GARAGE PARROT

PARTIE FRONT END



Compte rendu : Développement Front-End

Ce rapport décrit les principales étapes et réalisations effectuées lors du développement Front-End, en mettant particulièrement l'accent sur la mise en place de l'environnement de travail, l'utilisation de React, la gestion des dossiers, et l'intégration de Bootstrap via les CDN.

**Mise en Place de l'Environnement de Travail :**

- L'une des premières tâches du projet a été la mise en place de l'environnement de travail.

- Mon choix s'est porté sur la bibliothèque JavaScript React pour le développement des interfaces utilisateur.

- Il a été nécessaire de configurer les outils et dépendances nécessaires pour pouvoir démarrer mon projet.

Gestion des Dossiers et Problèmes d'Importation :

- Une des premières difficultés rencontrées a été liée à la gestion des dossiers et à l'importation de composants.

- Il a été observé que React ne reconnaissait pas automatiquement les majuscules dans les noms de dossiers, obligeant à les saisir manuellement.

- Ce problème a également été rencontré précédemment sur un autre projet.

\*\*Intégration de Bootstrap via les CDN :\*\*

- Pour améliorer l'aspect visuel de l'application, Bootstrap a été intégré en utilisant des CDN (Content Delivery Network).

- Cette approche a permis d'inclure rapidement Bootstrap dans le projet sans avoir à télécharger et à configurer les fichiers localement.

\*\*Création de Composants SF et SL :\*\*

- Deux types de composants ont été créés : les composants Stateless Function (SF) et les composants Stateful (SL).

- Les composants SF sont des composants React simples qui ne gèrent pas d'état interne.

- Les composants SL, en revanche, gèrent un état interne et sont utilisés pour des éléments interactifs de l'interface utilisateur.

\*\*Optimisation du Flux de Travail avec des Snippets :\*\*

- Pour faciliter le développement, des extraits de code (snippets) ont été créés dans Visual Studio Code.

- Ces extraits permettent de générer automatiquement la structure de base des composants SF et SL, accélérant ainsi le processus de création de composants.

Pour utiliser ces extraits, il suffit d'aller dans le menu "Fichier" (File), puis "Préférences" (Preferences), et enfin "Extraits d'utilisateur" (User Snippets), puis de sélectionner le langage (dans ce cas JavaScript).

En résumé, le stage de développement Front-End a permis de mettre en place un environnement de travail React, de résoudre des problèmes liés à la gestion des dossiers, d'intégrer Bootstrap via des CDN, de créer des composants SF et SL, et d'optimiser le flux de travail grâce à l'utilisation de snippets dans Visual Studio Code. Ce stage a offert une expérience précieuse dans le développement d'applications web modernes.

"import React & creer un composant stateless": {

        "prefix": "imp-r-sl",// (import r stateless) A definir soit même, on peut mettre le nom que l on veut afin qu'il soit bien repérable pour pouvoir l utiliser au plus vite

        "body": [

            "import React from \"react\"; \n",//\n = sauter une ligne

            "const ${component} = (props) => (",//${component} = permet de nommer plus facilement

            "\t//Ecrire le code ici", //\t = tabulation = permet au code d'être bien rangé

    ");\n",

        "export default ${component};"

        ],

        "description": "Import React & creer un composant stateless"

    },

    "import React & creer un composant statefull": {

        "prefix": "imp-r-sf",// (import r statefull) A definir soit même, on peut mettre le nom que l on veut afin qu'il soit bien repérable pour pouvoir l utiliser au plus vite

        "body": [

            "import React, { Component } from 'react';\n",

                "class  ${component} extends Component{",

                "\trender() {",

                    "\t\treturn (",

                    "\t\t\t//Ecrire le code ici",

                "\t\t);",

                "\t}",

            "}\n",

                "export default ${component};",

        ],

        "description": "Import React & creer un composant statefull"

    }

}

\*\*Rapport de Stage : Développement Front-End (Suite)\*\*

Ce rapport poursuit la description des tâches réalisées pendant le stage de développement Front-End, en se concentrant sur la création du composant Navbar avec Bootstrap, la mise en place d'une page d'erreur 404, l'utilisation d'Axios pour les requêtes, la création du composant de cartes pour les véhicules, la mise en place d'un système de pagination, et le développement d'un système de recherche de véhicules par filtre.

\*\*Création du Composant Navbar avec Bootstrap :\*\*

- Un composant Stateless nommé Navbar.js a été créé pour afficher la barre de navigation de l'application.

- Des images et du code JSX ont été insérés dans le composant, avec des classes `className` pour le style.

- Les services de réparation ont été présentés sous forme de menus déroulants (dropdowns) dans la barre de navigation.

\*\*Création de la Page d'Erreur 404 :\*\*

- Une route de type Error 404 a été créée pour gérer les cas où une page n'existe pas.

