

# Test & Sécurité

## Travaux Pratiques



Karim EL GHARBI

Florian REMY

DII5A

Promotion 2016 -2019

Test & Sécurité – Année 5

Décembre 2018

# TABLE DES MATIERES

Introduction.....	3
I. Cahier des charges.....	4
II. Problématiques.....	5
III. Réponse au besoin .....	5
1. <b>Architecture</b> .....	5
2. <b>Définition des vues IHM</b> .....	6
a. <b>Identification</b> .....	6
b. <b>Liste des véhicules</b> .....	7
c. <b>Panier</b> .....	8
IV. Aspect sécurité.....	10
V. Aspect tests .....	11
VI. Plan de développement.....	13

# Introduction

Le but de ce projet est de concevoir et de mettre en place une application proposant des annonces de véhicules de marque Rover.

Ce projet sera basé sur un ensemble de technologies, certaines maîtrisées, d'autres totalement nouvelles.

Le développement de l'application devra prendre en compte les notions de sécurité et de tests présentées en cours.

Le principe de l'application est inspiré des sites LeBonCoin et Ebay.



# I. Cahier des charges

L'application à développer devra comporter une interface listant les annonces de véhicules disponibles. La particularité de cette application de petites annonces est qu'elle offre la possibilité d'acheter immédiatement un ou plusieurs véhicules au moyen d'un panier. La nouveauté est donc ce croisement entre site de petites annonces et site de e-commerce.

L'utilisateur pourra donc sélectionner un ou plusieurs véhicules pour les ajouter à son panier. Une fois que l'utilisateur aura fini ses achats, il devra avoir la possibilité d'afficher son panier. Ainsi, l'utilisateur pourra régler la quantité et supprimer un article du panier (véhicule). La partie paiement ne sera pas gérée dans le cadre de ce projet.

Le développement sera réalisé en deux blocs. Une partie « back-end » s'occupera de tout l'aspect traitement et gestion des données. Ces données seront ensuite mises à disposition du second bloc « front-end » via une API WEB (API REST). Ainsi, via un certain nombre d'URL mises à disposition par le serveur WEB, le client aura accès à toutes les données qu'il souhaite afficher (liste des annonces, contenu du panier...). Le client pourra de même effectuer des requêtes via cette interface (API).



## II. Problématiques

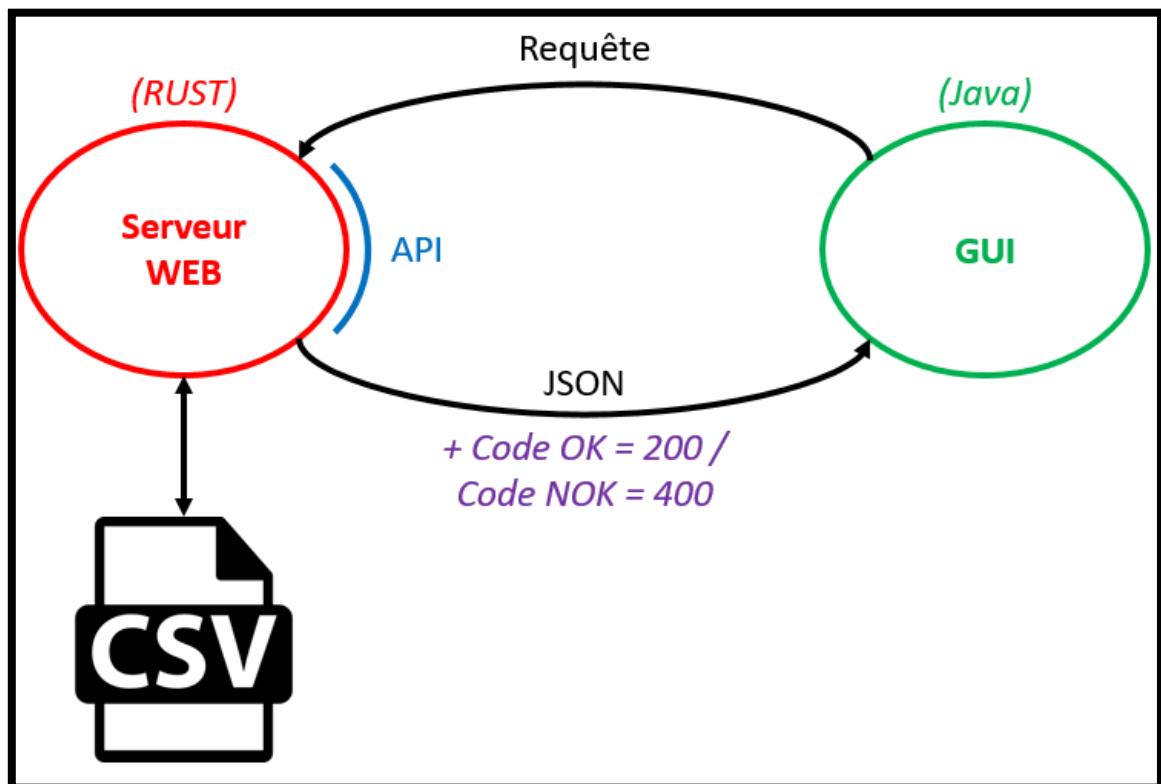
Plusieurs problématiques se présentent dans le cadre de ce projet. Premièrement, il s'agit de choisir des technologies plus ou moins connues et de développer l'application.

