



ENSSAT

L A N N I O N

Projet - Gestion de location de véhicules

Rapport JEE

Octobre - Novembre 2020

Membres de l'équipe :

PORHIEL Clément LE GALLO Benjamin

MAZÉ Gurvan SOULEYREAU Dorian

TOUTANT Steven

Responsables du module :

M. CHENNEBAULT

Introduction

Le projet **UML-JEE** consiste à mener une analyse UML à partir du cahier des charges et à concevoir en conséquence l'application JEE de **Gestion de location de véhicules**. L'analyse UML doit permettre d'identifier l'ensemble des acteurs puis les cas d'utilisation, avec les diagrammes qui en découlent. L'ensemble des fonctionnalités sera implémenté dans l'application JEE selon les conditions et règles de l'analyse UML.

Ce rapport a pour objectif de détailler l'implémentation des fonctionnalités de l'application en présentant nos choix d'architectures et les outils que nous avons utilisés. Ensuite, nous expliquons la méthodologie que nous avons choisie ainsi que les pratiques mises en place.

Table des matières

Développement	4
Objectifs	4
Framework	4
Gestionnaire de paquets	5
Dépendances	5
Dépendances tests	7
Architecture	8
Interface utilisateur	9
Authentification	9
Vue d'ensemble	11
Clients	11
Réservations	12
Véhicules	14
Collaborateurs	15
Statistiques	15
Gestion de projet	17
Méthodologie Kanban	17
Discord	17
Repository GitHub	18
Intégration continue	19
CircleCI	19
Codecov	19
Hook	19
Conclusion	20

Développement

Objectifs

L'application de Gestion de location de véhicules doit être implémentée en JEE (*Java Enterprise Edition*) qui est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées.

Le principe de l'**application distribuée** est de mettre en place une architecture logicielle permettant l'exécution d'un programme sur plusieurs ordinateurs.

Une architecture N-Tiers (ou **distribuée**) typique est celle constituée d'un client, d'un serveur d'**application** et d'un serveur de base de données.

Framework



Nous avons décidé d'utiliser le Framework Java : **Spring Web MVC** qui simplifie le développement d'une application web.

Seules deux personnes de l'équipe maîtrisaient ce framework, les autres membres de l'équipe ne l'avaient jamais utilisé auparavant. Ce projet a été l'occasion d'apprendre ou d'approfondir ce framework incontournable dans le monde de Java EE.

Gestionnaire de paquets

Pour la gestion des paquets (dépendances), nous utilisons le gestionnaire de paquets Maven.

Maven

C'est un outil de construction de projets open source développé par la fondation Apache. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java.

Il permet notamment :

- d'automatiser certaines tâches : compilation, tests unitaires et déploiement des applications qui composent le projet
- de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet

Dépendances

L'ensemble des dépendances que nous avons utilisées au sein du projet sont citées ci-dessous :

Spring Boot

Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar, totalement autonome. De plus, il embarque nativement un serveur Tomcat.

Spring Boot Web

Spring Boot Web est utilisé pour la création d'applications Web, y compris RESTful, utilisant Spring MVC. Cette dépendance utilise le serveur par défaut de Spring Boot, Tomcat.

Spring Security

Spring Boot Security est utilisé pour la sécurité d'applications. Il fournit un cadre d'authentification et de contrôle d'accès.

Spring Thymeleaf

Thymeleaf est un moteur de template Java moderne côté serveur pour les environnements Web et autonomes.

Spring Data JPA

Spring Data JPA traite de la prise en charge améliorée des couches d'accès aux données basées sur JPA (Java Persistence API). Cela facilite la création d'applications utilisant Spring, qui utilisent des technologies d'accès aux données.

Hibernate

Hibernate ORM est un outil de mappage objet-relationnel pour le langage de programmation Java. Il fournit une structure permettant de mapper un modèle de domaine orienté objet vers une base de données relationnelle.

MySQL

Le pilote JDBC de MySQL permet de communiquer avec notre base de données.

Lombok

Le projet Lombok est une bibliothèque Java qui se connecte à l'éditeur et ajoute des fonctionnalités. Il permet d'éviter d'écrire les "getters", "setters" ou autres méthodes équivalentes, et même les constructeurs. Ceci grâce à des annotations écrites dans nos classes.

Mailjet

Mailjet est une plate-forme de distribution de courrier électronique fiable et évolutive avec SMTP et API. Nous l'utilisons pour envoyer des mails transactionnels aux utilisateurs (oubli de mot de passe, première connexion).

Dépendances tests

On utilise plusieurs dépendances pour les tests, parfois complémentaires à celles utilisées pour l'application.

Spring DevTools

Spring DevTools est utilisé pour le rechargement automatique du serveur lors de modifications de code.

Spring Test

Spring Test est utilisé pour les tests de l'application. Il permet de tester notre application Spring Boot.

Spring Security Test

Spring Security Test permet de tester la sécurité et les accès de l'application.

