @enderror
14         @csrf
15         <textarea class="form-input" name="message" placeholder="message">{{ old('message') }}</textarea>
16         @error('message') <p style="color: red; "{{ $message }}"</p> @enderror
17
18         <button class="form-button" type="submit">Envoyer</button>
19     </form>
20
```

- Les formulaires de l'application doivent inclure la directive @csrf au sein de la balise <form>.
- Cette directive génère un jeton unique par session et vérifie sa validité lors de l'envoi du formulaire.

Grâce à cette configuration, Laravel protège efficacement les utilisateurs contre les tentatives de falsification de requêtes.

3. Sécurisé des mots de passes et accès

Dans l'Union européenne, la protection des données est encadrée par le RGPD (Règlement général sur la protection des données). Le stockage des mots de passe doit donc être réalisé de manière sécurisée.

Solution

1. Entité User

- Sur Laravel, il est possible de générer une entité User intégrant la gestion des utilisateurs via le kit **Laravel Breeze**, qui assure la sécurité des données.
- La commande suivante crée le modèle et sa migration :

Php artisan make:model User -m

- La structure générée inclut des propriétés de base telles que :
 - Email et pseudo, pour identifier l'utilisateur,
 - password, qui stocke le mot de passe de façon hachée pour plus de sécurité,

- role, afin de définir les droits d'accès de l'utilisateur.

2. Gestion des comptes utilisateurs

- Laravel Breeze fournit par défaut un contrôleur qui gère la création des utilisateurs et leur authentification.
- Lors de la création d'un utilisateur, la ligne suivante permet de chiffrer le mot de passe :

Php

Copier

'password' => Hash::make(\$Request->password),

- Ainsi, le mot de passe est haché et reste illisible dans la base de données.

```
// Créer un nouvel utilisateur
$user = User::create(attributes: [
    'name' => $request->pseudo,
    'email' => $request->email,
    'password' => Hash::make(value: $request->password),
]);
```

3. Sécurisation des pages et rôles

- L'authentification des utilisateurs est gérée par le fichier /config/auth.php, qui définit les mécanismes et permissions d'accès.
- En associant des middlewares d'authentification (auth) et de rôles (role) dans le fichier de routing, il est possible de restreindre l'accès à certaines routes aux seuls utilisateurs

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DashboardAdminController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\Auth\RegisteredEmployeeController;

Route::middleware(['auth', 'role:ROLE_ADMIN'])->group(function () { void {

    Route::get(uri: 'admin/userCreation/', action: [ProfileController::class, 'index'])->name(name: 'admin.userCreation.index');
    Route::get(uri: 'admin/userCreation/create', action: [RegisteredEmployeeController::class, 'createEmployee'])->name(name: 'admin.userCreation.create');
    Route::post(uri: 'admin/userCreation/create', action: [RegisteredEmployeeController::class, 'storeEmployee'])->name(name: 'admin.userCreation.store');
    Route::get(uri: 'admin/userCreation/{id}/edit', action: [ProfileController::class, 'edit'])->name(name: 'admin.userCreation.edit');
    Route::delete(uri: 'admin/userCreation/{id}', action: [ProfileController::class, 'destroy'])->name(name: 'admin.userCreation.destroy');
```

connectés et possédant le rôle requis.

- Par exemple, on peut limiter l'accès à l'interface de gestion des utilisateurs à l'administrateur de l'application.

Déploiement

Pour que mon site web soit utilisable par tous, je me suis tourné vers Heroku ayant une structure et plusieurs option.

Avantage	Inconvénient
<ul style="list-style-type: none">• Déploiement depuis le repository sur GitHub• Compatible avec les technologies du projet• Add-on pour les différents services• Gestion de l'environnement• Double identification pour accéder à l'Hebergeur	<ul style="list-style-type: none">• Stockages éphémère• Dépendance au cloud• Demande un investissement surtout si l'application gagne en clientèle

Quantité	Nom	Prix
1	Heroku Dyno - Projet Laravel	7
1	JawDb MySql plan Leopard Shared	10