Ensuite, il faut se poser la question de l'interfaçage. Cela permet de savoir s'il est possible d'interfacer deux technologies différentes. Ainsi, une fois cette étape passée, il faut définir les cinématiques.

La problématique suivante concerne le stockage. La question est de savoir comment stocker les données qui seront mises à disposition du client.

## III. Réponse au besoin

### 1. Architecture



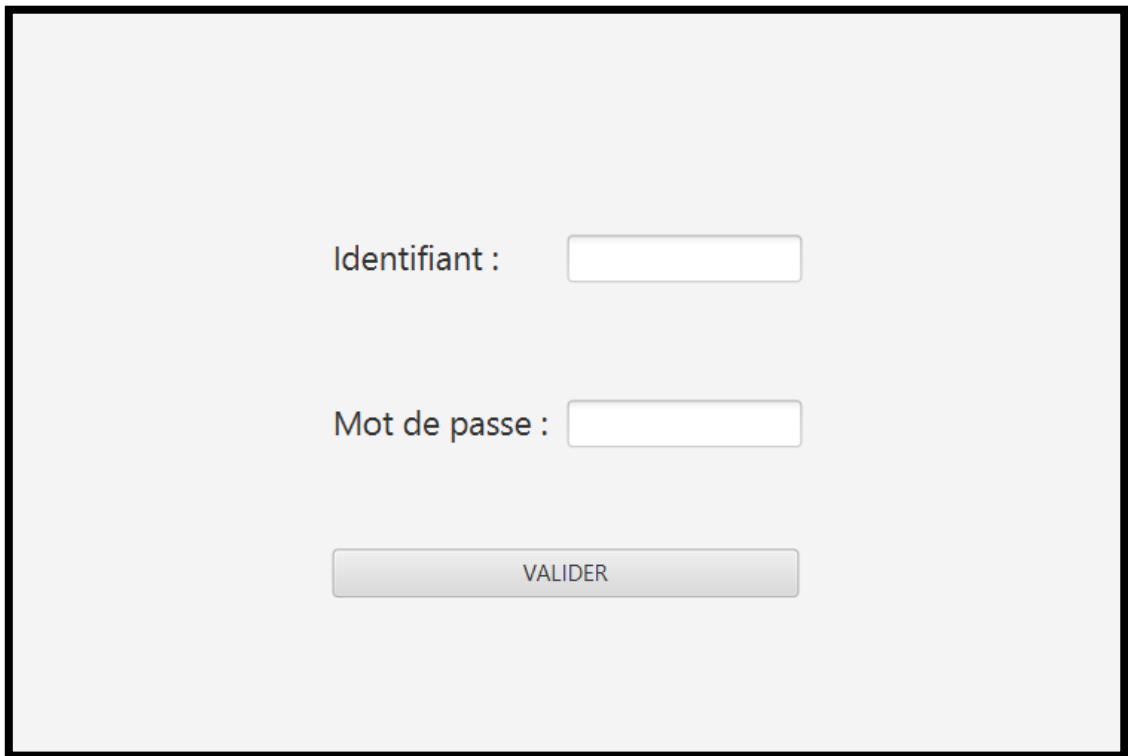
Voici ci-dessus l'architecture du projet à mettre en place. Le fonctionnement est le suivant : Le client effectue une requête au serveur via une API. Celui-ci renvoie le contenu s'il existe avec un message (ex : OK = 200 / NOK = 400). Le serveur met à disposition des données qu'il formate en json. Ces données proviennent d'un fichier csv qui remplace une base de données. Par conséquent, des données seront également écrites dans ce fichier csv.