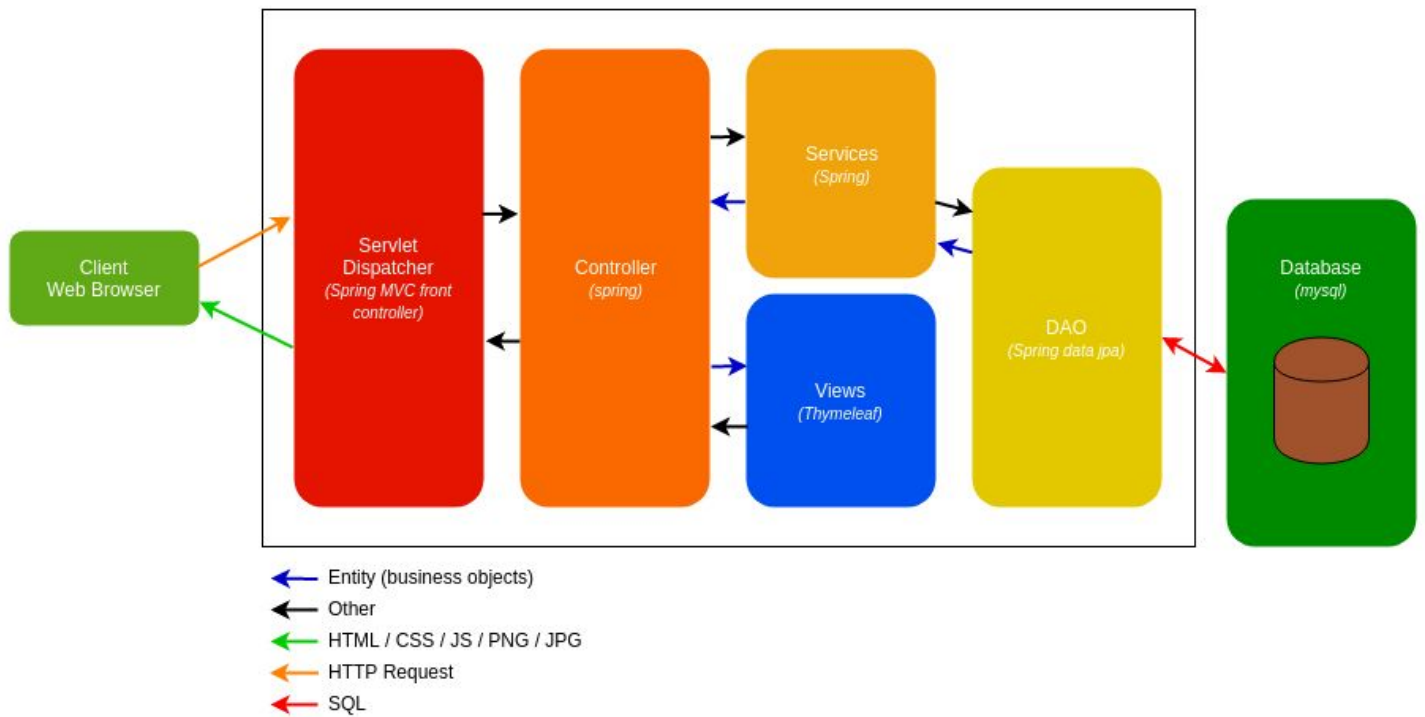
H2database

H2 est un système de gestion de base de données relationnelle écrit en Java. Il peut être intégré à une application Java ou bien fonctionner en mode client-serveur. Nous l'utilisons comme base de données embarquée pour nos tests en tant que base de données en mémoire. Elle se construit dès que nous lançons les tests et insère les données fournies dans un fichier *.sql. Ce qui nous permet de toujours tester le code avec les mêmes données.

Mockito

Mockito permet de tester unitairement les méthodes de nos classes java en s'assurant que chaque méthode est totalement indépendante du reste du programme au moment de l'exécution du test unitaire.

