Fransioli Matthias Licence CASIR

**Angular – Car Rental**

**Dans quelle mesure l'application répond aux requis fonctionnels**

* Nous avons une page d'accueil du site
* Nous avons une page contact qui envoie un message (affiché en console seulement)
* Nous affichons la liste de tous les véhicules disponibles
* Nous affichons les détails d'un véhicule sélectionné
* Un utilisateur connecté peut réserver un véhicule
* Peuvent se connecter et se déconnecter :
  + Un utilisateur
  + Un admin (qui aura le droit d'accéder à la partie Admin)
* Un utilisateur connecté peut voir ses infos sur sa page compte et l'historique des réservations effectuées
* ~~Nous effectuons au moins 2 filtres sur cette liste de véhicules~~
* ~~Ces pages sont protégées suite à un login et réservé à l'utilisateur qui est Admin~~
* ~~Nous affichons une liste d'éléments (Customer ou Vehicle ou Booking~~)
* ~~Il est possible d’Ajouter, Modifier et Supprimer un élément~~

**Les parties sur lesquelles j’ai travaillé**

Lors des séances à l’IUT, en cours ou en autonomie, nous avons privilégié le pair programming. Cependant lorsque nous avons travaillé chacun de notre côté, Hugo a travaillé sur la mise en place des différents composants, le style des pages et le contenu, tandis que je me suis plutôt concentré sur les parties fonctionnelles de l’application.

Ainsi j’ai pu travailler sur les aspects suivants :

* Le routage de l’application,
* La récupération et l’affichage du catalogue
* La connexion de l’utilisateur ou de l’administrateur, et le contrôle de ses droits (est ce que l’utilisateur qui s’est connecté est admin ou non ?) ; la déconnexion,
* L’adaptation du menu principal en fonction du statut de l’utilisateur (non connecté, connecté, admin),
* L’affichage en console du message de contact envoyé,
* La réservation de véhicule par l’utilisateur connecté,
* La récupération et l’affichage des informations personnelles de l’utilisateur connecté

**Les choix techniques effectués (Composants, Classe, Service, ...?)**

Sur chaque vue de l’application (écran), le corps correspond à un component. Le menu principal est un component à part que nous intégrons dans chacun des autres components. Cela nous permet de ne pas avoir à réécrire le component à chaque fois.

Les interactions avec l’API (back-office) sont regroupées dans des services selon leur thèmes. Un service pour ce qui concerne les véhicules, un service pour les réservations, etc.. . Afin de manipuler plus facilement les données, la récupération de ces dernières ne se fait pas directement dans les services. Les services fournissent seulement des fonctions sur lesquelles on peut appliquer la méthode `.subscribe()` pour obtenir ou envoyer les données voulues à l’API.

Pour gérer la connexion de l’utilisateur et avoir une persistance de cette connexion entre les différentes pages de l’application, nous avons utilisé le `sessionStorage` afin de stocker l’id et le rôle de l’utilisateur connecté. Cependant, même si cette méthode permet de gérer la connexion et la déconnexion de manière efficace, elle possède des limites en termes de sécurité. Ainsi, une V2 de l’application pourrait voir le processus de connexion être implémenté à l’aide de Classes.

**Une rétrospective du projet (ce qui s'est bien passé, moins bien passé, ce qui vous a aidé, ...)**

**Ce qui s’est bien passé** :

La première partie de story mapping et de rédaction des issues nous a permis de poser les bases du projet. L’objectif et le contenu du projet ont rapidement été clairs pour nous.

La gestion du code avec git est également un point positif. Nous arrivons à travailler de façon fluide avec cet outil, même si la gestion des merges reste à parfaire afin d’avoir un historique de commits plus lisible.

**Ce qui s’est moins bien passé :**

Nous avons connu des difficultés d’organisation pendant l’exécution de la partie technique du projet. Cela s’explique en partie parce que nous devions mener de nombreux projets en même temps dans différentes matières, mais également parce que nous n’avons pas assez exploité les possibilités de la méthode agile pendant la réalisation. Nous aurions gagné à mettre en place un sprint backlog et le mettre à jour régulièrement.

Cette erreur d’organisation nous a conduit à avoir du mal à évaluer précisément l’importance respective des tickets définis pendant l’étape de préparation du projet.

**Difficultés :**

Au départ, utiliser Angular a été difficile car c’est un Framework que nous découvrions. Mais ce problème a évolué au fil du projet puisque nous avons désormais compris le fonctionnement d’Angular et avons appris à l’utiliser.

**Ce qui nous a aidés :**

Comme dit précédemment, la partie de planification nous a permis de poser le projet et d’avoir les idées claires avant de démarrer dans le code.  
Les différentes ressources et la documentation à propos d’Angular nous ont aussi été d’une aide précieuse afin de comprendre et d’approfondir ce Framework.