



Dossier de projet Location de voiture CaramazinLease

29 NOVEMBRE

Caramazinlease

Créé par : Jean-jerome Sical



Sommaire

1.	LISTE DES COMPETENCES	3
2.	RESUME DU PROJET EN ANGLAIS	4
3.	CAHIER DES CHARGES	5
4.	GESTION DE PROJET	6
1.	<i>Cycle en V</i>	6
2.	<i>Méthode Agile</i>	7
5.	SPECIFICATIONS FONCTIONNELLES DU PROJET	8
1.	<i>Diagramme de cas d'utilisation</i>	8
2.	<i>Diagramme de classe</i>	9
3.	<i>La conception d'une base de données</i>	12
4.	<i>Les schémas de la base de données</i>	13
5.	<i>Les scripts de la base de données</i>	14
6.	<i>Diagramme d'activités</i>	16
7.	<i>Diagramme des séquences</i>	17
8.	<i>Maquettes des interfaces</i>	21
6.	SPECIFICATIONS TECHNIQUES DU PROJET	21
1.	<i>L'architecture du projet</i>	21
2.	<i>L'architecture J2EE (Java Enterprise Editions)</i>	22
3.	<i>Maven</i>	22
4.	<i>L'architecture 3-Tiers</i>	22
5.	<i>Méthodologie d'implémentation MVC</i>	24
6.	<i>Choix Techniques</i>	24
7.	LES EXTRAITS DE CODE LES PLUS SIGNIFICATIFS	28
8.	PRESENTATION DU JEU D'ESSAI ELABORE	33
1.	<i>Module 1 : Page de connexion</i>	33
2.	<i>Module 2 : Dashboard</i>	34
9.	DESCRIPTION DE LA VEILLE, SUR LES VULNERABILITES DE SECURITE	34
1.	<i>La gestion des vulnérabilités</i>	34
2.	<i>Les cinq étapes du cycle de gestion des vulnérabilités</i>	34
3.	<i>Les mesures de sécurité pour les applications web Les moyens pour sécuriser son Système d'Information ?</i>	36
10.	DESCRIPTION D'UNE SITUATION DE TRAVAIL	36
1.	<i>Spring boot sécurité (Authentication et Autorisation)</i>	37

2. Spring boot sécurité avec OAuth 2.0.....	48
11. REFERENCE	46

1. Liste des compétences

- **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité**
 - Faire la conception, la Modélisation de données
 - Faire de l'Algorithmique
 - Développer la partie front-end d'une interface utilisateur web.
 - Développer la partie back-end d'une interface utilisateur web.

- **Concevoir et développer la persistance des données en intégrant les recommandation sécurité**
 - Concevoir une base de données.
 - Mettre en place une base de données.
 - Développer des composants dans le langage d'une base de données.
 - Interroger une Base de Données

- **Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité**
 - Faire de l'intégration web
 - Concevoir une application
 - Développer des composants métier
 - Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
 - Construire une application organisée en couches
 - Préparer et exécuter les plans de test d'une application
 - Préparer et exécuter le déploiement d'une application
 - Faire de l'intégration continue

2. Résumé du projet en anglais

CarAmazinLease agency, is a Paris-based car renting company. The company would like a web application for theirs employees, in order to help its activities.

The web interface will be tailored to meet the client needs regarding the site's ergonomics and the expected contains.

This web application will allow the agency's employees to access the clients profile and their personal data, the leasing contracts as well as available vehicles, while integrating the recommendations for personal data protection.

They will have the possibility to :

- Search for their client profile their clients's contracts, whether they have subscribed to a contract or are simply registered within the application.
- Search for car, contrats and invoices; add new clients, new contracts and new cars to lease and billes.
- Modify the registered clients data, cars and even contracts subscribed by clients in case of errors.
- At last, delete clients, contracts, car and invoices.

Moreover, in order to access the web interface, employees would need to register an ID and password. This, to guarantee the company's safety.

My role in the project is to gather the agency's need, organize and implement the web application, step by step, from design to delivery.

The Application designer developper training from IBS Global Services, allowed me to acquire a new skillset and now I am able to design and develop a quality web application, compliant to CarAmazinLease's requirements.

3. Cahier des charges

L'objectif de ce projet est de créer une application de location de véhicules.

Nous devons créer une section administrateur pour les employés de l'agence CarAmazinLease. Chaque Employé accède à un Dashboard où il peut voir les informations sur les contrats, les clients, les voitures, les factures et les options. En particulier, les fonctionnalités suivantes seront requises :

Panneau d'administration :

- Ajouter, rechercher, lister toutes les voitures disponibles : Matricule, Marque, couleur, carburant, Puissance, vitesse maximale, Kilomètre actuel, En service, Date de mise en service
- Ajouter, rechercher, lister tous les clients : Nom, prénom, adresse, date d'anniversaire, fidélité
- Ajouter, rechercher, lister tous les contrats : Jour de location, date de début, date de fin, Total à payer, avance, Reste à payer, Lieu de Restitution, id voiture, id client
- Ajouter, rechercher, lister toutes les factures : Date, montant, adresse
- Ajouter, rechercher, lister toutes les options : Nom, description

Vous devez implémenter la sécurité comme la connexion.

Pour mener à bien ce projet, j'ai suivi les différentes étapes de développement à savoir : le recueil d'informations et de données, l'analyse et l'organisation des besoins des employés de l'agence et la phase de développement en traduisant les données recueillis en langage informatique.

Pour ce projet, j'ai utilisé le langage graphique de modélisation UML (Unified Modling Language), pour représenter visuellement une approche orientée Object pour les cas d'utilisation. La plateforme Java Entreprise Edition (JEE) a été utilisée pour le développement du projet.

L'ensemble du système de développement se basera sur des cas d'utilisation.

On va définir ces cas d'utilisation en UML, ce qu'on appelle : Diagramme de cas d'utilisation constitué des employés de l'agence et les tâches qu'ils auront à effectuer (Ajouter, Rechercher, modifier, supprimer).

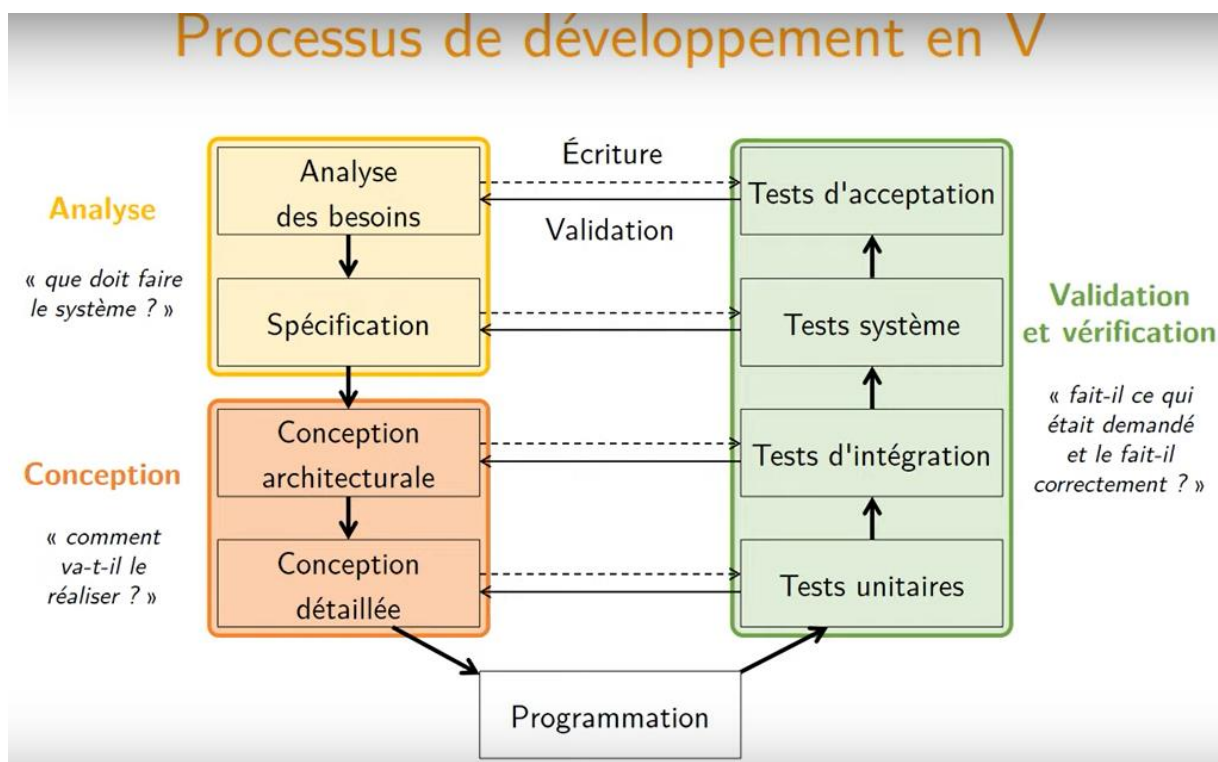
Et pour représenter l'application d'un point de vue concepteur, un diagramme de classe participative et un diagramme de séquence d'interaction ont été décrits pour le cas d'utilisation.

4. Gestion de projet

La gestion de projet consiste à élaborer toutes les étapes qui jalonnent le cycle de vie d'un projet, de la naissance de son idée à la livraison finale.

- ❖ Être planifié dans le temps
- ❖ Suivre des enjeux opérations et financiers importants

4.1 Cycle en V



Le cycle en V est un modèle d'organisation des activités d'un projet qui se caractérise par un flux d'activité descendant qui détaille le produit jusqu'à sa réalisation, et un flux ascendant, qui assemble le produit en vérifiant sa qualité.

Les avantages :

- Simple à élaborer
- Maîtriser les risques
- Une meilleure planification
- Amélioration de la qualité du produit grâce à l'intégration de mesures liées à l'assurance qualité
- Réduction des coûts

Dans l'ensemble, ce modèle peut permettre d'éviter **les malentendus ainsi que les tâches inutiles**. De plus, il permet de s'assurer que toutes les tâches soient exécutées en temps voulu, dans le bon ordre en réduisant les temps morts au maximum.

Les inconvénients :

- Trop simpliste
- Ne permet guère de réagir avec souplesse aux modifications en cours de développement et favorise donc un déroulement relativement linéaire du projet

Cependant pour palier au problème est nécessaire d'adopter les méthodes AGILES particulièrement la méthode SCRUM, qui est une méthodologie conçue pour fonctionner dans une structure itérative, s'adaptant aux changements, répondant à des retours constants dans le but de fournir des résultats constants.

4.2 Méthode AGILE

Dans la méthode AGILE, le projet est divisé en plusieurs sous-projets et je passe à l'étape suivante une fois que l'étape d'avant est concrétisée. Le client en fait partie tout au long et il valide chaque étape.