Architecture



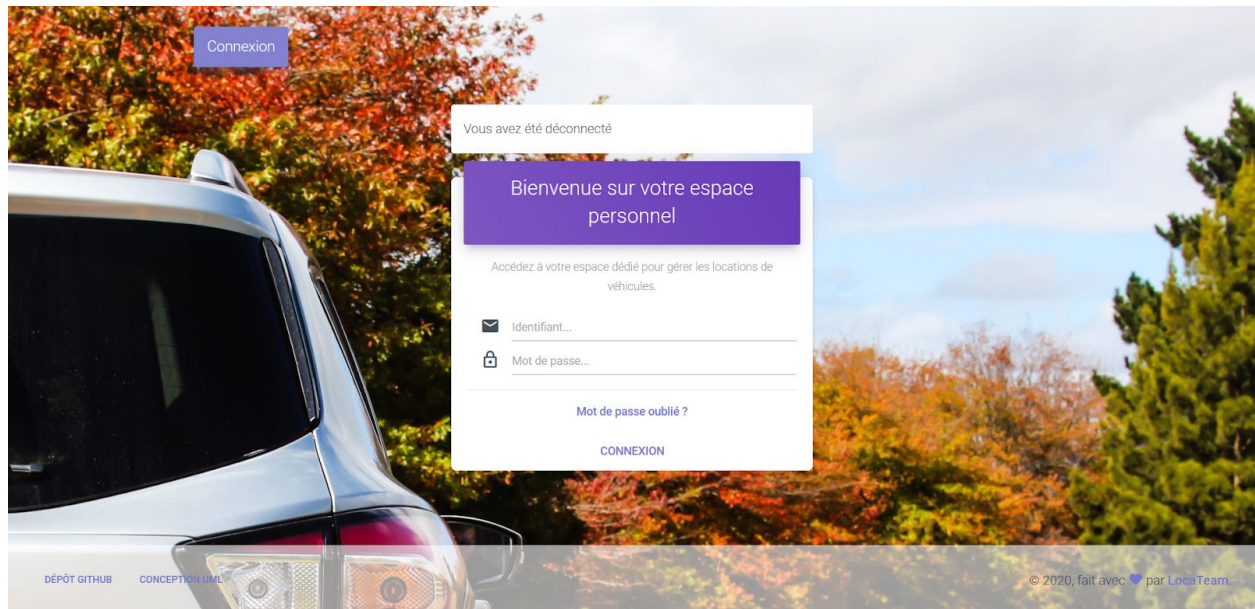
Architecture de notre application

Notre architecture est découpée en 3 parties :

- La première est la partie **DAO** pour Data Access Object, ce modèle permet de récupérer les données persistantes.
- Ensuite, nous avons la partie **Services**, elle permet de faire l'interface entre le **DAO** et la dernière partie les **Controller**.
- La dernière partie concerne les **Controller**, son objectif est de traiter la requête du client et de lui fournir les données à travers des vues.

Interface utilisateur

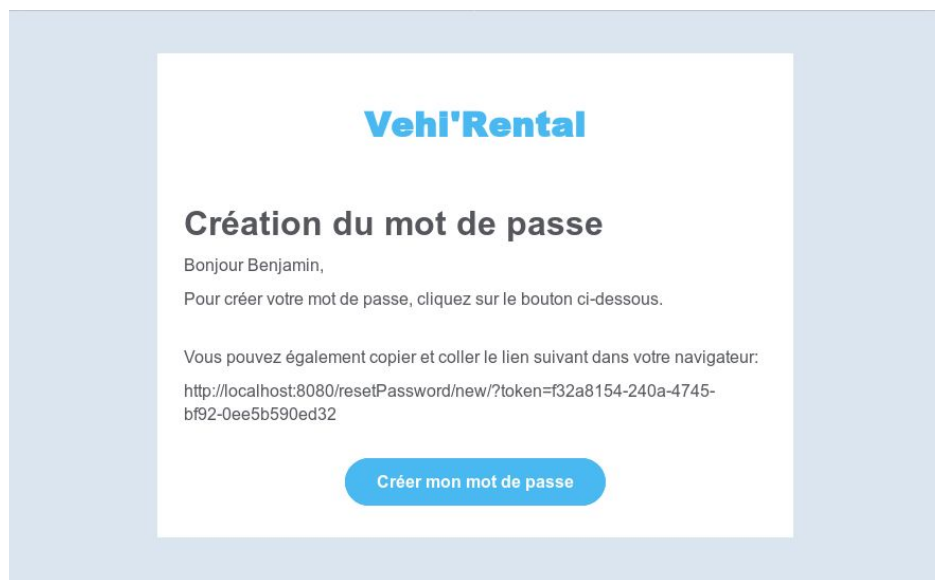
Authentification



Page de connexion

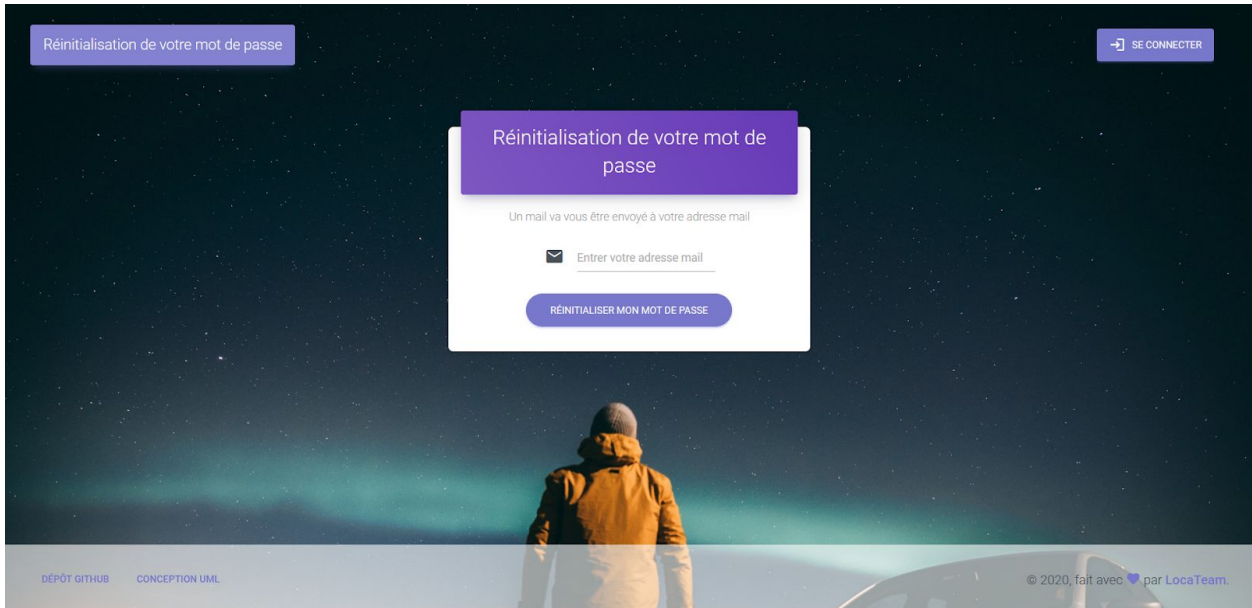
Voici la page de connexion sur laquelle l'utilisateur se connecte à l'application. Il peut également demander à réinitialiser son mot de passe en cas d'oubli.

