ECF - GARAGE VINCENT PARROT

Document technique

Garage V. Parrot

Technologies frontend

1 - REACT

Ayant une nette préférence pour l'environnement Node.js, je me suis tourné naturellement vers REACT pour développer ce projet.
J'ai choisi cette technologie pour différentes raisons :

- Les composants réutilisables qui permettent de mieux structurer son code et gagner en temps en développement
- Le support de la communauté, mes références ont été principalement Stackoverflow, Reddit et Discord. La communauté de REACT est très active et l'évolution est constante et grandissante
- L'écosystème, les innombrables librairies de REACT offrent des outils adaptés à toute sorte de projets.

2 - Tailwind

Tailwind est un framework CSS basé sur les classes, il utilise des classes prédéfinies pour styliser des éléments HTML.
Il possède une documentation complète et bien organisée qui fournit des templates et des explications détaillées.
De plus, il s'intègre très facilement à REACT.

Pour l'utiliser il suffit de modifier les classes directement dans les rendus, c'est aussi une des ses faiblesses car il fait perdre en lisibilité le code dans lequel il est intégré.

3 - Cloudinary

Est un hébergeur de média qui propose une offre gratuite et un support réactif et à l'écoute.
L'intégration est facile d'utilisation grâce à la documentation, les tutoriels mis à disposition sur leur site et leur chaîne Youtube.
Il propose des bibliothèques et SDK pour différents langages de programmation et framework, il a d'ailleurs un module pour REACT.

4 - Les dépendances

a - Axios

Bibliothèque très populaire pour effectuer des requêtes HTTP, il possède une syntaxe concise pour l'utilisation du CRUD avec GET, POST, DELETE, UPDATE.
Il est pris en charge à la fois en frontend et en backend, idéal pour les projets fullstack.
On peut bien entendu l'utiliser pour des requêtes asynchrones pour une expérience utilisateur plus fluide.

b - Cookie Parser

Cookie Parser va permettre de traiter les cookies dans l'application.
Il va gérer automatiquement l'extraction des cookies à partir des en-têtes HTTP.
Je l'utiliserai aussi pour la gestion de sécurité notamment avec les indicateurs "secure", "httpOnly" et "sameSite".

c - CORS

Je vais l'utiliser pour gérer les politiques de partage des ressources entre les domaines client et serveur.
Concrètement, il sera mis en place dans les requêtes autorisées, ce qui aidera à prévenir des attaques XSS et CSRF.

d - DomPurify

Va nous servir à nettoyer et sécuriser les inputs HTML provenant de sources non fiables telles que, les entrées utilisateurs en supprimant les balises, attributs potentiellement dangereux.
Même si ce module est reconnu pour son efficacité, il ne sera pas suffisant pour permettre une sécurité totale du site, il faudra mettre en place d'autres dispositifs comme la validation des entrées utilisateur.

e - Google Recaptcha

C'est une protection contre les robots et les spams qui bloquera les requêtes automatisées, réduisant considérablement les activités malveillantes.
Pour l'utiliser il suffira simplement à l'utilisateur de cocher une case avant d'envoyer un formulaire.

Développé et maintenu par Google c’est un outil fiable pour limiter les risques de spams.

Technologies Backend

1 - Express.js

C’est un framework très populaire pour Node.Js qui sert à développer la partie backend d’une application.
Il propose une interface intuitive qui correspond à mes attentes de développeur débutant offrant une courbe d’apprentissage plus douce.
Son système de middleware est facile à maintenir et permet de créer des tâches comme de l’authentification, requêtes CRUD etc…
Il a aussi des performances élevées notamment grâce à sa gestion de requêtes asynchrones.

2 - MYSQL

Ce langage SQL est reconnu pour sa fiabilité et sa stabilité, parfaitement adapté pour le stockage et la gestion de données.
Il utilise des techniques d'optimisation (mise en cache, indexation etc..) pour traiter de grandes quantités de données ce qui le rend très performant.
Mysql conviendra parfaitement à notre environnement de travail, il suffira d’installer le module adéquat dans le backend de notre application et de configurer la connexion.