Les avantages :

- La flexibilité
- La possibilité de s'adapter en fonction des nouvelles exigences du client
- Meilleur contrôle des coûts
- Participation du client au projet

Les inconvénients :

- Il ne peut être véritablement mis en œuvre que lorsque les clients sont disponibles
- Le temps peut être gênant
- Difficultés d'adaptation pour la prochaine équipe

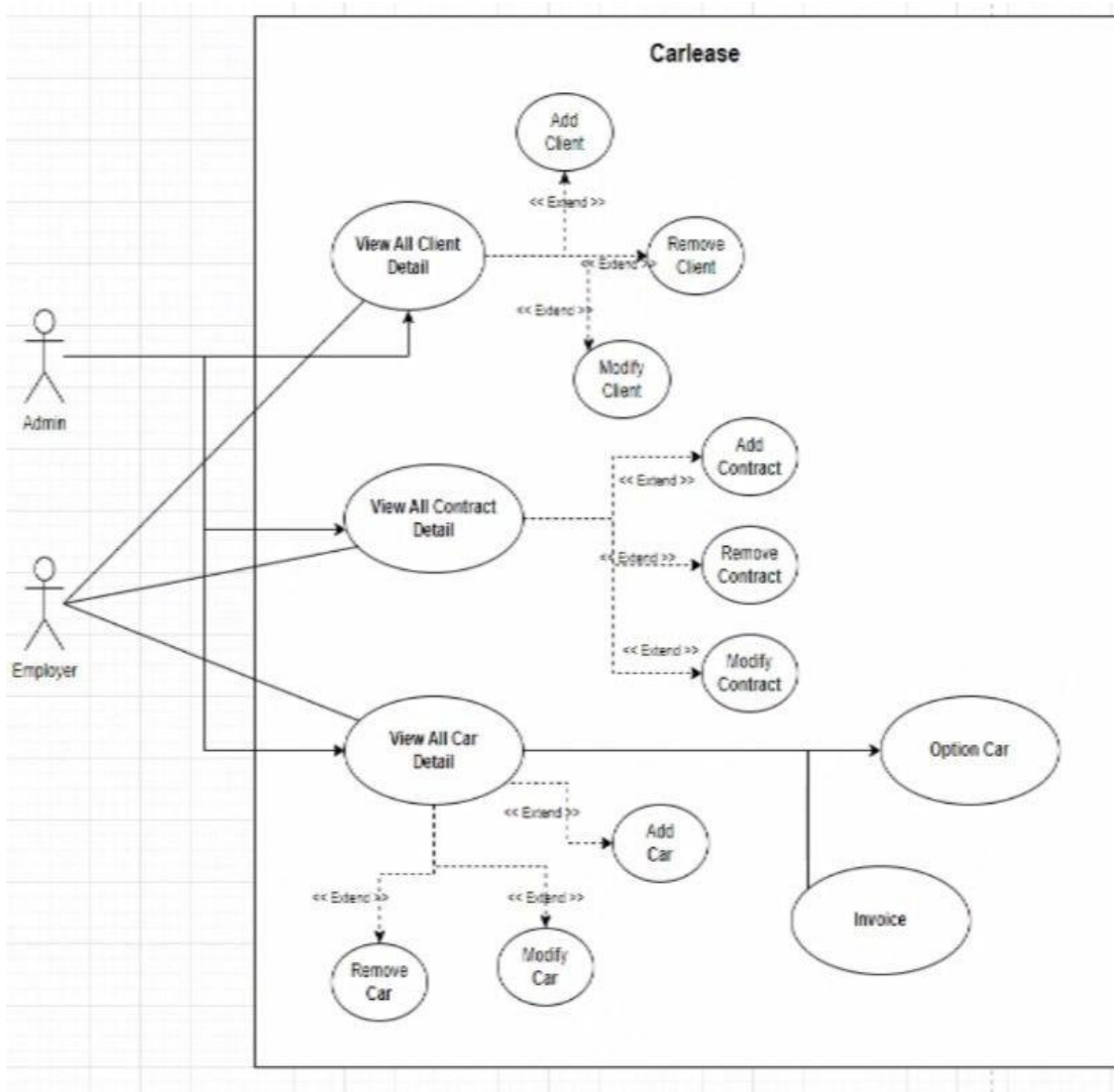
5. Spécifications fonctionnelles du projet

Les spécifications fonctionnelles ont pour objectif de décrire précisément l'ensemble des fonctions d'un logiciel ou d'une application afin de déterminer le périmètre fonctionnel du projet. Les spécifications sont basées sur l'expression de besoins puis retranscrites dans le cahier des charges du projet.

Les spécifications fonctionnelles se concentrent sur le fonctionnement de l'interface utilisateur. Il faut donc décrire avec précision les interactions avec l'utilisateur final pour chaque écran qu'il rencontre.

5.1 Diagramme de cas d'utilisation

Permet de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système ! Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les *use case* (cas d'utilisation) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur et un cas d'utilisation sont mis en relation par une association représentée par une ligne.



5.2 Diagramme de Classe

Le diagramme de classe montre la **structure statique du modèle d'information**, particulièrement les choses qui existent, leur structure interne, et leurs relations aux autres choses.

Composants de base d'un diagramme de classes

Le diagramme de classes standard est composé de trois sections :

- Section supérieure : contient le nom de la classe. Cette section est toujours nécessaire, que vous parliez du classifieur ou d'un objet.
- Section intermédiaire : contient les attributs de la classe. Utilisez-la pour décrire les qualités de la classe. Elle n'est nécessaire que lors de la description d'une instance spécifique d'une classe.

- Section inférieure : contient les opérations de la classe (méthodes), affichées sous forme de liste. Chaque opération occupe sa propre ligne. Les opérations décrivent la manière dont une classe interagit avec les données.

Modificateurs d'accès des membres

Toutes les classes ont des niveaux d'accès différents, en fonction du modificateur d'accès (indicateur de visibilité). Voici les niveaux d'accès existants et les symboles qui leur sont associés :

- **Public (+)**
- **Privé (-)**
- **Protégé (#)**
- **Paquetage (~)**
- **Dérivé (/)**
- **Statique (souligné)**

Portées des membres

Il existe deux portées pour les membres : les classifieurs et les instances.

Les classifieurs sont des membres statiques alors que les instances sont des instances spécifiques de la classe. Si les bases de la théorie orientée objet vous sont familières, alors il n'y a rien de révolutionnaire dans tout cela.

Structure de base d'un diagramme de classe

Composant	Description
Classes	<p>Modèle pour créer des objets et mettre en œuvre un comportement dans un système. En langage UML, une classe représente un objet ou un ensemble d'objets possédant une structure et un comportement communs. On les représente par un rectangle comprenant des lignes pour le nom de la classe, ses attributs et ses opérations. Lorsque vous dessinez une classe dans un diagramme de classes, seule la ligne supérieure est obligatoire, les autres sont facultatives et ne servent qu'à fournir des détails supplémentaires.</p> <ul style="list-style-type: none"> • Nom : première ligne d'une forme de classe. • Attributs : deuxième ligne d'une forme de classe. Chaque attribut de la classe apparaît sur une ligne distincte.

	<ul style="list-style-type: none"> Méthodes : troisième ligne d'une forme de classe. On les appelle aussi opérations ; elles apparaissent sous forme de liste, chaque opération occupant une ligne différente.
Signaux	Symboles qui représentent les communications à sens unique et asynchrones entre des objets actifs.
Types de données	Classifieurs qui définissent des valeurs de données. Les types de données peuvent modéliser les types primitifs et les énumérations.
Paquetages	Formes conçues pour organiser les classifieurs connexes d'un diagramme. On les symbolise par une grande forme rectangulaire à onglets.
Interfaces	Groupe de signatures d'opération et/ou de définitions d'attributs définissant un ensemble cohérent de comportements. Les interfaces sont semblables à des classes, mais une classe peut avoir une instance de son type et qu'une interface doit compter au moins une classe pour la mettre en œuvre.
Énumérations	Représentations de types de données définis par l'utilisateur. Une énumération comprend des groupes d'identificateurs qui représentent des valeurs de l'énumération.
Objets	Instances d'une ou plusieurs classes. On peut ajouter des objets à un diagramme de classes pour représenter des instances concrètes ou prototypiques.
Artefacts	Éléments du modèle qui représentent les entités concrètes d'un système logiciel, tels que des documents, des bases de données, des fichiers exécutables, des composants logiciels, etc.

Les interactions

Le terme « interactions » désigne les relations et liens divers qui peuvent exister dans les diagrammes de classes et d'objets. Voici quelques-unes des interactions les plus courantes :

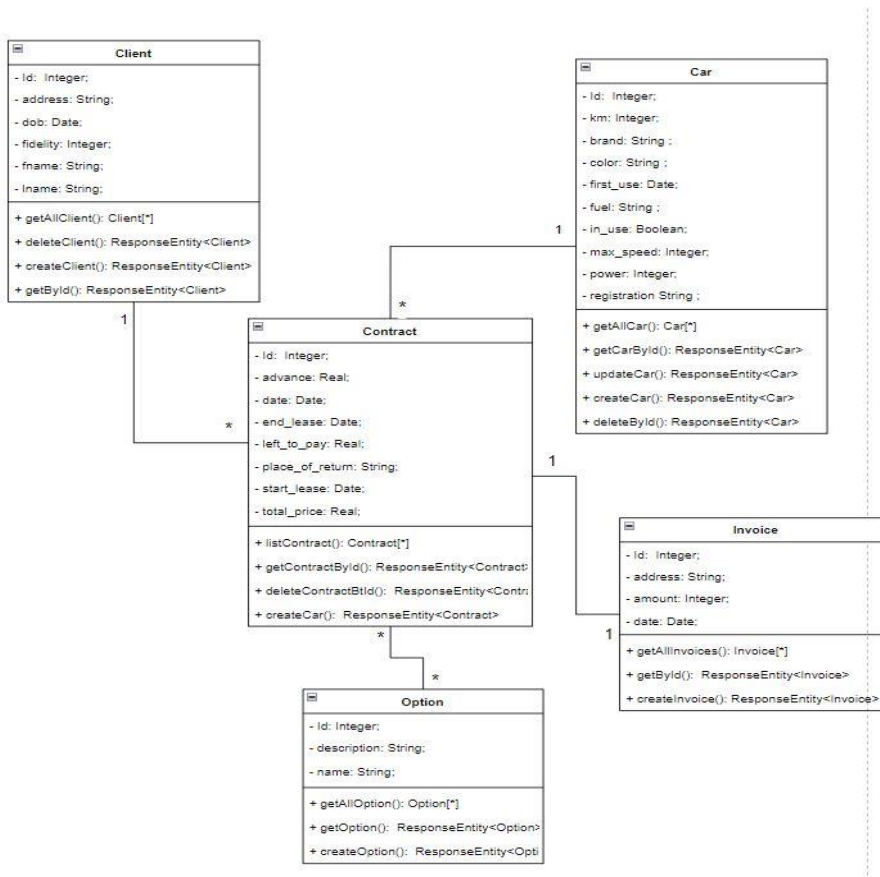
Héritage	Également connu sous le nom de généralisation, il s'agit du processus par lequel un enfant ou une sous-classe adopte la fonctionnalité d'un parent ou d'une super-classe. On le symbolise par une ligne de connexion droite avec une pointe de flèche fermée orientée vers la super-classe.
Association bidirectionnelle	Relation par défaut entre deux classes. Chacune des deux classes a conscience de l'existence de l'autre et de sa

	relation avec elle. Cette association est représentée par une ligne droite entre deux classes.
Association unidirectionnelle	relation un peu moins courante entre deux classes. Une classe a conscience de l'existence de l'autre et interagit avec elle. Une association unidirectionnelle est représentée par une ligne de connexion droite avec une pointe de flèche ouverte allant de la classe « sachante » vers la classe « connue ».

Composés principalement du projet **CaramazinLease** : 4 classes

- Car
- Client
- Contract
- Invoice
- Option

➤ La classe **Contract** est associée à **Car**, à **Client** et à **Invoice**



5.3 La conception d'une base de données

Le diagramme de classes n'a pas juste pour fonction de montrer le visuel, la description et de documenter les différents aspects d'un système, mais il va aussi Décrire la vue

statique du système, montrer la collaboration entre les éléments de la vue statique, de décrire les fonctionnalités réalisées par le système, de construire des applications logicielles utilisant des langages orientés objet, comme le cas de ce projet de location de véhicule.

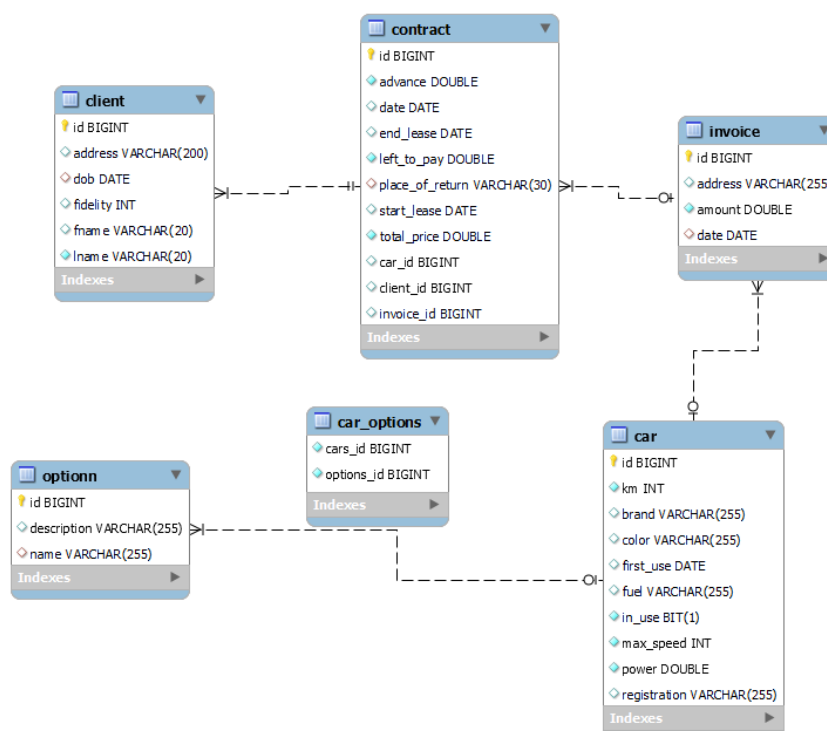
En outre, le Diagramme va aussi servir à concevoir une base de données relationnelle, qui permettra de stocker et de retrouver les données structurés, semi-structurés ou des données brutes ou l'information, en rapport avec les activités à réaliser lors de la conception et du développement.