- Cette page d'erreur fournira une réponse conviviale aux utilisateurs en cas d'accès à des pages inexistantes.

\*\*Utilisation d'Axios pour les Requêtes :\*\*

- Axios, une bibliothèque JavaScript populaire pour effectuer des requêtes HTTP, a été installé à l'aide de la commande `npm install axios`.

- Axios sera utilisé pour effectuer des requêtes vers le backend ou d'autres services.

\*\*Création du Composant Card pour les Véhicules :\*\*

- Un composant de carte a été développé pour afficher les informations des véhicules.

- Ce composant sera connecté à la base de données à l'aide d'Axios pour afficher les détails des véhicules.

\*\*Mise en Place de la Pagination :\*\*

- Un système de pagination a été mis en place pour limiter le nombre de véhicules affichés par page.

- Le résultat maximal affiché est de 12 véhicules par page, améliorant ainsi la performance de l'application.

\*\*Système de Recherche de Véhicules par Filtre :\*\*

- Un système de filtrage en temps réel a été ajouté au-dessus des curseurs pour les filtres d'année, de kilométrage et de prix.

- Un dropdown a été mis en place pour la sélection de la marque du véhicule, facilitant ainsi la recherche personnalisée.

En résumé, le stage de développement Front-End a vu la création du composant Navbar avec Bootstrap, la mise en place d'une page d'erreur 404, l'utilisation d'Axios pour les requêtes, le développement du composant de cartes pour les véhicules, l'intégration d'un système de pagination, et la réalisation d'un système de recherche de véhicules par filtre. Ces réalisations contribuent à améliorer l'expérience utilisateur et à rendre l'application plus fonctionnelle.

\*\*Rapport de Stage : Utilisation de React Router et Mise en Place de Composants Additionnels (Suite)\*\*

Dans la continuité du stage, j'ai entrepris la mise en place de la gestion des routes à l'aide de React Router, l'ajout de composants pour l'accueil, la création d'un formulaire de contact, et l'intégration d'un Captcha Google pour le formulaire.

\*\*React Router :\*\*

- React Router a été installé à l'aide de la commande `npm install react-router-dom`.

- Une fois installé, il a été importé dans le fichier `App.js` pour être utilisé dans la gestion des routes de l'application.

- Le composant `Site` a été configuré pour contenir toutes les routes de l'application.

\*\*Page d'Accueil :\*\*

- La page d'accueil a été mise en place avec l'ajout d'un composant `h1` pour afficher le titre.

- Pour améliorer la présentation, la bibliothèque Font Awesome a été installée à l'aide de la commande `npm install --save font-awesome`.

- Un composant de carrousel a été utilisé pour afficher les prestations du garage sous forme de cartes avec des boutons interactifs.

\*\*Formulaire de Contact :\*\*

- Une route vers le formulaire de contact a été créée dans le fichier `Site.js`, avec son composant associé.

- Le composant `Formulaire` a été développé pour gérer le formulaire de contact.

- La bibliothèque Formik a été utilisée pour faciliter la gestion des formulaires et la bibliothèque Yup pour imposer des restrictions lors du remplissage des champs du formulaire.

\*\*Ajout de Google reCAPTCHA :\*\*

- Pour améliorer la sécurité du formulaire de contact, un reCAPTCHA de Google a été intégré.

- L'installation du reCAPTCHA a été réalisée à l'aide de la commande `npm install react-google-recaptcha`.

- Un ID unique a été créé pour le reCAPTCHA.

Ces développements contribuent à enrichir l'application en ajoutant des fonctionnalités essentielles telles que la gestion des routes avec React Router, une page d'accueil attrayante avec un carrousel, et un formulaire de contact sécurisé grâce à l'intégration de Google reCAPTCHA. Ces fonctionnalités améliorent l'expérience utilisateur et renforcent la sécurité de l'application.

**Page Voitures d’Occasions**

**Mise en Place des Composants de Recherche et de Filtrage**

Au cours de cette phase de développement, j'ai conçu des éléments d'interface utilisateur essentiels, notamment des sliders, des sélecteurs (dropdowns), et des cases à cocher (checkboxs), visant à faciliter la recherche et le filtrage des données.

Pour améliorer l'expérience utilisateur, je vais créer des sliders prix, km et année en offrant la possibilité à l'utilisateur de définir à la fois une valeur minimale et maximale pour chaque paramètre. Cette approche permettra aux utilisateurs de mieux cibler leurs critères de recherche.

J'ai structuré le code en créant des composants distincts pour chaque élément de filtrage, qu'il s'agisse de cases à cocher, de menus déroulants, ou de sliders.

Ces composants individuels communiquent avec un composant parent centralisé nommé "VehiculeFilters" pour gérer l'état des filtres de manière cohérente.