Lors de la création de son compte, le nouveau collaborateur reçoit par mail une demande de création de mot de passe grâce à un token qui expire dès lors qu'il a été utilisé ou que sa durée de vie est dépassée.



Email de bienvenue lors de la création d'un collaborateur

S'il oublie son mot de passe, un mail sera envoyé et l'utilisateur aura la possibilité de réinitialiser son mot de passe à partir d'un lien.



Réinitialisation de votre mot de passe

SE CONNECTER

Réinitialisation de votre mot de passe

Un mail va vous être envoyé à votre adresse mail

✉ Entrer votre adresse mail

RÉINITIALISER MON MOT DE PASSE

DÉPÔT GITHUB CONCEPTION UML

© 2020, fait avec par LocaTeam.

Réinitialisation du mot de passe



Vehi'Rental

Réinitialisation du mot de passe

Bonjour Benjamin,

Pour réinitialiser votre mot de passe, cliquez sur le bouton ci-dessous.

Vous pouvez également copier et coller le lien suivant dans votre navigateur:

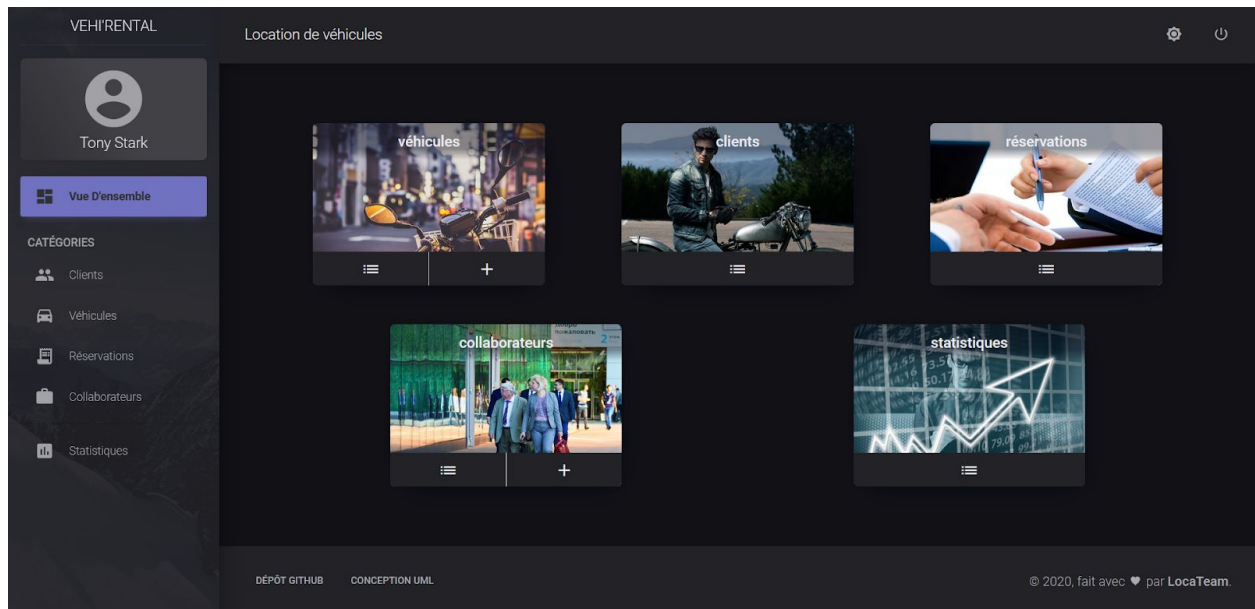
<http://localhost:8080/resetPassword/pwd?token=86229d7a-863e-43b2-b21e-993ee447cc73>