On relève un point très important de l'utilisation des bases de données, la création de tables.

Une table est un ensemble de données organisées sous forme d'un tableau où les colonnes correspondent à des catégories d'information (les attributs).

5.3.1 Les Schémas de la base de données

Un schéma de base de données représente la configuration logique de tout ou partie d'une base de données relationnelle. Le schéma est une vue structurelle d'une base de données. Il peut se présenter à la fois sous la forme d'une représentation visuelle et d'un ensemble de formules, appelées « contraintes d'intégrité », qui régissent une base de données.



5.3.2 Les scripts de la base de données

L'application Location de véhicule, 5 tables (car, client, contract, invoice, option) ont été créées avec les mêmes noms de classes, et les champs de ces tables sont les mêmes que l'attribut de la classe.

```
-----  
-- Table `caramazinlease`.`car`  
-----
```

```
⊖ CREATE TABLE IF NOT EXISTS `caramazinlease`.`car` (  
  `id` BIGINT NOT NULL AUTO_INCREMENT,  
  `km` INT NOT NULL,  
  `brand` VARCHAR(255) NULL DEFAULT NULL,  
  `color` VARCHAR(255) NULL DEFAULT NULL,  
  `first_use` DATE NULL DEFAULT NULL,  
  `fuel` VARCHAR(255) NULL DEFAULT NULL,  
  `in_use` BIT(1) NOT NULL,  
  `max_speed` INT NOT NULL,  
  `power` DOUBLE NOT NULL,  
  `registration` VARCHAR(255) NULL DEFAULT NULL,  
  PRIMARY KEY (`id`))  
ENGINE = MyISAM  
AUTO_INCREMENT = 24  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `caramazinlease`.`client`  
-----
```

```
⊖ CREATE TABLE IF NOT EXISTS `caramazinlease`.`client` (  
  `id` BIGINT NOT NULL AUTO_INCREMENT,  
  `address` VARCHAR(200) NULL DEFAULT NULL,  
  `dob` DATE NULL DEFAULT NULL,  
  `fidelity` INT NULL DEFAULT '0',  
  `fname` VARCHAR(20) NULL DEFAULT NULL,  
  `lname` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`id`))  
ENGINE = MyISAM  
AUTO_INCREMENT = 6  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```

-----
-- Table `caramazinlease`.`contract`
-----

CREATE TABLE IF NOT EXISTS `caramazinlease`.`contract` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `advance` DOUBLE NOT NULL,
  `date` DATE NULL DEFAULT NULL,
  `end_lease` DATE NULL DEFAULT NULL,
  `left_to_pay` DOUBLE NOT NULL,
  `place_of_return` VARCHAR(30) NULL DEFAULT NULL,
  `start_lease` DATE NULL DEFAULT NULL,
  `total_price` DOUBLE NOT NULL,
  `car_id` BIGINT NULL DEFAULT NULL,
  `client_id` BIGINT NULL DEFAULT NULL,
  `invoice_id` BIGINT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FKixovqbw2hcpmeuelx4a7la9td` (`car_id` ASC) VISIBLE,
  INDEX `FKlhq3p3xl25vvnfyfc51ica0j` (`client_id` ASC) VISIBLE,
  INDEX `FK98iwe4169r0dg75e3r5d82764` (`invoice_id` ASC) VISIBLE)
ENGINE = MyISAM
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-----

-- Table `caramazinlease`.`invoice`
-----

CREATE TABLE IF NOT EXISTS `caramazinlease`.`invoice` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `address` VARCHAR(255) NULL DEFAULT NULL,
  `amount` DOUBLE NOT NULL,
  `date` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = MyISAM
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-----

-- Table `caramazinlease`.`optionn`
-----

CREATE TABLE IF NOT EXISTS `caramazinlease`.`optionn` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `description` VARCHAR(255) NULL DEFAULT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = MyISAM
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Car_id est la clé primaire dans la table Car

Client_id est la clé primaire dans la table Client

Contract_id est la clé primaire dans la table Contract

Invoice_id est la clé primaire dans la table Invoice

Option_id est la clé primaire dans la table Option

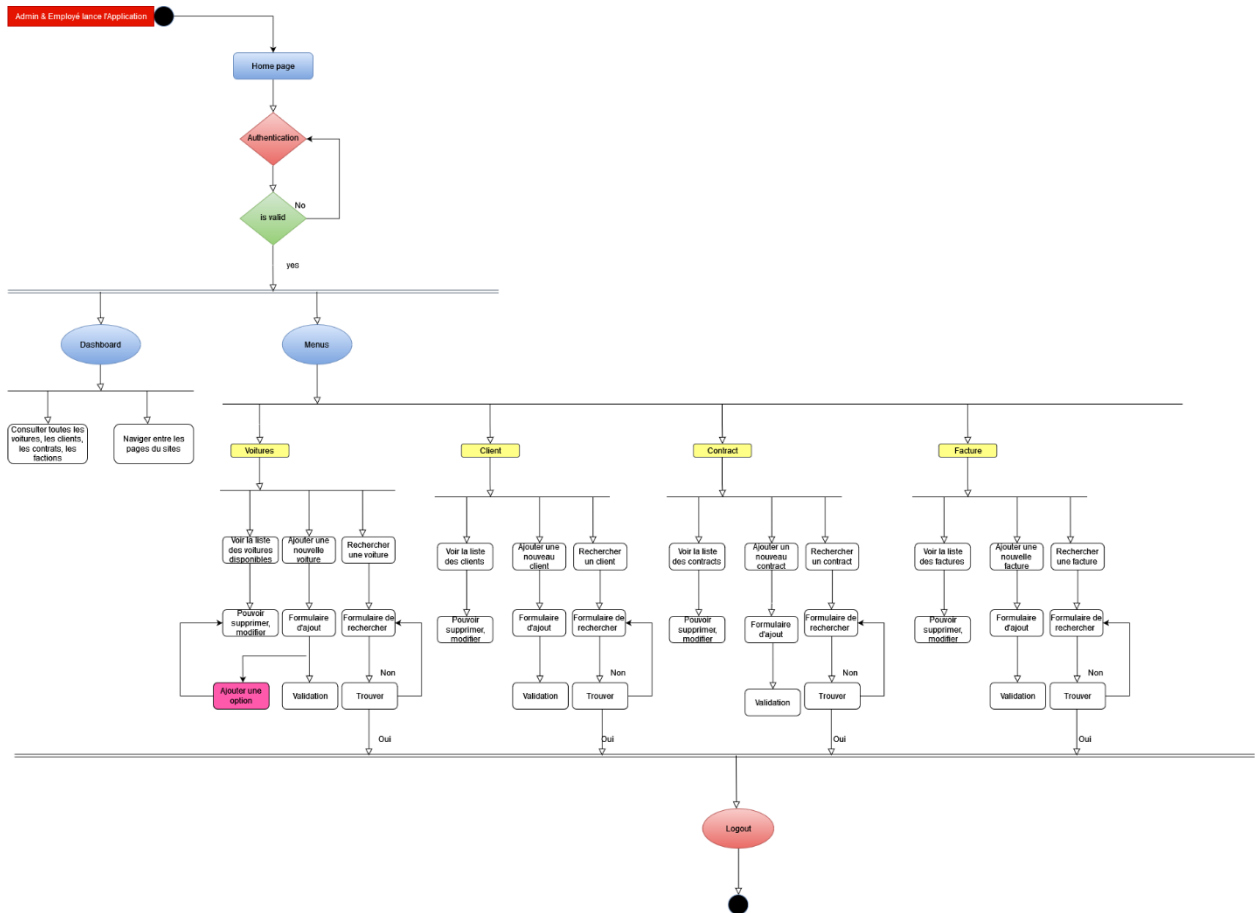
5.4 Diagramme des Activités

Les **diagrammes** d'activités permettent de mieux visualiser les schémas de procédés, d'identifier les points à améliorer et de mettre en évidence les domaines performants.

Les composants d'un diagramme d'Activité

Action	Étape dans l'activité où les utilisateurs ou le logiciel exécutent une tâche donnée.
Nœud de décision	Embranchement conditionnel dans le flux, qui est représenté par un losange. Il comporte une seule entrée et au moins deux sorties.