## 2. Définition des vues IHM

### a. Identification

#### ➤ Vue écran

Une première vue IHM permettra de saisir un identifiant et un mot de passe. Cette opération ne servira qu'à récupérer l'identifiant. Ainsi, le panier sera associé à un utilisateur. Aucune gestion du mot de passe ne sera effectuée.



Identifiant :

Mot de passe :

#### ➤ Règles de gestion

Règle	Scénario	Conséquence
#login-r0	Saisie de l'identifiant seul	Champ mot de passe en rouge
#login-r1	Saisie du mot de passe seul	Champ identifiant en rouge
#login-r2	Validation sans saisie	Champs identifiant / mot de passe en rouge
#login-r3	Saisie de l'identifiant et du mot de passe + validation	Identifiant enregistré en mémoire

## b. Liste des véhicules

### ➤ Vue écran

Voici ci-dessous la vue qui permettra de visualiser l'ensemble des annonces disponibles et d'ajouter des articles au panier (**bouton +**). Une fois tous les achats effectués, le client devra cliquer sur le bouton en forme de caddie pour visualiser les articles sélectionnés.



### ➤ Règles de gestion

Règle	Scénario	Conséquence
#list-r0	Clic sur un véhicule	Surlignement de la ligne en bleu
#list-r1	Clic sur une colonne	Classement des articles par ordre alphabétique
#list-r2	Clic sur le bouton « + » (avec sélection)	Ajout de l'article sélectionné au panier
#list-r3	Clic sur le bouton « + » (sans sélection)	Aucune action
#list-r4	Clic sur le bouton « panier »	Affichage de la page « panier »

Remarque : Tous les formats de photos sont acceptés (*jpeg, jpg...*).

### c. Panier

#### ➤ Vue écran

Enfin, voici la vue qui regroupe tous les véhicules que le client souhaite acheter. Il est ainsi possible de régler la quantité voir de supprimer un article (**bouton rouge**). Le prix total du panier sera mis à jour au fur et à mesure.

Titre	Description	Quantité	Prix
Aucun contenu dans la table			

Total :

#### ➤ Règles de gestion

Règle	Scénario	Conséquence
#cart-r0	Clic sur un véhicule	Surlignement de la ligne en bleu
#cart-r1	Clic sur une colonne	Classement des articles par ordre alphabétique
#cart-r2	Clic sur le bouton « delete » (avec sélection)	Suppression de l'article sélectionné du panier



Règle	Scénario	Conséquence
#cart-r3	Clic sur le bouton « delete » (sans sélection)	Aucune action
#cart-r4	Affichage de la page	Envoi d'une requête au serveur pour récupérer le prix total

Remarques :

- *Tous les prix sont affichés en € sous forme de chaîne de caractères.*
- *La gestion de la disponibilité des articles a été jugée inutile*
- *L'utilisateur pourra saisir une quantité illimitée (entière) dans le champ quantité*

## IV. Aspect sécurité

L'aspect sécurité sera satisfait par la partie « back-end ». Pour répondre aux exigences de l'exercice, il s'agira d'effectuer de la vérification et du contrôle des requêtes. L'objectif est de contrôler les requêtes effectuées par le client. De plus, les échanges entre le serveur et le client pourront éventuellement être chiffrés.

Si le temps le permet, il peut être intéressant d'utiliser des outils pour identifier les failles et les vulnérabilités. Au minimum, il faudra tester le code manuellement en effectuant de la relecture de code en binôme.

## V. Aspect tests

Pour mettre en œuvre des notions de tests dans ce projet, il a été décidé que cet aspect sera traité du côté « front-end ». En effet, il a été choisi de développer l'application indépendamment de la partie serveur de manière à valider l'interface et les fonctionnalités. Ainsi, l'interfaçage sera simplifié car il ne restera qu'à implémenter les requêtes http à destination du serveur WEB (via l'API REST). Une phase de tests sera donc planifiée à la suite du développement voir pendant le développement (Test sprint 1, Test sprint 2 ...). Cela permet d'anticiper sur les tests d'intégration.