Réinitialiser mon mot de passe

Email de réinitialisation du mot de passe

Vue d'ensemble

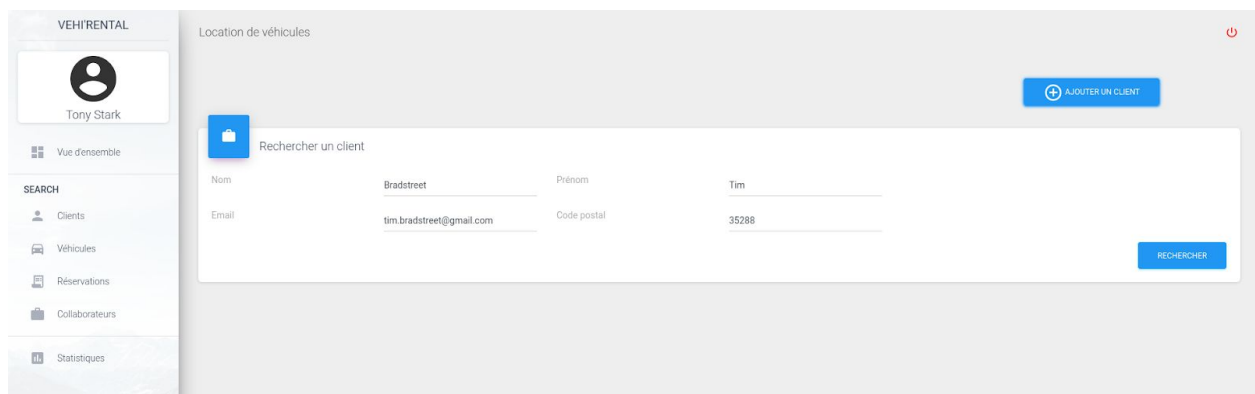
Voici la page d'accueil d'un responsable de location, après s'être authentifié à l'application.



Page d'accueil du responsable de location

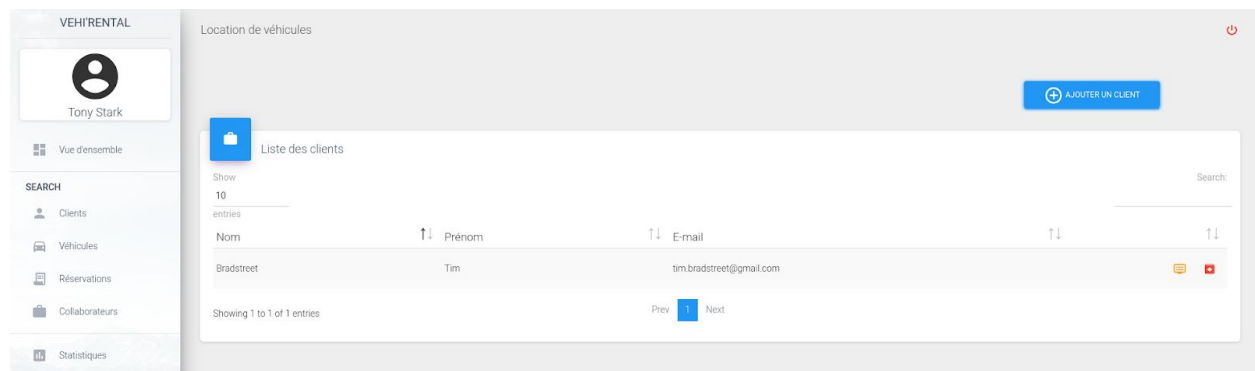
Clients

Seuls le responsable de location et les gestionnaires commerciaux peuvent accéder aux fonctionnalités concernant les clients.



Formulaire de recherche d'un client

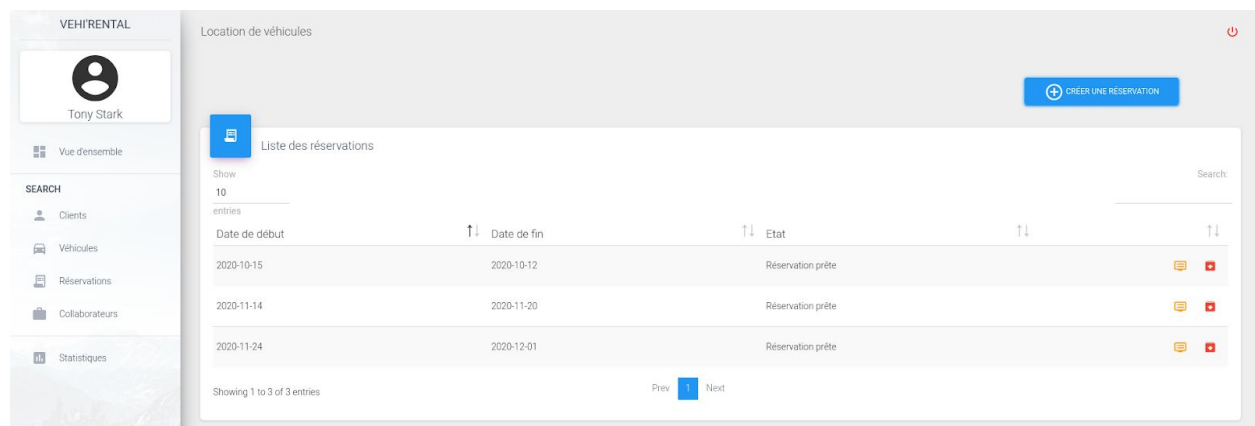
Une fois la recherche effectuée, l'utilisateur peut interagir avec les clients. En effet, il peut créer, modifier ou archiver un client.



Liste des clients avec les différentes actions possibles

Réservations

Pour consulter les réservations, il faut être le responsable de location ou un gestionnaire client.