Flux de contrôle	Autre nom donné aux connecteurs qui illustrent le flux entre les étapes du diagramme.
Nœud de départ	Élément symbolisant le début de l'activité, que l'on représente par un cercle noir.
Nœud de fin	Élément symbolisant l'étape finale de l'activité, que l'on représente par un cercle noir avec un contour.

Diagramme des Activités – Employeurs & Admin



5.5 Diagramme des Séquences

Les diagrammes de séquence modélisent les interactions entre les objets dans un cas d'utilisation unique. Ils illustrent la manière dont les différentes parties d'un système interagissent entre elles pour exécuter une fonction, et l'ordre dans lequel les interactions se produisent lorsqu'un cas d'utilisation particulier est exécuté.

Quelques scénarios pour l'utilisation d'u diagrammes de séquence

Scénario d'utilisation	Un scénario d'utilisation est un diagramme décrivant comment votre système pourrait potentiellement être utilisé. C'est un bon moyen de s'assurer que vous avez pris en compte la logique de tous les scénarios d'utilisation du système.
Logique de méthode	De la même façon que vous pouvez utiliser un diagramme de séquence UML pour analyser la logique d'un cas

	d'utilisation, vous pouvez aussi vous en servir pour analyser la logique d'une fonction, d'une procédure ou d'un processus complexe.
Logique de service	Si vous considérez un service comme étant une méthode générale utilisée par différents clients, un diagramme de séquence est le moyen idéal de le schématiser.
Diagramme de séquence Visio	Une vue en vidéo du diagramme de séquence

Diagramme des Séquences – Login

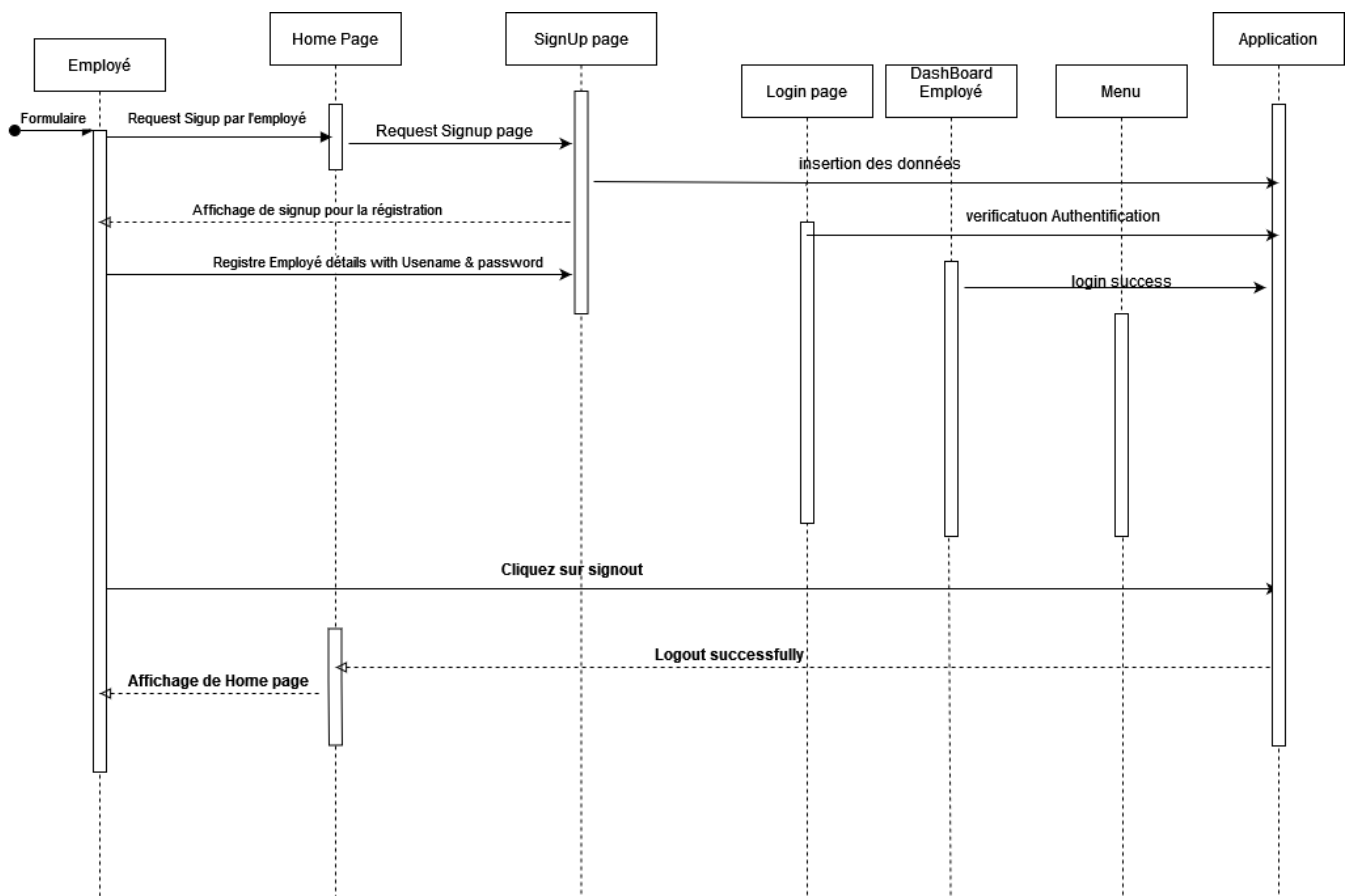


Diagramme des Séquences – Ajouter Un élément

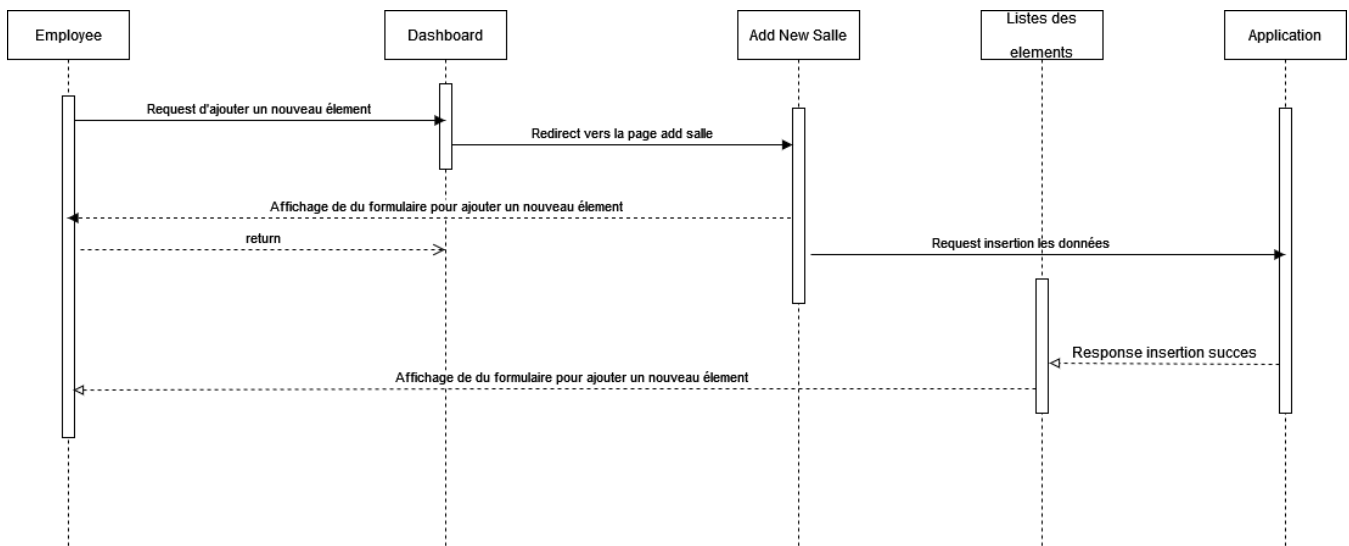


Diagramme des Séquences – Rechercher un élément

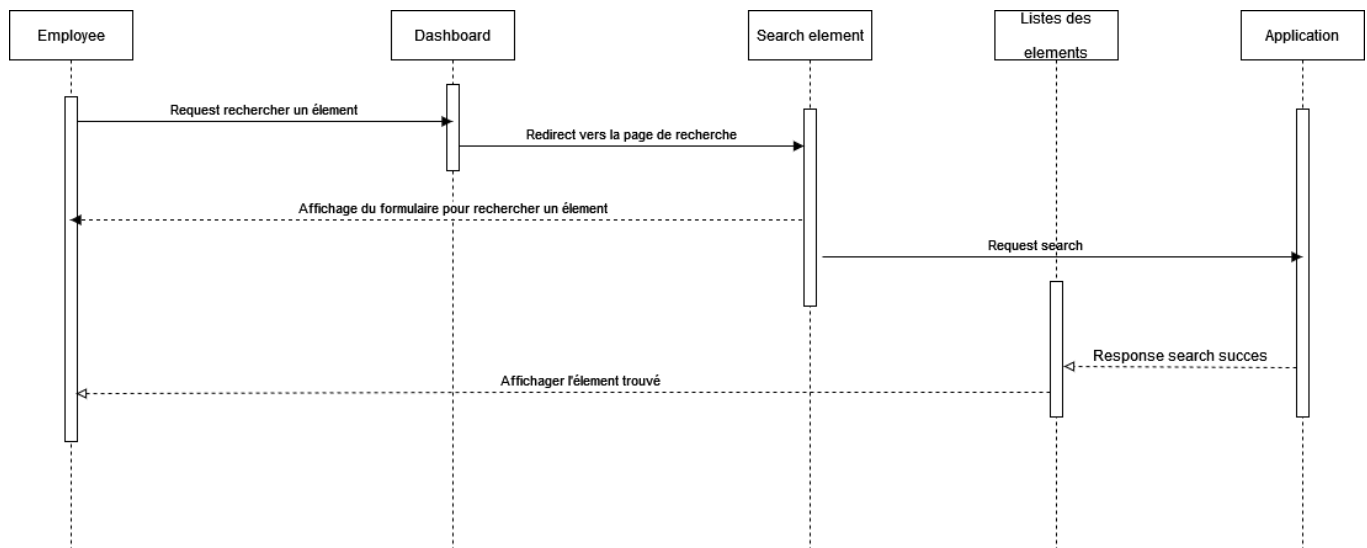


Diagramme des Séquences – Afficher tous les éléments d'une table

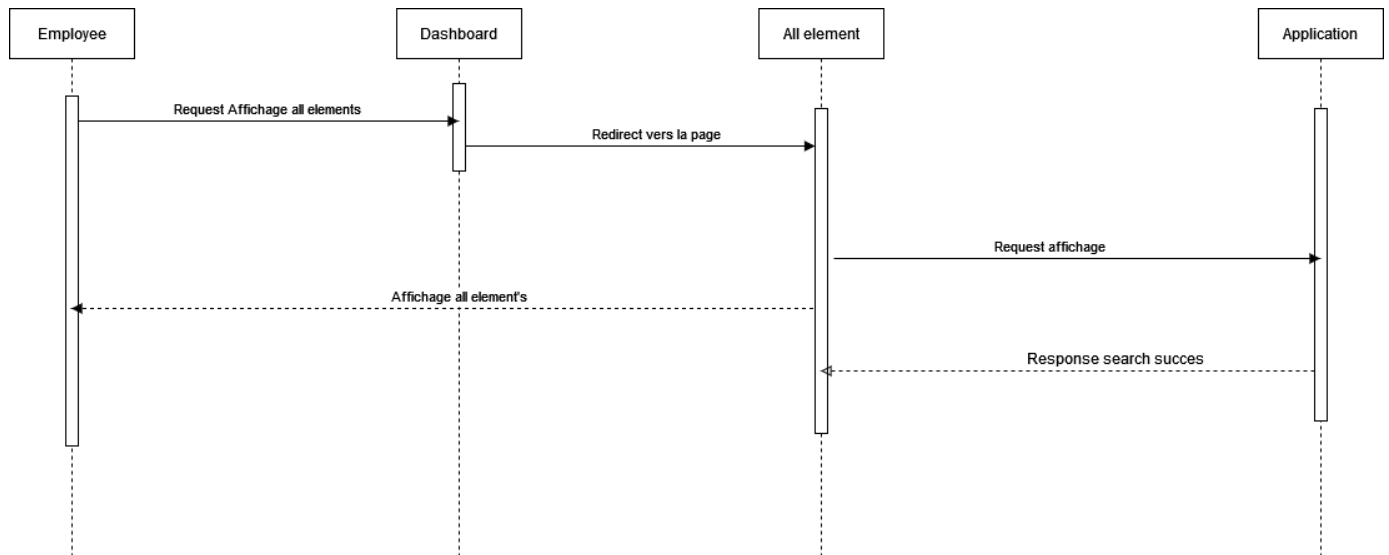
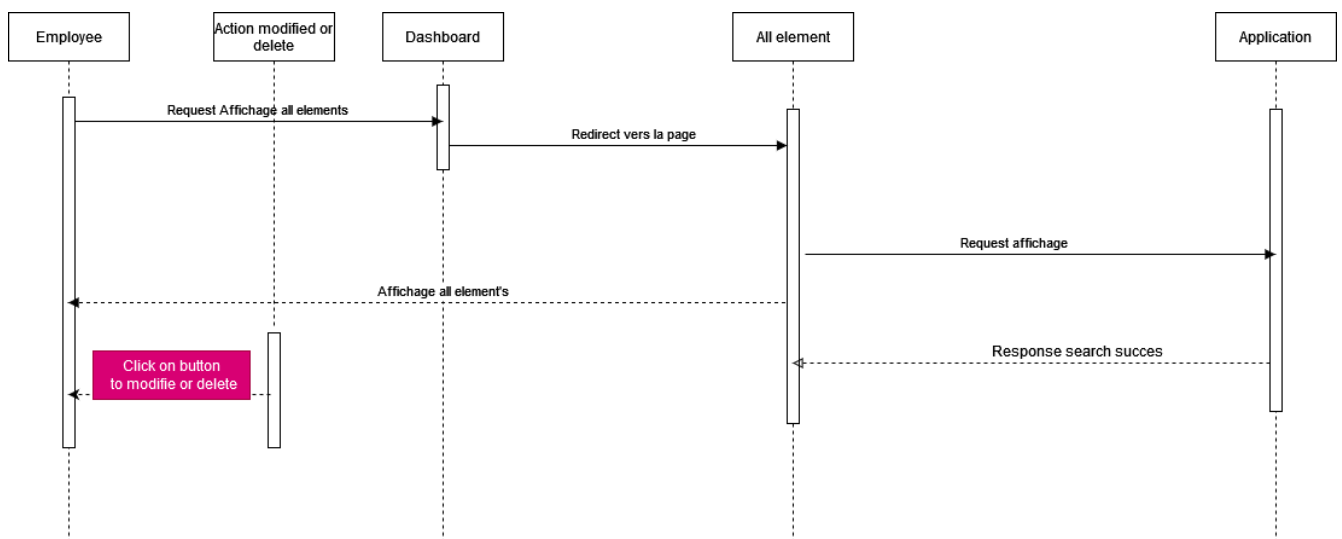


Diagramme des Séquences – Modifier ou Supprimer Un élément

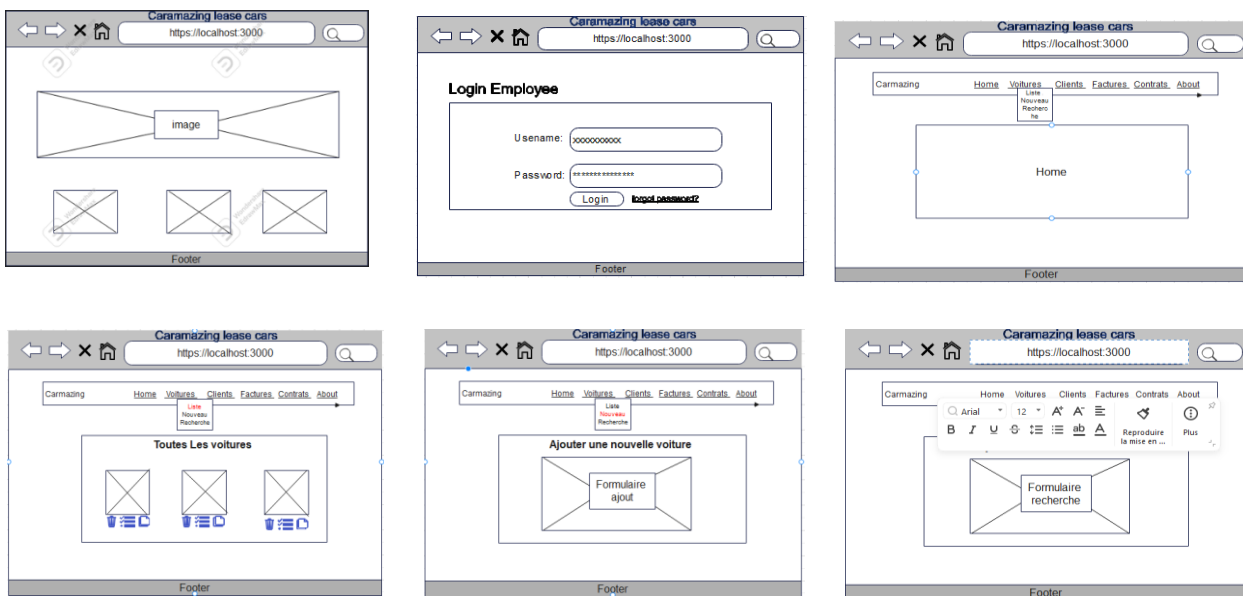


5.6 Maquettes des interfaces

Les maquettes sont indispensables pour visualiser les zones de texte, l'emplacement des images ou vidéos ainsi que la place des différents éléments graphiques.

Les éléments principaux que l'on retrouvera dans la maquette de l'application location de véhicule sont notamment :

- Le logo
- Un menu
- Un footer



6. Spécifications techniques du projet

La conception technique permet de représenter l'application du point de vue du développeur.

De ce fait, « L'architecture du projet » fera l'objet des technologies utilisées pour la réalisation du projet, à travers l'architecture Java EE, l'architecture 3-tiers et MVC.

6.1 L'architecture du Projet

Pour Le développement de notre web application, la solution Java EE avec Maven sera utilisée.

6.2 L'architecture J2EE (Java Enterprise Edition)

SUN Microsystems (racheté par Oracle Corporation) est à l'origine d'un ensemble de techniques informatiques permettant de réaliser des applications Web reposant sur le langage de programmation Java.

Java EE (Java Enterprise Edition) est une plateforme de développement, de déploiement et d'exécution d'application Web regroupant l'ensemble de ces techniques informatiques tels que des programmes Java (servlets) qui permettent de créer dynamiquement des données du côté serveur. Un ensemble des services nommé API (Application Programming Interface) qui consiste à fournir des fonctionnalités aux applications développées en langage de programmation Java.

Cette technologie est essentiellement utilisée pour la mise en place d'application Web dont l'architecture est basée sur l'architecture trois tiers.

6.3 Maven

Un outil d'automatisation de la construction pour les projets Java. Maven est un puissant outil de gestion de projet basé sur POM (modèle d'objet de projet). Il est utilisé pour la construction de projets, les dépendances et la documentation.

Maven facilite le travail quotidien des développeurs Java.

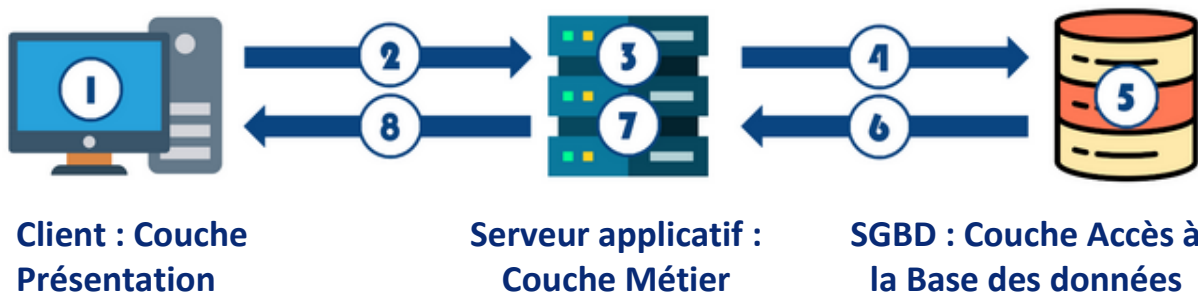
- ✓ Nous pouvons facilement construire un projet en utilisant maven.
- ✓ Nous pouvons facilement ajouter des jars et d'autres dépendances du projet à l'aide de Maven.