Ne pouvant directement communiquer entre eux, je vais créer pour les composants parents et fils une fonction handlechange qui prendra en paramètre un évènement lorsque l’utilisateur cliquera sur le bouton rechercher. Elle sera définie pour mettre à jour l’état des filtres lorsque l’utilisateur effectuera une sélection dans l’un des composant filtre. Je la passerait en tant que prop aux composant fils pour qu’ils puissent mettre à jour l’état parent.

Chaque composant a été créé en utilisant la bibliothèque Material-UI. J'ai installé les dépendances nécessaires pour garantir leur bon fonctionnement.

**Composant fils BasicRange :**

Le destructuring a été employé pour extraire les valeurs des propriétés, avec des valeurs par défaut définies pour une utilisation plus flexible.

Le composant **BasicRange** comporte les éléments suivants :

- handleChange : Cette fonction est appelée à chaque modification de la valeur du slider et reçoit deux paramètres : le nom du paramètre (comme prix, kilométrage, année) et la nouvelle valeur sélectionnée.

- label : Un libellé qui s'affiche au-dessus du slider pour aider l'utilisateur à comprendre les valeurs sélectionnées.

- name : Le nom du slider.

- marks : Un tableau d'objets spécifiant les marques (valeurs) sur le slider, chaque objet ayant une valeur et une étiquette (label). Par exemple, pour le prix : "5 000 - 50 000".

- range :La plage de valeurs possibles du slider. J'ai utilisé une plage de base de 0 à 100, qui convient à de nombreuses situations. J'ai veillé à ce que la plus petite valeur soit toujours en première position en utilisant une fonction de tri pour maintenir l'ordre correct.

**Composant fils BasicSelect :**

Ce composant permet de créer un menu déroulant avec des options configurables, telles que les marques de voitures. Lorsque l'utilisateur sélectionne une option, la fonction handleChange est réutilisée pour communiquer avec le composant parent.

**Composant Parent VehiculeFilters :**

Ce composant sera donc le parent et se composera des composants fils BasicCheckbox, BasicSelect et BasicRange.

Pour le slider "année", j'ai créé une fonction permettant de mettre à jour la date de fin du slider en temps réel, évitant ainsi une mise à jour manuelle chaque année.

J'ai implémenté la fonction handleChange, qui est transmise aux composants enfants via les props et qui met à jour l'état local des filtres en fonction des modifications apportées par l'utilisateur.

La fonction handleCheckbox sera utilisée pour gérer les cases à cocher. Elle mettra à jour l’état des filtres en fonction de ce qui est coché ou pas.

Je vais utiliser le hook useEffect pour effectuer les requêtes HTTP lorsqu’il sera monté (affiché pour la première fois) grâce à AXIOS qui effectuera ces requêtes et ainsi récupérer les données des véhicules en fonction des paramètres de filtres.

**Difficultés rencontrées :**

Cette partie du développement a été l'une des plus complexes pour moi. Tout d'abord, j'ai initialement essayé de créer mes propres composants de sliders, de cases à cocher, etc., alors que des bibliothèques existaient pour simplifier ce processus. J'ai aussi éprouvé des difficultés à bien cibler mes recherches, mais j'ai heureusement bénéficié de conseils avisés qui m'ont facilité la tâche. Mon apprentissage s'améliore au fil du temps, et je suis désormais conscient des ressources disponibles pour résoudre de tels problèmes techniques.

De plus, j'ai rencontré des défis lors de la mise en place de la fonction handleChange et de sa transmission en tant que props aux composants enfants, mais j'ai compris l'importance de cette fonction pour établir la communication entre les composants.

J'ai également été confronté à plusieurs erreurs liées à des noms de composants mal orthographiés, tant au niveau des noms de fichiers que des imports. Je suis conscient de la sensibilité à la casse dans ce contexte, mais dans ce cas précis, j'ai eu du mal à identifier l'origine de l'erreur, car tous les noms semblaient correctement écrits. Pour résoudre ce problème, j'ai dû supprimer tous les fichiers des composants et les recréer un à un afin que l'erreur cesse de s'afficher. C'est l'un des mystères de l'informatique qui peut parfois entraîner une perte de temps considérable pour ce qui semble être une petite erreur en apparence.

.

**\*\*Conclusion :\*\***

En fin de compte, la mise en place des composants de recherche et de filtrage a constitué une étape cruciale de mon stage. Malgré les défis rencontrés, j'ai pu les surmonter en utilisant des approches manuelles. Cette expérience m'a permis de développer mes compétences en gestion de bases de données et d'acquérir une compréhension plus approfondie des aspects pratiques du développement. Mon apprentissage continu dans ce domaine m'aidera à aborder de futurs projets avec plus de confiance et de compétence.