Liste des réservations avec les différentes actions possibles

Lors de l'ajout d'une réservation, l'utilisateur doit sélectionner un client et le véhicule souhaité. Afin de ne pas conduire à des incohérences la date de début de la réservation doit être située après la date au moment où l'on crée la réservation. Ainsi, il n'est pas possible de créer une réservation qui commence avant le jour où nous la créons.

VEHIRENTAL

Tony Stark

Vue d'ensemble

SEARCH

- Clients
- Véhicules
- Réservations
- Collaborateurs
- Statistiques

Location de véhicules

Créer une réservation

Date de début de location

Date de fin de location

Selectionnez le véhicule

F-LARX cessna 172sp

Selectionnez le client

Bradstreet Tim tim.bradstreet@gmail

Nombre de kilomètres (voiture ou moto) ou heures estimées (avions) selon le véhicule

0

ANNULER

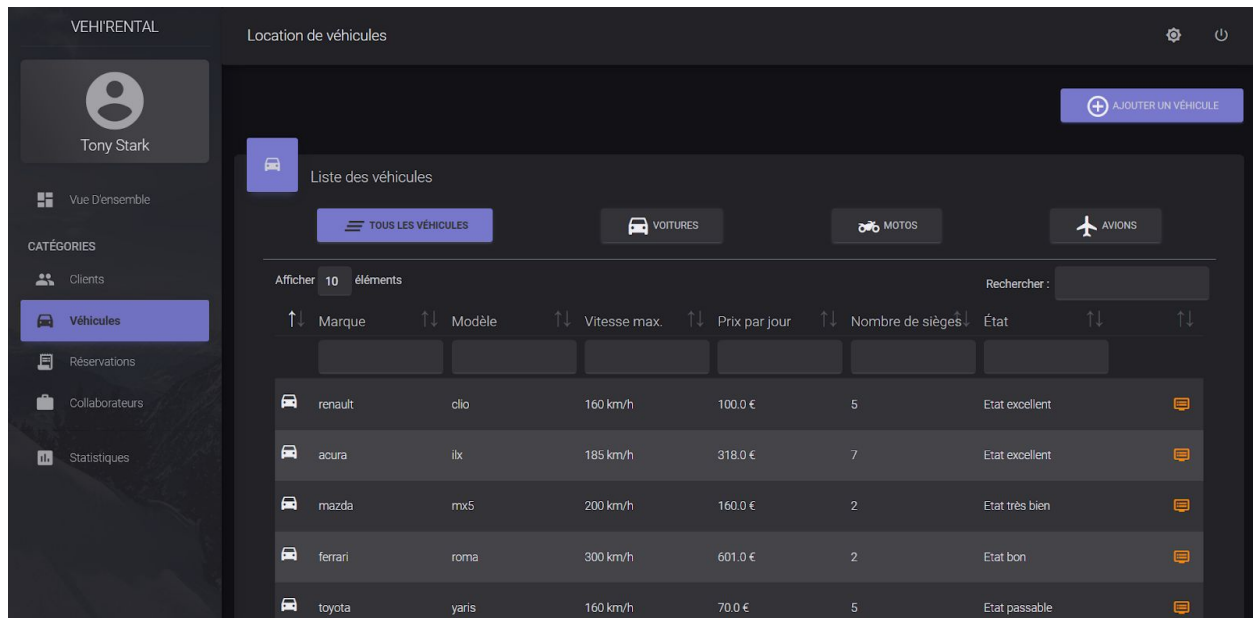
ENREGISTRER

Ajout d'une réservation

Afin de tester le fonctionnement du service et du repository chargé de gérer la réservation, nous avons mis en place des données fictives dans une base de données de test. Cependant, nous avons eu des difficultés à rendre ces tests génériques et fonctionnels tout le temps. Les dates associées à une réservation sont dépendantes de la date du jour, donc nous avons fait en sorte d'utiliser la date du jour pour construire nos données de test.