3 - Les dépendances

a - Bcryptjs

Bcryptjs utilise l'algorithme Bcrypt que nous n’utiliserons pas dans ce projet par souci de compatibilité avec l’environnement de Vercel.
Sa technique de hachage est extrêmement sécurisée pour le stockage de données, il utilise un processus relativement lent et coûteux qui rend les attaques par force brute difficiles à réaliser.
Sa méthode de “salage” ajoute une couche de sécurité en plus en générant un sel unique pour chaque mot de passe.

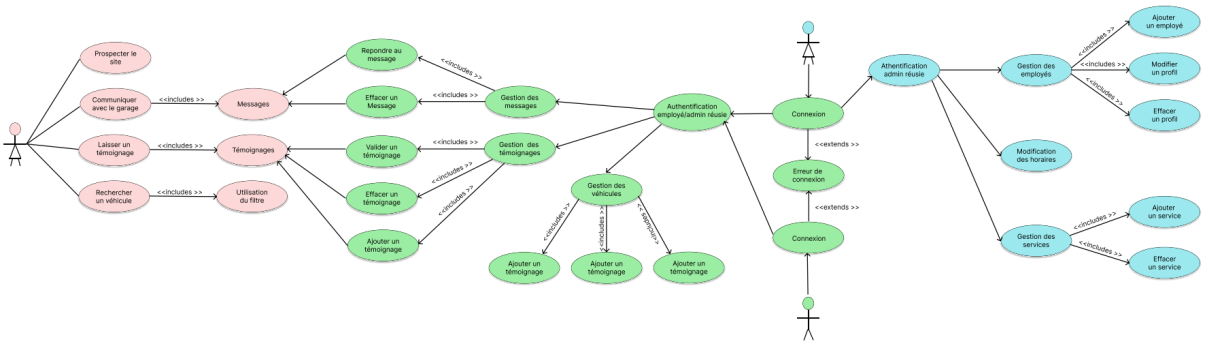
b - Dompurify

Comme dans la partie frontend on utilisera Dompurify pour assainir les données utilisateurs, ce n'est pas simplement une double sécurité qui ralentit inutilement le code mais une réelle protection dans le cas d'une injection par middleware non autorisée.

c - Dotenv

Technologie utile pour récupérer des données dans des variables d'environnements, il est utilisé aussi en frontend mais il n'aura pas besoin d'installation car il est intégré à REACT.

Diagramme de cas d'utilisation



**Une version pdf plus détaillée est fournie à la racine du repository ou sur [Figma]*

Dans ce diagramme nous avons 3 acteurs;

- Le visiteur : qui pourra prospecter toute la partie public, laisser un message via les pages *Occasions* et *Contact,* ajouter un commentaire et faire des recherches de véhicules.

- L'employé : son rôle lui donnera accès au tableau de bord dans lequel il pourra ;

    - Gérer les messages client (répondre ou effacer)
        - *La fonctionnalité 'Répondre' est toujours en cours de développement*
    - Gérer les témoignages (valider, effacer ou ajouter)