- ✓ Maven fournit des informations sur le projet (document de journal, liste de dépendances, rapports de tests unitaires, etc.)
- ✓ Maven est très utile pour un projet lors de la mise à jour du référentiel central des fichiers JAR et d'autres dépendances.
- ✓ Avec l'aide de Maven, nous pouvons créer n'importe quel nombre de projets
- ✓ Dans des types de sortie tels que JAR, WAR, etc. sans faire de script.
- ✓ En utilisant Maven, nous pouvons facilement intégrer notre projet avec un système de contrôle de code source (tel que Subversion ou Git).

6.4 L'architecture 3-Tiers

L'architecture 3 tiers, est une architecture logicielle qui divise notre application en trois niveaux/couches logiques et physiques : la couche présentation, la couche application

et la couche accès aux données. Cette architecture est très répandue surtout pour les applications client-serveur traditionnelles, et elle comprend 8 étapes :

- ✓ L'identification du serveur web par le client
- ✓ L'envoi de sa requête HTTP au serveur web
- ✓ Le traitement de la requête par le serveur web
- ✓ La requête du serveur web vers la base de données afin de récupérer les données nécessaires pour la construction de la page
- ✓ La sélection des données par le moteur de la base
- ✓ L'envoi des données au serveur web par la base
- ✓ La construction de la page par le serveur web
- ✓ L'envoi du résultat dans une réponse HTTP au client.



Présentation des données : La couche présentation correspond à la partie de l'application visible et interactive avec les utilisateurs. Elle relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour, lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche métier.

Traitements métiers : La couche métier correspond à la partie fonctionnelle de l'application, responsable de l'implémentation de la « logique ». Elle décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation. Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche. La couche métier offre des services applicatifs et métiers à la couche présentation. Pour fournir ces services, elle s'appuie sur les données du système, accessibles au travers des services de la couche d'accès aux données. En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

D'accès aux données : Elle consiste en la partie gérant l'accès aux données du système. Ces données peuvent être propres au système, ou gérées par un autre système. La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents **pour elle ; et elle**

accède aux données de manière uniforme. Le design pattern Modèle-Vue-Contrôleur (MVC) permet de traiter ces trois couches.

6.5 Méthodologie d'implémentation MVC

Méthodologie d'implémentation – MVC (Model View Controller) C'est un modèle de conception logiciel pour le développement d'applications Web.

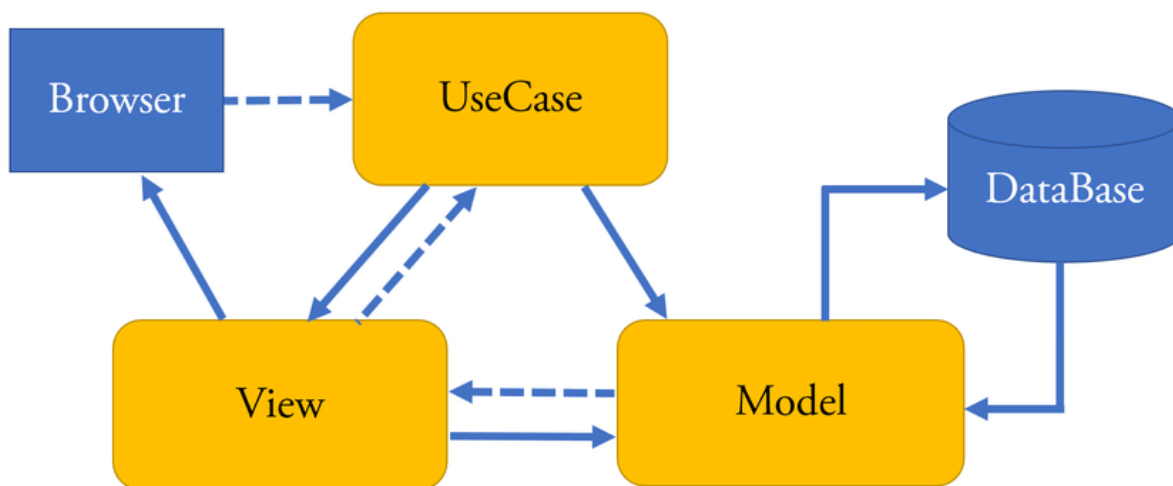
Le modèle MVC permet de séparer le code en 3 parties en Modèle, Contrôleur et Vue. Ces 3 parties sont interconnectées.

Modèle : Il contient toutes les fonctions nécessaires pour l'affichage des informations sur site. Il est utilisé pour faire la connexion avec la base de données.

Contrôleur : Il agit comme une interface entre le modèle et la vue. Il récupère toutes ces informations à partir du modèle pour le renvoyer vers la vue. Simultanément, il récupère les informations rentrées dans la vue vers le modèle pour enregistrer dans la base de données.


Vue : C'est l'interface utilisateur, qui contient toutes les informations qu'un utilisateur peut voir. Il peut entrer des informations à partir de cette interface.

Le schéma ci-dessous montre les interactions entre chaque partie.










6.6 Choix techniques



Logiciel Utilisés

	Eclipse est un environnement de développement intégré (IDE) utilisé dans la programmation informatique. Eclipse est écrit principalement en
---	---



	Java et son utilisation principale est pour le développement d'applications Java.
	MySQL est un système de gestion de base de données qui vous permet de gérer des bases de données relationnelles. C'est un logiciel open source soutenu par Oracle.