L'outil principal qui sera mis en œuvre pour tester la partie « front » sera JUnit. Celui-ci permet de faire du test unitaire et par conséquent de valider les fonctionnalités. Cette méthode permet d'éviter que toute sorte d'erreur ou de défaut ne génère de défaillance.



Les tests fonctionnels (boîte noire) ne seront pas ceux qui seront privilégiés durant ce projet. Néanmoins, en plus des tests structurels (boîte blanche), des tests IHM pourront être effectués si le temps le permet. Si les outils existent ou sont compatibles (ex : cypress...) pour Java, ceux-là pourront être automatisés. Dans le cas contraire, les tests seront faits manuellement mais décrits explicitement dans une procédure de tests.

Pendant le développement, il pourra être intéressant de mettre en œuvre Sonarlint pour traiter les problématiques de complexité et de propreté du code (qualimétrie). Cela accentue l'aspect qualité à toujours respecter.



Remarque : il existe un plugin pour Eclipse.

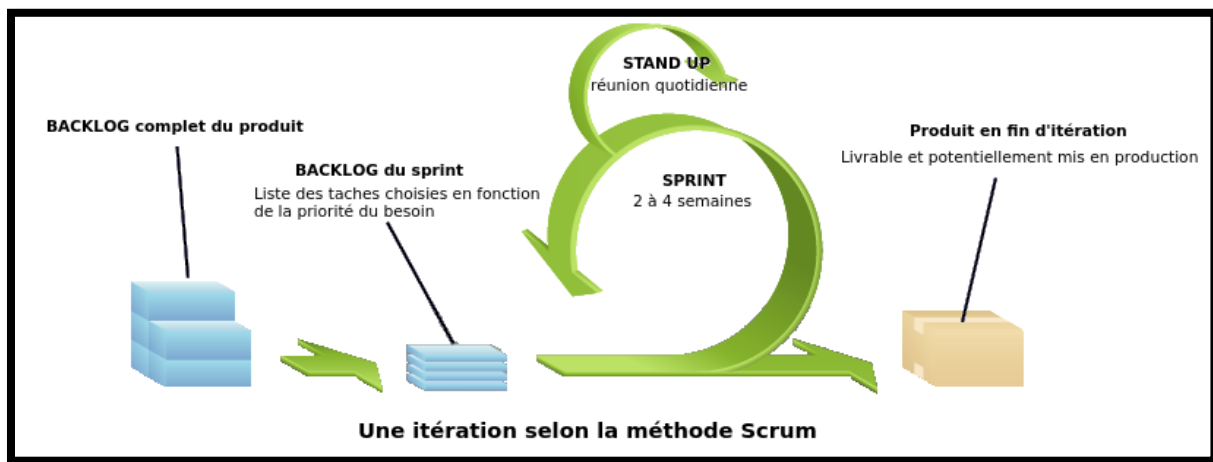
La mise en production pourra être étudiée sans être forcément mise en œuvre. Néanmoins, si le temps le permet, il peut être utile d'essayer de mettre en œuvre l'outil Jenkins.



En résumé, JUnit est l'outil qui doit impérativement être utilisé. Les autres seront optionnels selon l'échéance du projet.

## VI. Plan de développement

Ce projet va se décomposer en plusieurs phases. L'idée est de mettre en place une méthode agile. La méthode agile qui a été choisie est la méthode SCRUM. Ce choix se justifie par le fait que ce projet peut être amené à évoluer en fonction des priorités et des attentes du client. C'est justement une particularité de la méthode SCRUM qui privilégie l'implication constante du client de façon à ce que les exigences soient bien prises en compte. De plus, cette méthode permet de réagir rapidement à un changement, ce qui donne plus de flexibilité sur la réalisation du projet. Voici ci-dessous le schéma de la démarche à mettre en place.



Cette démarche est incrémentale / itérative, il sera donc possible de fournir des parties de livrables au client au fur et à mesure. Ainsi, le développement du projet suivra un bon chemin.

Pour mettre en œuvre cette méthode, plusieurs sprints ont été définis :

- 1<sup>ère</sup> sprint : Cahier de spécifications v1 (13/12 à 12h30)
- 2<sup>ème</sup> sprint : 1<sup>ère</sup> version de l'application (21/12 à 16h00)
- 3<sup>ème</sup> sprint : Cahier de spécifications v2 (21/12 à 18h15)
- 4<sup>ème</sup> sprint : 2<sup>ème</sup> version de l'application (Fin Janvier)