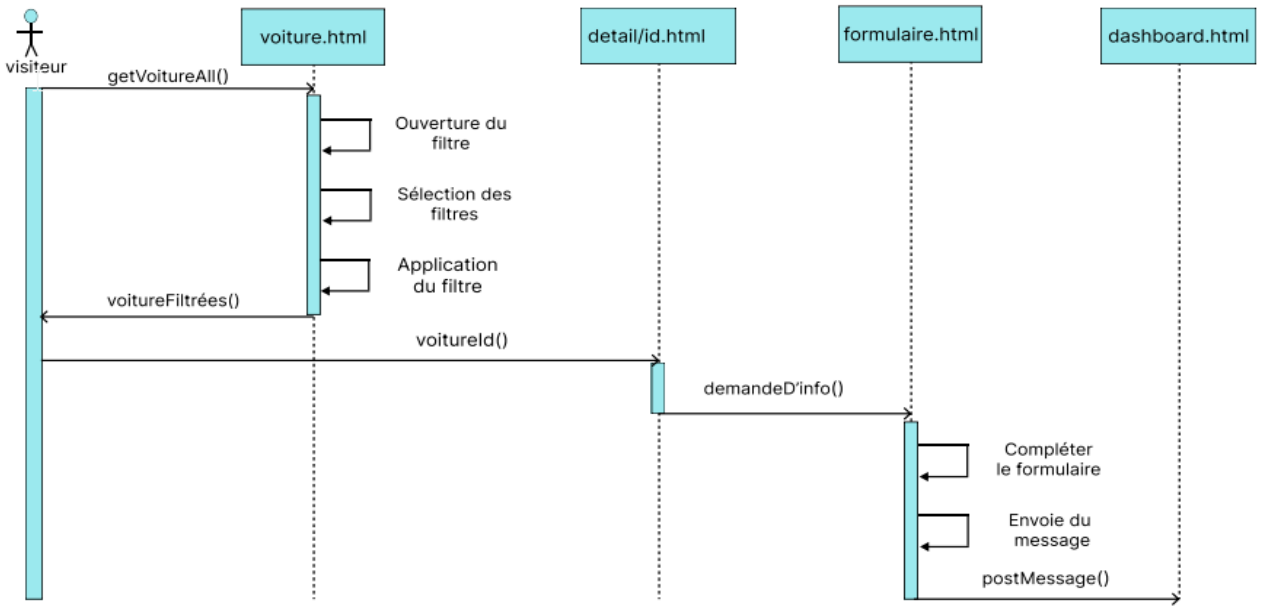
- Gérer les véhicules (ajouter, éditer ou supprimer)
- L'administrateur : aura les mêmes droits que l'employé ainsi que d'autres privilèges ;

- Gérer les employés (ajouter, modifier ou supprimer)
- Modifier les horaires d'ouverture/fermeture du garage
- Editer les services (ajouter ou supprimer)

Diagramme de séquence

Contexte

Décrivez grâce à ce schéma le processus d'un nouveau visiteur qui:
- découvre la liste des véhicules d'occasion
- filtre selon ses critères
- rempli le formulaire pour en savoir plus à propos d'une voiture en particulier