 Visual Studio Code	Visual Studio Code est un éditeur de code redéfini et optimisé pour la création et le débogage d'applications Web et cloud modernes. Visual Studio Code est gratuit et disponible sur votre plateforme préférée : Linux, macOS et Windows.
	C'est un système de contrôle de version open source.
 Balsamiq	C'est un logiciel de conception de wireframes qui permet aux équipes de créer des maquettes.
 Jenkins	Jenkins est un outil logiciel open source d'intégration continue écrit en Java.
 Adobe Illustrator CC	Adobe Illustrator est un logiciel de création graphique vectorielle. Il fait partie de la gamme Adobe, peut être utilisé indépendamment ou en complément de Photoshop, et offre des outils de dessin vectoriel puissants. Les images vectorielles sont constituées de courbes générées par des formules mathématiques.

Langages utilisés

	J2EE est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées.
	Java Spring Framework (Spring Framework) est un framework d'entreprise open source populaire permettant de créer des applications autonomes de niveau production qui s'exécutent sur la machine virtuelle Java (JVM).
	Maven est un outil de construction de projets (build) open source développée par la fondation Apache. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java.
	React est une collection JavaScript open-source connue pour construire des interfaces utilisateur très réactives et dynamiques.
	Hyper Text Markup Language est un langage de balisage employé pour structurer une page web et son contenu.
	Cascading Style Sheet est un langage informatique servant à décrire la présentation et le style d'un document HTML et XML.
	C'est un langage de script populaire permettant d'ajouter des fonctionnalités interactives et d'autres contenus web dynamiques aux pages web.

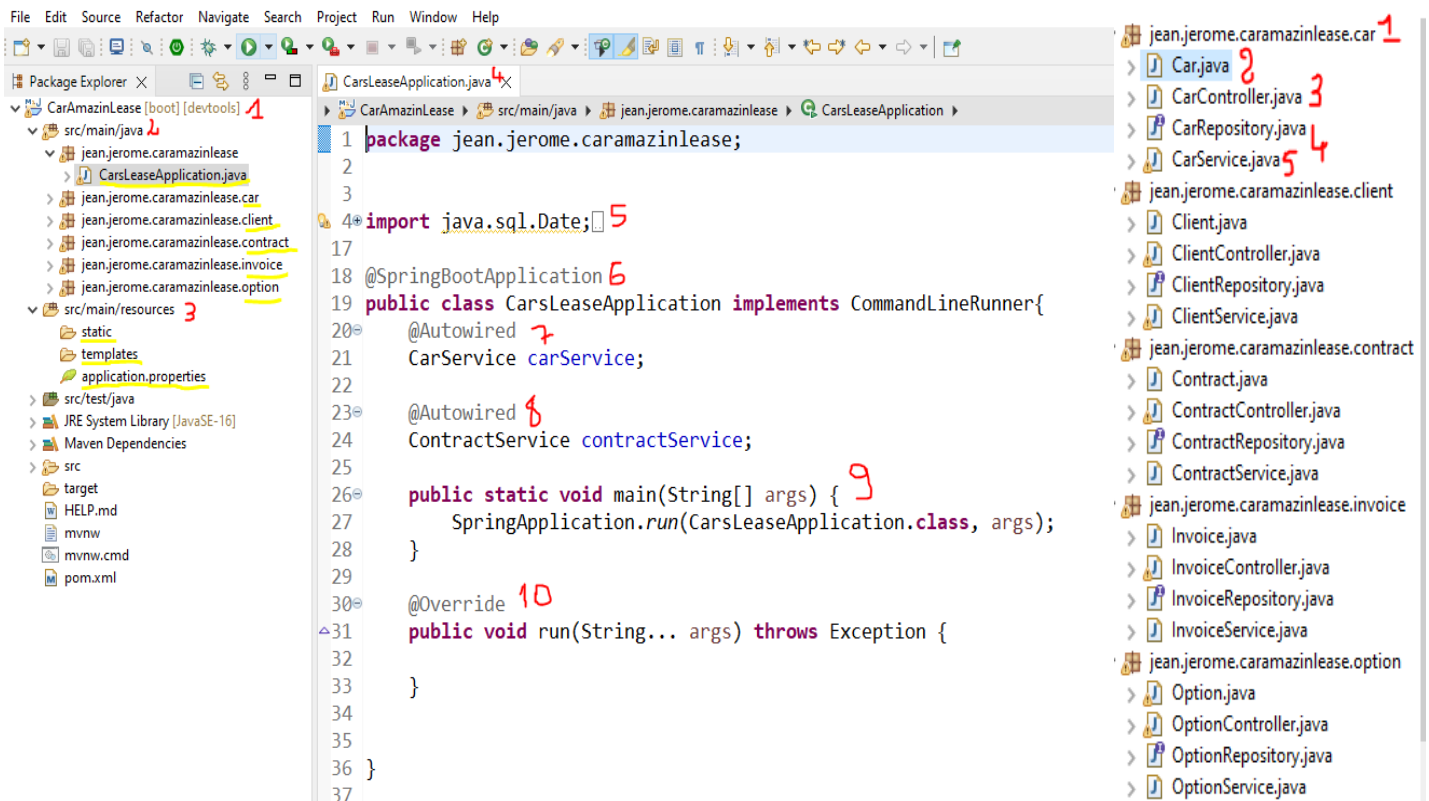
	<p>Structured Query Language(SQL) permet de manipuler les données et les BD relationnelles.</p>
	<p>Bootstrap est un framework CSS qui organise et gère la mise en page d'un site web. Le but de Bootstrap est de permettre, par exemple, de rendre facilement un site responsive design</p>

Outils de gestion de projet

	<p>UML (Unified Modeling Language) est un langage de modélisation visuelle commun, qui combine les notions orientées objets.</p>
	<p>Une plateforme pour créer des diagrammes et des organigrammes. Un outil très flexible</p>

7. Les extraits de code les plus significatifs

- L'ensemble du code Java avec sprint boot qui a permis de mettre en place le serveur de l'application caramazinLease



- 1 -> Le Spring starter Projet : est l'ensemble des champs du projet Maven créé avec sprint boot (framework Java)
- 2,9->Le main/Java : est le point de départ pour que la JVM(Java virtual machine) démarre l'exécution d'un programme Java. Sans la méthode main (), la JVM n'exécutera pas le programme.
- 3->Le main/ressources : Le dossier resources est l'emplacement par défaut où de nombreuses bibliothèques Java stockent leurs fichiers de configuration, généralement basés sur XML, par exemple la bibliothèque de journalisation Logback stocke
- 4->La classe Application.java : C'est la classe qui contient le main qui utilise la méthode SpringApplication.run () de Spring Boot pour lancer une application.
- 5->Import : c'est l'ensemble des imports de bibliothèques java
- 6->SpringBootApplication : est une annotation pratique introduite à partir de Spring Boot 1.2.0.
- 7,8->Autowired : L'annotation @Autowired peut être utilisée pour câbler automatiquement la méthode setter tout comme @Required annotation,

constructeur, une propriété ou des méthodes avec des noms arbitraires et/ou plusieurs arguments.

9. voir la 1

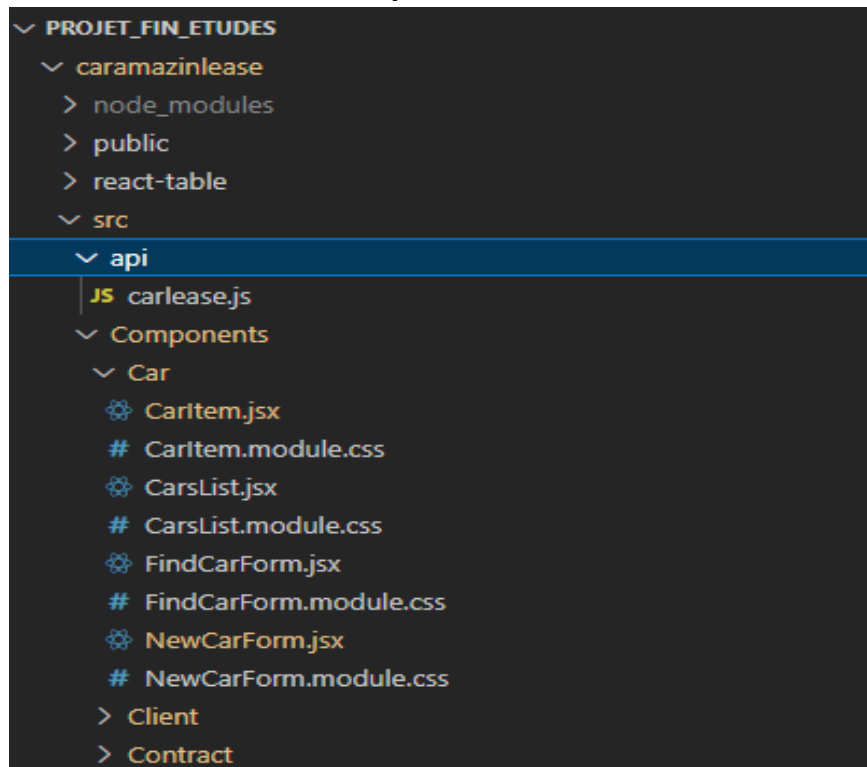
10. L'annotation Override : L'annotation `@Override` indique au compilateur Java que nous voulons surcharger une méthode de la superclasse.

➤ L'ensemble du code React avec tous les composants

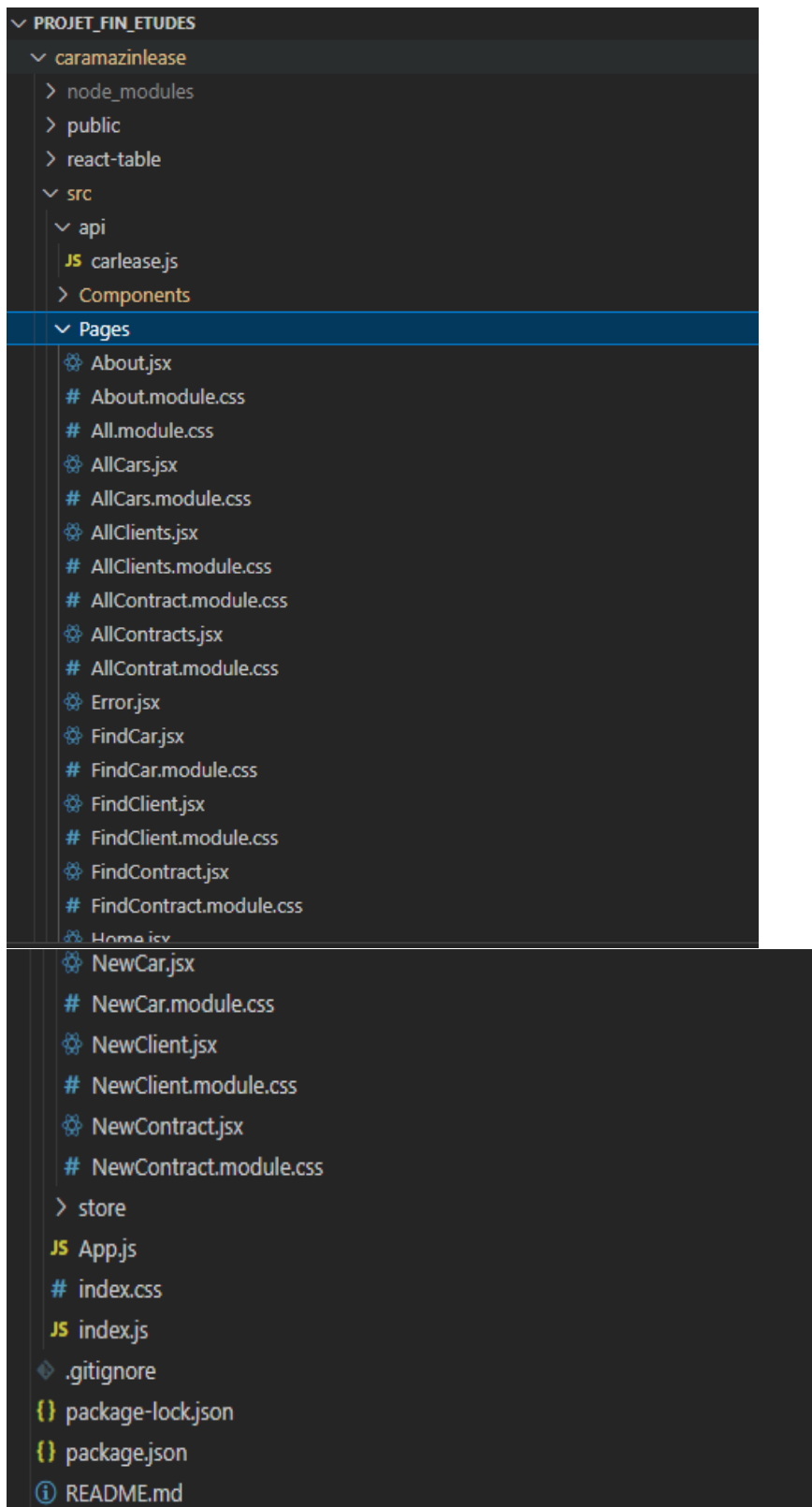
React n'a pas d'opinion sur la manière dont vous ordonnez vos fichiers à l'intérieur de vos dossiers. Ceci dit, vous souhaitez peut-être envisager l'une des approches populaires de l'écosystème.

Cependant on classer les dossiers et le code par :

▪ Par fonctionnalité ou par route



- par type de fichier



Afficher les éléments d'une classe, exemple des voitures

Code java