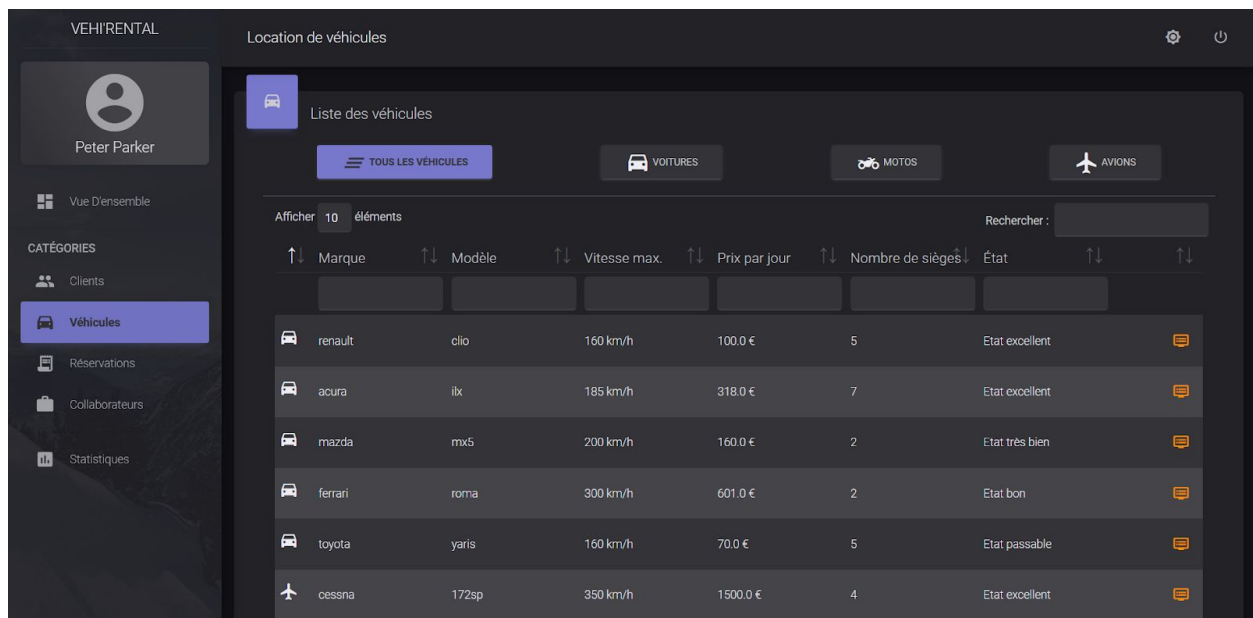
Véhicules

L'ensemble des collaborateurs peut visualiser les véhicules mais certaines actions sont réservées à un type de collaborateur en particulier.



Liste des véhicules avec les différentes actions possibles

Par exemple, un gestionnaire commercial ou client n'aura pas la possibilité d'ajouter un véhicule.



Liste des véhicules avec la possibilité d'ajouter un nouveau véhicule

Collaborateurs

Le responsable de location a la possibilité de gérer les collaborateurs. Il peut visualiser un collaborateur et modifier sa fiche.

The screenshot shows a web application interface for 'VEHIRENTAL' with the title 'Location de véhicules'. On the left is a sidebar with a user profile for 'Tony Stark' and a list of categories: 'Vue D'ensemble', 'Clients', 'Véhicules', 'Réservations', 'Collaborateurs' (highlighted), and 'Statistiques'. The main area displays the 'Fiche du collaborateur' form with the following fields:

Nom	Parker	Prénom	Peter		
Numéro d'immatriculation	893	Téléphone	+33612482274		
Date de naissance	03/05/1969	Email	spiderman@marvel.com		
Adresse	quelque part				
Code Postal	000	Ville	New York	Pays	USA
Fonction	Collaborateur				

At the bottom right of the form is a 'MODIFIER' button. The footer contains 'DÉPÔT GITHUB', 'CONCEPTION UML', and '© 2020, fait avec ❤️ par LocaTeam'.

Modification de la fiche d'un collaborateur

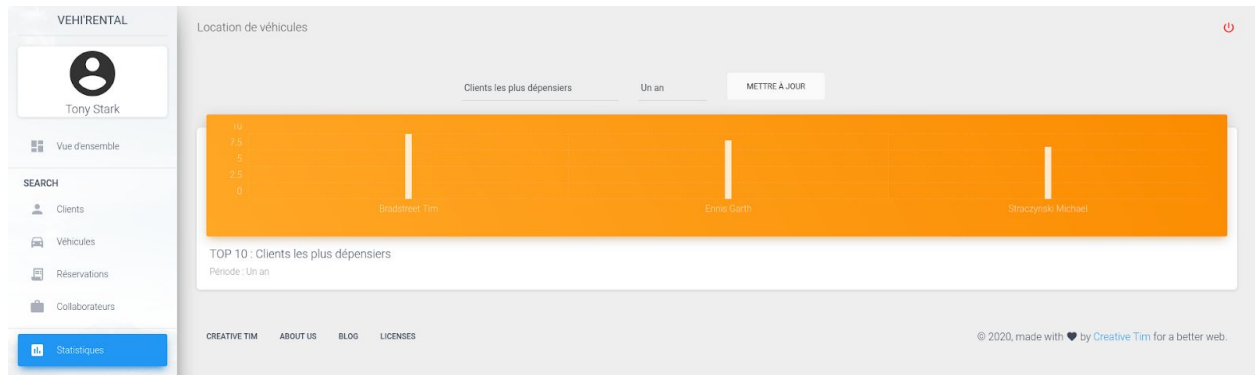
Statistiques

Seuls le responsable de location et un gestionnaire commercial ont la possibilité de visualiser les statistiques. Les statistiques concernent le TOP 10 des clients les plus dépensiers et ayant le plus de réservations. L'utilisateur peut choisir le type de TOP 10 ainsi que sa période.

La mise en place du TOP 10 dans le code est un peu particulière. Les top 10 contiennent des données qui évoluent au cours du temps, donc il n'est pas possible de créer des tables dans la base de données. Notre analyse nous a conduit à réfléchir sur deux types de mise en place. Une première idée a été de mettre en place des vues dans la base de données qui contiennent chaque top 10 et de les utiliser directement comme des entités dans le programme Java. Une deuxième idée a été de créer des requêtes sql qui sont situées directement dans le code java, et qui permettent d'obtenir les statistiques.