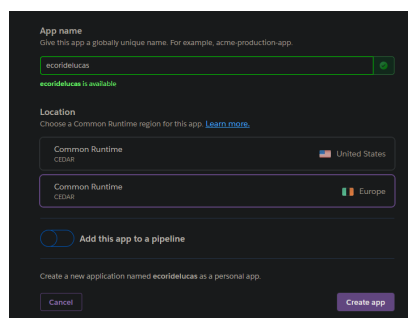
Estimation des coûts

17 euros par mois

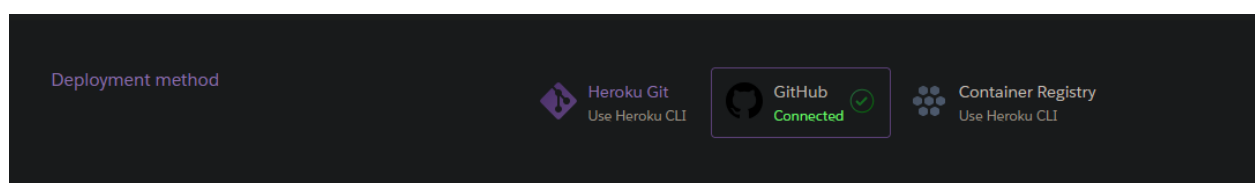
Processus et configuration de déploiement

Dans mon dashboard Heroku, je peux créer un nouveau projet dans l'onglet « new app »

La page de creation me permet d'entre le nom du projet, sa localisation (America ou Europe).



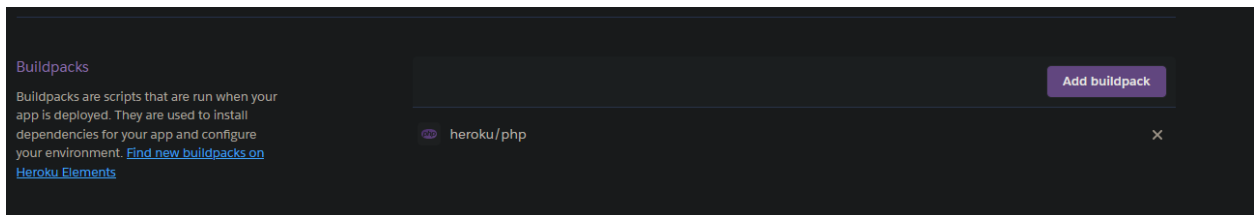
Ensuite, le projet Heroku, créée, je peux importer le repository Github du projet directement.



Avant de publier mon projet, je configure les informations de l'application

Buildpack

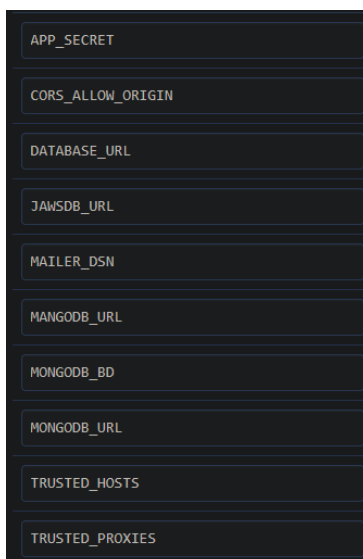
Permet de configurer des scripts de dépendance par rapport aux technologies utilisées (ici PHP via Laravel)



Config Vars

C'est ici que j'enregistre les différentes informations sensibles du fichier .env, pour que mon projet puisse récupérer les données de cette espace, je configure le fichier .env comme ceci.

Config Vars



fichier .env

```
###> doctrine/doctrine-bundle ###  
# Connexion à la base de données  
DATABASE_URL=${DATABASE_URL}
```

Le fichier .env ira rechercher les données stocké dans le Config Vars de Heroku

Conclusion

La réalisation de ce projet constitue le second travail de développeur, Ma volonté était de travailler avec un nouveau Framework afin de diversifier les technologies utilisé et gagner en polyvalence.

Le travail et les lacunes de mon ancien projet m'ont permis de gérer la conception plus aisément en configurant mon environnement d'une meilleur manière surtout la configuration du git.

Utiliser Laravel m'a appris à m'adapter sur une nouvelle technologie...

Ma prochaine volonté serait de travailler en groupe sur des projets plus ambitieux afin d'apprendre dans un milieu professionnel et pouvoir avoir les tâches réparties.