```
1 package jean.jerome.caramazinlease.car;
2
3 import java.util.List;
4
5 @Service
6 public class CarService {
7     @Autowired
8     CarRepository carRepository;
9     //carRepository = new Hibernate();
10
11     public Car save(Car car) {}
12
13     public List<Car> getAll() {
14         return carRepository.findAll();
15     }
16 }
```

Code react

```
import React from "react";
import CarItem from "../CarItem";
import styles from "../CarsList.module.css";
import api from "../../api/carlease";

function CarsList(props) {
    const deleteCar = async (id) => {
        try {
            await api.delete("/cars/" + id);
            props.refresh();
        } catch (error) {
            console.log(error);
        }
    };

    return (
        <div className={styles["cars-container"]} >
            {props.cars.map((car) => {
                return <CarItem car={car} key={car.id} onDelete={deleteCar} />;
            })}
        </div>
    );
}

export default CarsList;
```

Ajout un élément, exemple une voiture

Code java

```
package jean.jerome.caramazinlease.car;

import java.util.List;

@Service
public class CarService {
    @Autowired
    CarRepository carRepository;
    //carRepository = new Hibernate();

    public Car save(Car car) {
        return carRepository.save(car);
    }

    public List<Car> getAll(){
        return carRepository.findAll();
    }
}
```

Code react

```
import React, { useContext, useState } from "react";
import styles from "../NewCarForm.module.css";
import { useRef } from "react";
import { useNavigate } from "react-router-dom";
import api from "../../api/carlease";
import MainContext from "../../store/Main";

function NewCarForm() {
    let navigate = useNavigate();
    const context = useContext(MainContext);
    const car = context.car;
    console.log(car);
    const registrationInputRef = useRef();
    const brandInputRef = useRef();
    const colorInputRef = useRef("");
    const fuelInputRef = useRef("");
    const powerInputRef = useRef("");
    const maxSpeedInputRef = useRef("");
    const kmInputRef = useRef("");
    const firstUseInputRef = useRef("");
    const [inUse, setInUse] = useState(false);

    const submitHandle = async (e) => {
        e.preventDefault();
        const registrationValue = registrationInputRef.current.value;
        const brandValue = brandInputRef.current.value;
        const colorValue = colorInputRef.current.value;
        const fuelValue = fuelInputRef.current.value;
        const powerValue = powerInputRef.current.value;
        const maxSpeedValue = maxSpeedInputRef.current.value;
        const kmValue = kmInputRef.current.value;
        const firstUseValue = firstUseInputRef.current.value;
```

Rechercher un élément, exemple une voiture

Code java

```
1 package jean.jerome.caramazinlease.car;
2
3 import java.util.List;
4
5 @Service
6 public class CarService {
7     @Autowired
8     CarRepository carRepository;
9     //carRepository = new Hibernate();
10
11     public Car save(Car car) {}
12
13     public List<Car> getAll(){}
14
15     public Car getById(long id) {
16         return carRepository.findById(id).orElse(null);
17     }
18 }
```

Code react

```
import React, { useRef, useContext } from "react";
import api from "../../api/carlease";
import styles from "../FindCarForm.module.css";
import MainContext from "../../store/Main";

function FindCarForm(props) {
    const context = useContext(MainContext);
    const idInputRef = useRef("");

    const submitHandler = async (event) => {
        event.preventDefault();

        const idValue = idInputRef.current.value;

        try {
            const response = await api.get("/cars/" + idValue);
            if (response.data) {
                props.setCar(response.data);
                context.setCar(response.data);
            }
        } catch (error) {
            props.setCar(null);
            context.setCar(null);
        }
    };

    return (
        <div className={styles["form-container"]} >
            <form onSubmit={submitHandler}>
                <div className={styles["input-group"]} >
                    <label htmlFor="id">id</label>
                    <input type="text" name="id" id="id" required ref={idInputRef} />
                </div>

                <div className={styles["submit-container"]} >
                    <input
                        type="submit"
                        name="submit-btn"
                        id="submit-btn"
                        value="Chercher"
                    />
                </div>
            </form>
        </div>
    );
}

export default FindCarForm;
```

Supprimer un élément, exemple une voiture

code java

```
public boolean deleteById(long id) {
    try {
        carRepository.deleteById(id);
        return true;
    } catch (Exception e) {
        return false;
    }
}
```

Code react

```
function CarsList(props) {
    const deleteCar = async (id) => {
        try {
            await api.delete("/cars/" + id);
            props.refresh();
        } catch (error) {
            console.log(error);
        }
    };