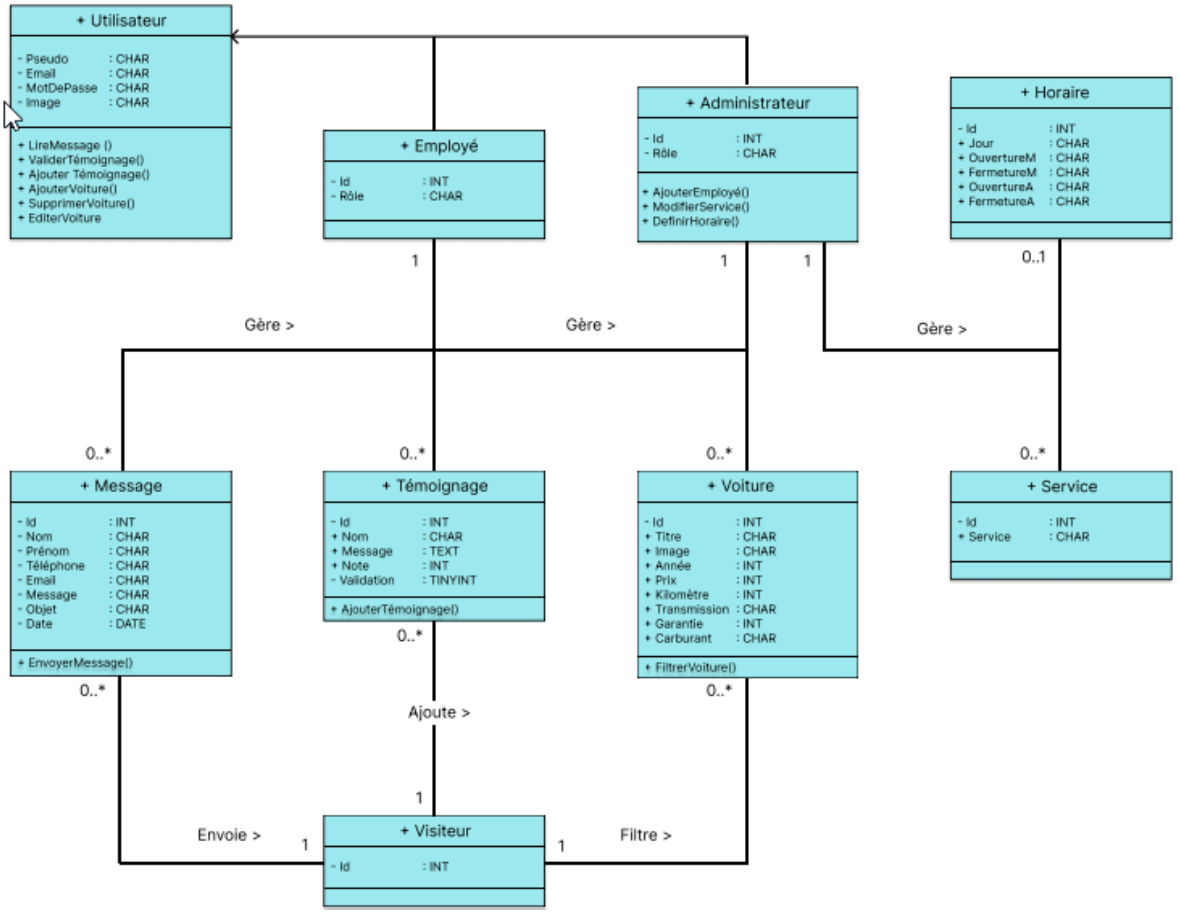


Description de la séquence

1. Le visiteur ouvre la page des Occasions (getVoitureAll())
2. Il ouvre le filtre en cliquant sur un bouton
3. Sélectionne les filtres désirés

4. Applique le filtre en cliquant sur un bouton
5. Les voitures filtrées lui sont retournées (filteredCars())
6. Il clique sur une voiture en particulier (carId())
7. Il décide d'envoyer un message via le formulaire de la page
8. Le visiteur rempli le formulaire
9. Et clique sur le bouton pour envoyer le formulaire
10. Le formulaire est envoyé dans la BDD (postMessage())

Diagramme de séquence



**Une version pdf plus détaillée est fournie à la racine du repository ou sur [Figma]*

Description du diagramme de classes

Ici on peut reprendre les fonctionnement du diagramme de cas d'utilisation et garder la même cohérence avec les actions et fonctions possibles.

Concernant les cardinalité voici comment j'ai visualisé les actions possibles.

Le visiteur

- 1 visiteur a le choix d'envoyer 0 ou plusieurs messages.

- 1 visiteur a le choix de filtrer 0 ou plusieurs véhicules.

- 1 visiteur a le choix d'ajouter 0 ou plusieurs témoignages.

Important
Ici il faudrait mettre un dispositif pour limiter un témoignage par personne, solution possible:

- Récupérer l'IP de la personne et la stocker dans la base de données, celle-ci étant personnelle, il faudra conséquemment se soumettre au RGPD Européennes.
  De plus cette méthode n'est pas fiable à 100%, notamment si l'utilisateur utilise un VPN.

- Créer un cookie visiteur pour limiter les abus, là encore, même si le cookie semble être essentiel, il faudra respecter le ePrivacy Directives et le RGPD.
  Cette solution n'est pas complètement effective non plus, il suffira d'effacer le cookie pour laisser un autre témoignage.

L'employé

- 1 employé pourra gérer 0 (il pourra juste consulter le contenu du dashboard) ou plusieurs Messages, Témoignages ou Voitures.
-

L'administrateur aura les même droits que l'employé et :

- 1 administrateur gérera 0 (consulter) ou 1 formulaire d'horaire
- 1 administrateur gérera 0 ou plusieurs services