Annexe

GitHub du projet : <https://github.com/LucasCherbuin/Ecoride>

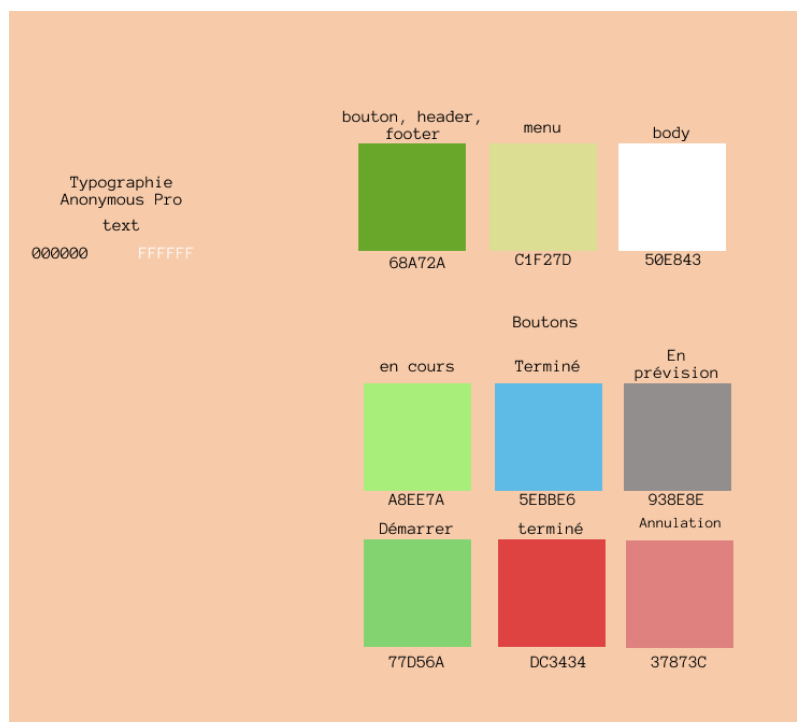
Lien du projet en ligne : <https://ecoridegecko-02c5182522d3.herokuapp.com>

Vous pouvez tester les comptes suivants

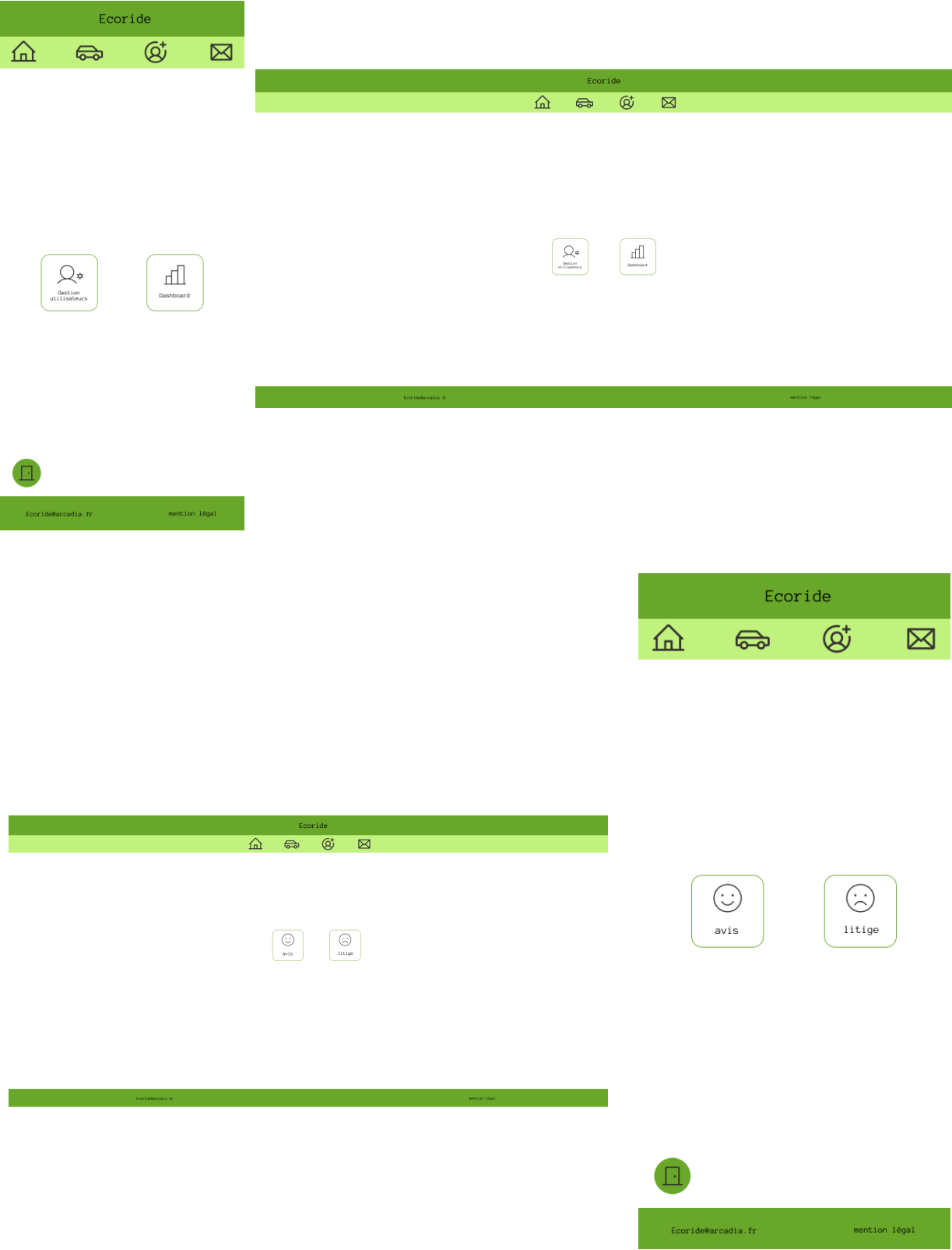
Email	Rôle	Mot de passe
admin@ecoride.fr	ROLE_ADMIN	admin
employe@ecoride.fr	ROLE_EMPLOYE	employe
conducteur@ecoride.fr	ROLE_CONDUCTEUR	conducteur
passager@ecoride.fr	ROLE_PASSAGER	passager
ChauffeurPassager@ecoride.fr	ROLE_CHAUFFEURPASSAGER	chauffeurPassager