Finalement notre choix s'est porté sur la mise en place de requêtes sql directement dans le programme en java. Nous avons sélectionné cette option, car l'utilisation des vues dans spring boot ont entraîné des soucis au niveau des tests. En effet, dans l'application spring boot les vues sont assimilées à des entités (ou table), donc lors des tests, le framework essaie de créer les vues comme si elles étaient des tables. Cependant cela ne fonctionne pas, car une vue n'est pas créer comme une table.

Donc plus aucun test ne pouvait fonctionner. De plus, la quantité de documentation sur la manipulation des vues dans spring était bien moins importante que sur la mise en place des requêtes sql dans le programme java.



TOP 10 des clients les plus dépensiers durant la dernière année

Gestion de projet

Méthodologie Kanban

L'avantage de cette méthode permet d'éviter la surproduction. Grâce à sa flexibilité, il est plus facile de gérer la mise en place de nouvelles fonctionnalités.

Le système d'étiquette permet de facilement visualiser l'avancé du projet.

Pour mettre en place cette méthodologie, nous avons utilisé Trello. Les étiquettes sont classées parmi cinq catégories :

- **Backlog** : permet de stocker les fonctionnalités à créer.
- **Todo** : ce sont les fonctionnalités à prioriser.
- **Doing** : la fonctionnalité est en cours de développement.
- **To validate** : le développement de la fonctionnalité est terminé et prêt à être validé.
- **Done** : la fonctionnalité est opérationnelle.

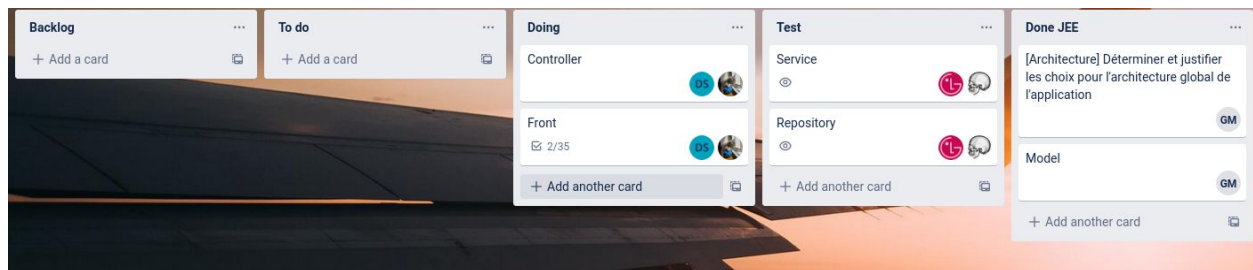


Tableau de bord Trello du projet

Discord

Nous avons utilisé Discord comme messagerie instantanée. Cela est très utile pour poser nos questions en direct ou bien encore mieux en vocal. De plus, on peut partager notre écran ce qui permet de déboguer à distance une autre personne de l'équipe. On gagne beaucoup de temps avec cet outil car on peut partager nos avancées et avoir un feedback très rapidement.

Repository GitHub

Nous avons utilisé Git pour le versionning de notre code et également pour travailler en équipe sur la même application en simultané.

Nous avons mis en place plusieurs règles sur le repository GitHub.

- La branche main (*anciennement master*) n'est pas directement modifiable, il faut passer par des merge requests. Pour valider le merge request, il faut que les tests avec *CircleCI* soient passés avec succès et qu'une personne de l'équipe valide le merge après une revue de code.
- Les autres branches quant à elles, sont libres

L'avantage de "bloquer" la branche main, est d'avoir une version valide et vérifiée de l'application.

Voici l'adresse de notre repository GitHub :



<https://github.com/jiben22/vehicles-rental>

Intégration continue

CircleCI

L'intégration continue est un ensemble d'outils liés au développement pour tester les nouvelles versions en continu. Techniquement, cela permet d'automatiser des tâches : compilation, tests unitaires et fonctionnels, validation produit... À chaque modification du code source.

Nous utilisons l'outil CircleCI pour lancer les tests unitaires et la création du WAR (Web Archive Java) de l'application.



<https://circleci.com/gh/jiben22/vehicles-rental/tree/master>

Codecov

Lors du lancement des tests unitaires avec CircleCI, nous faisons également une analyse du code avec l'outil CodeCov. Cela permet de voir la couverture et la complexité du code au fil du projet.

 <https://codecov.io/gh/jiben22/vehicles-rental/branch/master>

Hook

Le fichier *README* contient deux badges. Le premier badge permet de savoir si les tests avec CircleCI sont passés avec succès ou s'ils ont échoué. Le deuxième badge permet de connaître le pourcentage de couverture de code.



Badges affichés dans le README

L'URL de ces badges contient le nom de la branche de travail, pour les mettre à jour on utilise le hook *pre-commit*, qui récupère le nom de la branche en cours et met à jour les URL des badges dans le README à chaque commit.

Conclusion

Le projet **UML-JEE** nous a apporté un contexte concret afin de mettre en pratique nos connaissances en UML et JEE.

Chaque personne de l'équipe est désormais capable de créer une application Spring Boot avec l'ensemble des éléments de l'architecture citées précédemment.

De plus, travailler avec de nombreuses dépendances nous a permis d'augmenter la qualité de notre code et d'approfondir nos connaissances dans leurs applications.