Maquettes design

Charte graphique



Menus utilisateurs





20

20



Pages covoiturages

Ecoride



Choix de l'itinéraire

Marseille

Paris

08/03/25

Rechercher

Filtre

énergie propre

Prix max 5 \$

Durée max 3 h

Note 3

résultat

4.0

jean

reserv

er

1 février 9h30

trajet

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

Ecoride@arcadia.fr

mention légale

Ecoride



Choix de l'itinéraire

Marseille

Paris

08/03/25

Rechercher

Filtre

énergie propre

Prix max 5 \$

Durée max 3 h

Note 3

résultat

4.0

jean

reserv

er

1 février 9h30

trajet

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

Ecoride@arcadia.fr

mention légale

4.0

jean

reserv

er

1 février 9h30

Lyon Marseille

3.99 \$ 3

no smoking

no pets

Numéro plaques

VD213000

détail

ne pas boire dans le véhicule

véhicule utilisé

Marque

Tesla

Couleur

Blanche

Modèle

Modele Y

Energie

electrique

avis

Louis

Très poli et compréhensif

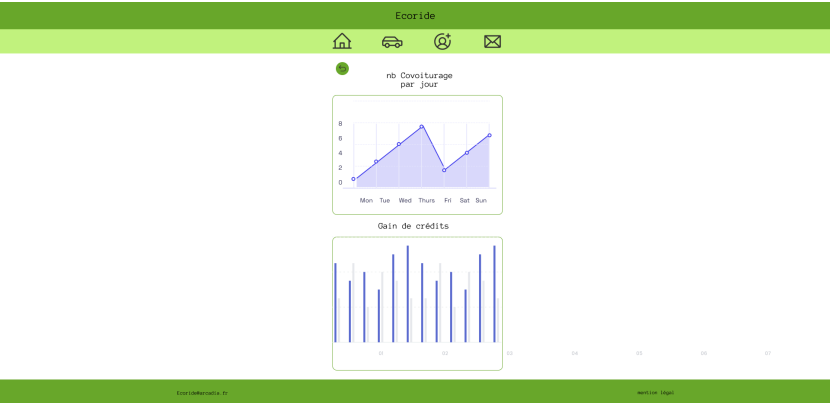
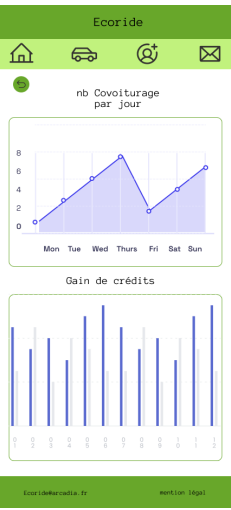
Confirmer ?

Inscrivez ou Connectez vous pour reserver

Inscription

Connexion

Dashboard



Resultat responsive