    return (
        <div className={styles["cars-container"]} >
            {props.cars.map((car) => {
                return <CarItem car={car} key={car.id} onDelete={deleteCar} />;
            })}
        </div>
    );
}
```

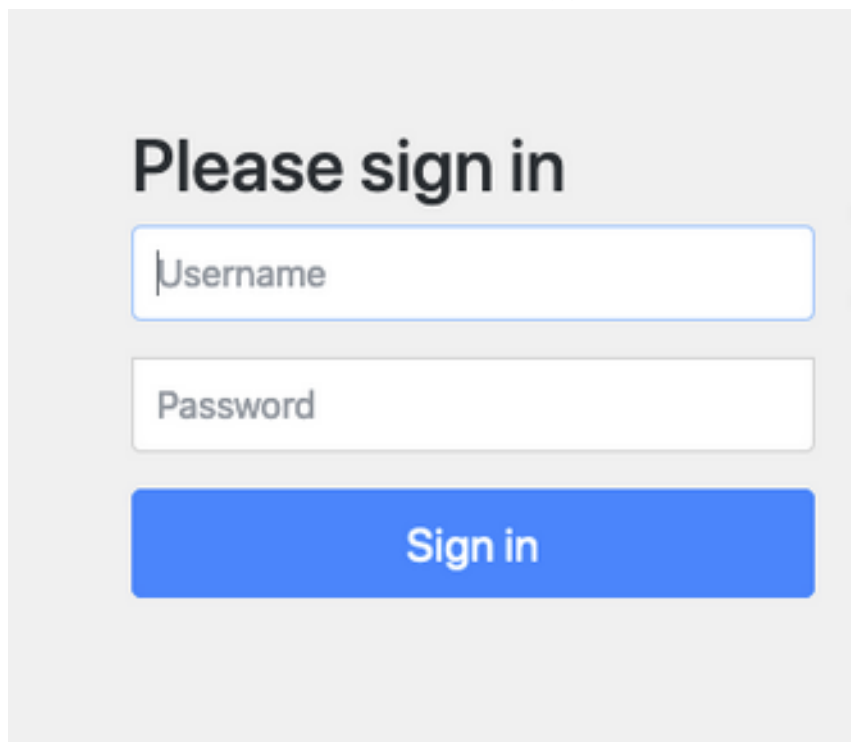
8.Présentation du jeu d’essai élaboré

La réalisation du projet consiste dans un premier temps à traduire en langage informatique les concepts qui ont été élaborés pendant la phase de conception. Les langages informatiques « Java », « Javascript », « HTML » et SQL ont été utilisés pour le développement du projet. Les modules développés et testés seront livrés aux utilisateurs afin que ces derniers puissent les valider.

8.1 Module 1 : Page de connexion

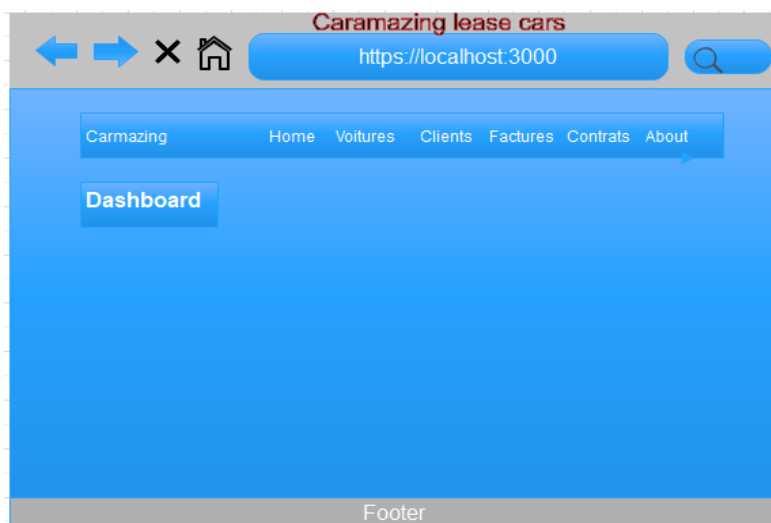
La page de connexion propose à l'utilisateur. Ce dernier devra saisir le couple identifiant et mot de passe.

Seuls, les utilisateurs habilités auront accès à l'application.

The image shows a login interface with a light gray background. At the top, the text 'Please sign in' is displayed in a bold, dark font. Below this, there are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. Both fields have a light gray border and a small blue outline. Below the password field is a blue button with the text 'Sign in' in white. The entire form is centered on the page.

Après la vérification d’identifiant et le mot de passe, l’application affichera soit, un message d’erreur en cas de saisie erronée du mot de passe ou de l’identifiant, ou la page d’accueil pour les utilisateurs habilités

8.2 Module 2 : Dashboard



9. Description de la veille, sur les vulnérabilités de sécurité

La veille en vulnérabilités est la méthode la plus efficace. Elle consiste à suivre quotidiennement la découverte de nouvelles failles de sécurité, **mais surtout elle** s'inscrit dans un processus global : détection, alerte, remédiation et suivi.

9.1 La gestion des vulnérabilités

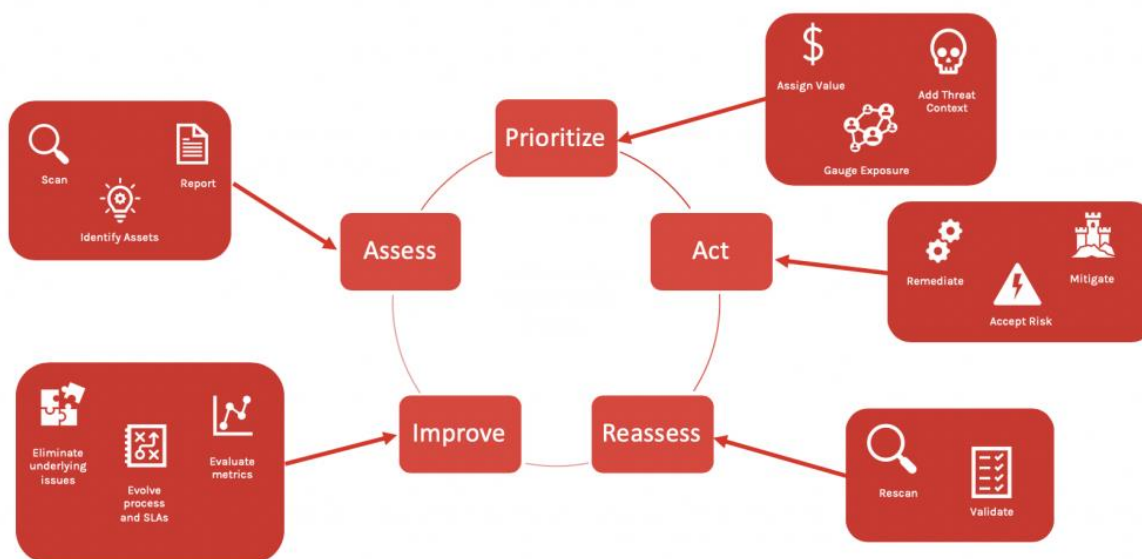
La gestion des vulnérabilités est un processus régulier et continu consistant à identifier, évaluer, signaler, gérer et corriger les vulnérabilités de sécurité des endpoints, des workloads et des systèmes. En général, les équipes de sécurité utilisent un outil de gestion des vulnérabilités pour détecter les vulnérabilités et mettent en place différents processus pour les corriger.

Un programme de gestion des vulnérabilités efficace s'appuie sur la cyber veille et la connaissance des opérations informatiques et commerciales pour prioriser les risques et corriger les vulnérabilités au plus vite.

9.2 Les cinq étapes du cycle de gestion des vulnérabilités

- Étape 1. Évaluer
- Étape 2. Prioriser
- Étape 3. Agir

- Étape 4. Réévaluer
- Étape 5. Perfectionner



<https://www.crowdstrike.fr/cybersecurity-101/vulnerability-management/>

Et la société IBM Security X-Force a classé le top 10 des CVE de 2020 en fonction de la fréquence à laquelle les acteurs de la menace les ont exploités ou ont tenté de les exploiter

- CVE-2019-19871: Citrix Application Delivery Controller (ADC)
- CVE-2018-20062: Aucun Exécution de code à distance CMS ThinkPHP
- CVE-2006-1547: ActionForm dans Apache Software Foundation (SAF) Struts
- CVE-2012-0391: composant ExceptionDelegator dans Apache Struts
- CVE-2014-6271: Injection de commandes GNU Bash
- CVE-2019-0708: «Bluekeep» Microsoft Remote Desktop Services Remote Code Execution
- CVE-2020-8515: injection de commande Draytek Vigor
- CVE-2018-13382 et CVE-2018-13379: autorisation incorrecte et traversée de chemin dans Fortinet FortiOS
- CVE-2018-11776: Exécution de code à distance Apache Struts
- CVE-2020-5722: HTTP: Injection SQL Grandstream UCM6200

9.3 Les mesures de sécurité pour les applications web

Il existe trois principes de sécurité qui peuvent être appliqués pour garantir la sécurité d'une application ou d'une infrastructure.

1. La confidentialité : C'est l'assurance que les personnes non autorisées n'accèdent pas à des informations sensibles.
2. L'intégrité : Elle permet d'être sûr que les données sont fiables et n'ont pas été modifiées par des personnes non autorisées.
3. La disponibilité : C'est l'assurance qu'il n'y a pas de perturbation d'un service ou de l'accessibilité aux données.

La sécurité de l'information repose sur l'équilibre entre ces trois principes. Ces principes fondamentaux sont des éléments clés dans l'élaboration des politiques de sécurité en entreprise.

9.4 les moyens pour sécuriser son Système d'Information ?

1. Former ses équipes informatiques.
2. Tester ses utilisateurs.
3. Effectuer un audit de sécurité
4. Appliquer un système de mots de passe robustes.
5. Sécuriser sa messagerie des cyberattaques.

10. Description d'une situation de travail

10.1 Spring boot Sécurité

Alors que de plus en plus de sociétés ont recours au e-commerce pour faire vivre leur activité, un nombre croissant de données exclusives et d'informations personnellement identifiables sont transférées de serveur en serveur partout dans le monde.

Ces données devraient être protégées.

Toutefois, de nombreux développeurs et sociétés ne savent pas comment correctement les sécuriser lorsqu'elles sont stockées ou en transmission. En d'autres termes : les applications web sont vulnérables face à des attaques toujours plus perfectionnées, à moins d'avoir recours à des mesures spécifiques pour les éviter.

Un exemple :

Au début du mois de juillet 2021, la société américaine de logiciel **Kaseya** a été victime d'une **cyberattaque par ransomware** qui a affecté environ 1 500 entreprises à travers le monde. La plupart des 800 supermarchés Coop de Suède ont dû fermer leurs portes pendant plusieurs jours. Le gang REvil avait réussi à exploiter une **faille informatique** jusqu'ici inconnue, présente sur VSA, le logiciel phare de l'entreprise. La société a finalement obtenu un outil de déchiffrement pour déverrouiller les réseaux des entreprises clientes touchées, mais n'a pas indiqué si elle avait payé les 70 millions de dollars en bitcoins exigés par les pirates informatiques.

L'injection de code représente un autre type d'attaque. Selon les recherches, celle-ci occupe la première place des attaques les plus communes à l'encontre des applications web, l'injection SQL étant la plus populaire.

Il est donc primordial de porter attention à la sécurité. En tant que développeur Java, on doit prévenir les piratages informatiques.

Spring Security est un framework garantissant une solution facile d'utilisation pour les applications Java, en particulier avec Spring Boot. Il permet de configurer Spring pour prévenir les attaques en ayant recours à très peu, voire à aucune configuration.

À grande échelle, l'application Spring est composée de plusieurs **modules** qui fonctionnent indépendamment les uns des autres. Spring Security est l'un de ces modules. Les **modules Spring** sont comme des récipients fermés, permettant à d'autres modules Spring de fonctionner ensemble, sans se bloquer mutuellement.

Lorsque vous ajoutez Spring Security, son premier objectif est de protéger les **requêtes HTTP** traitées sur votre application. En effet, chaque fois qu'un utilisateur clique sur un bouton ou qu'une information est transférée d'une partie à l'autre de l'application web, une requête HTTP est envoyée. Celles-ci sont utiles à la sécurisation.

Ces requêtes HTTP adressées à votre application web traversent différents niveaux de protection après l'installation de Spring Security :

1. **Un pare-feu HTTP.**
2. **Un proxy.**

3. Des filtres.

Explorons ces trois niveaux plus en profondeur.

1er niveau : le pare-feu http

Le **pare-feu HTTP** se trouve en première ligne. C'est un dispositif qui filtre le flux de communication, en étant très sélectif sur les informations qui pénètrent dans l'application.



2e niveau : le proxy

En deuxième lieu se trouve le proxy, une expression de geek faisant référence à une autorité extérieure qui gère les accès à une source protégée : votre application web.

Il se charge de classer le trafic HTTP, et le dirige vers les filtres de servlet appropriés dans la chaîne de filtres.



3e niveau : les filtres

Les **filtres** s'assurent que toutes les **requêtes HTTP** qui pénètrent dans votre application web sont sécurisées. Chaque filtre fournit une configuration de sécurité que vous pouvez intégrer à votre application web. Cette collection de filtres servlet implantés est appelée **chaîne de filtres de Spring Security**. 😊



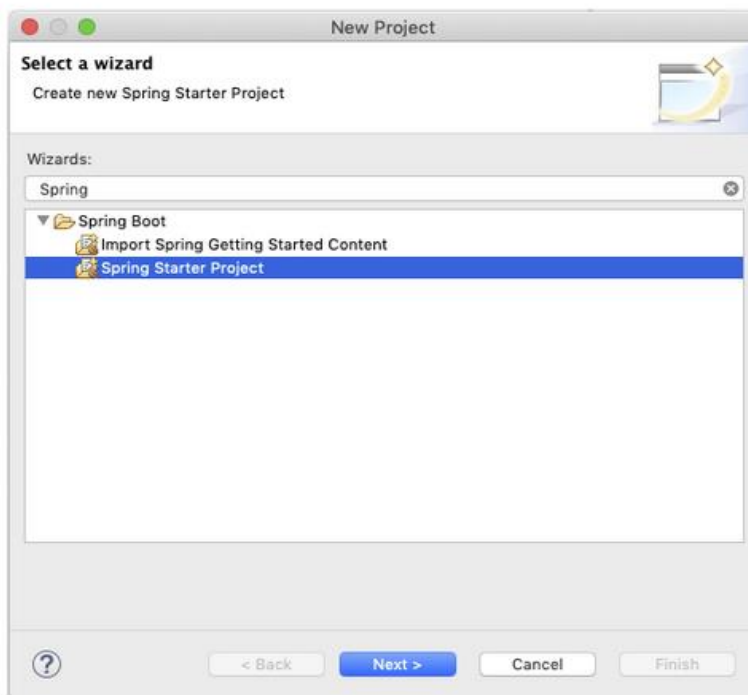
Lorsque HTTP formule sa requête depuis le front-end et passe le premier pare-feu, `DelegatingFilterProxy` s'occupe de classer le trafic HTTP, et de l'envoyer vers les filtres de la chaîne de Spring Security prenant en charge les informations de connexion. Tout est donc bien contrôlé au bon endroit.

En résumé, Spring Security sécurise les requêtes HTTP de votre application web en les faisant passer par trois niveaux :

- Premièrement, le **pare-feu HTTP** bloque les requêtes suspectes.
- Deuxièmement, le **proxy** (`DelegatingFilterProxy`) prend en charge le reste des requêtes HTTP, et les envoie vers la chaîne de filtres de Spring Security.
- Enfin, les **filtres de la chaîne** s'assurent que ces requêtes HTTP soient conformes à leurs critères de sécurité.

Création du projet spring boot sécurité

- Choisissez le **Spring Starter Project** pour commencer **Spring Initializr**, et cliquez sur *Next*.



Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

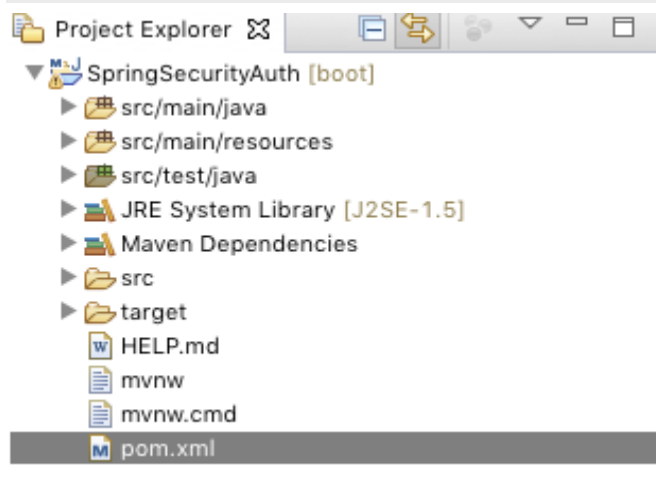
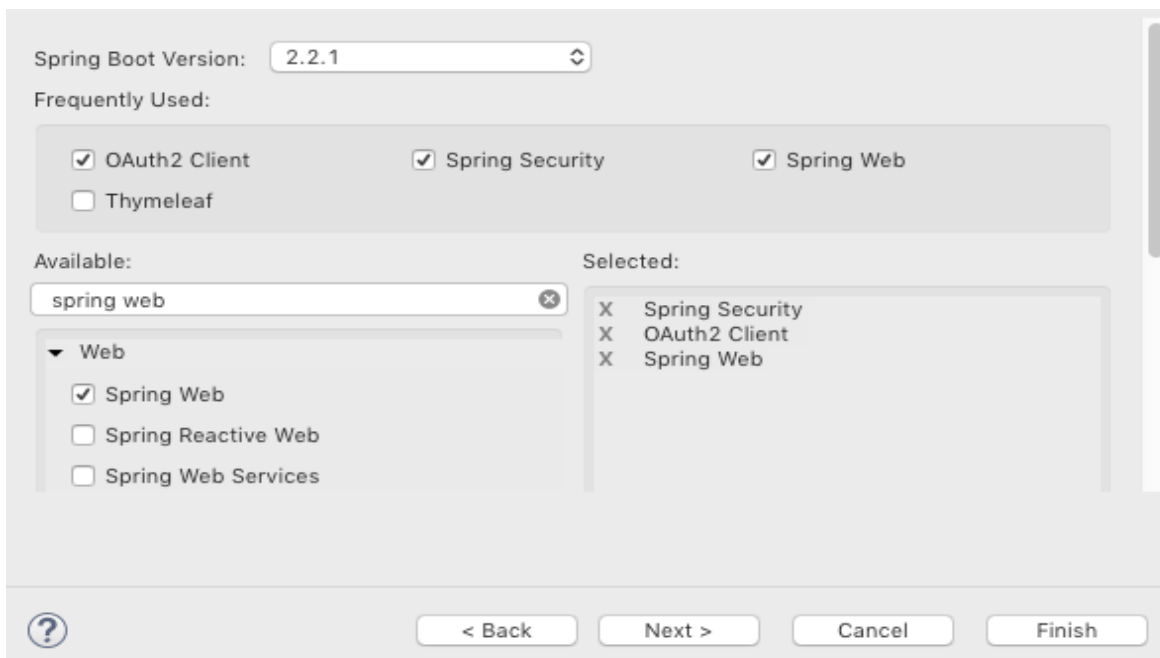
Package:

Working sets

☐ Add project to working sets

Working sets:

< Back Next > Cancel Finish



Si vous avez appelé votre application **SpringSecurityAuth**, la classe principale s'intitulera **SpringSecurityAuthApplication**. Lorsque vous l'ouvrirez, elle ressemblera à ça :



- L'**architecture Spring** se sert de Spring Security pour ajouter trois niveaux de sécurité à votre application web :

- Le **pare-feu HTTP** empêche les flux suspects de pénétrer dans votre application ;
- **DelegatingFilterProxy** dirige le reste des flux HTTP vers les filtres de sécurité appropriés ;
- **La chaîne de filtres sécurisée** héberge les règles de sécurité pour votre application web.
- **Spring Initializr** est un outil qui sert à configurer une application web Spring Boot.
- Les **dépendances** sont les principales librairies de code que vous pouvez importer dans votre application web.
- Nous avons ajouté les dépendances **Spring Web**, **Spring Security**, et **OAuth 2.0**.

L'utilisation de Spring Security implique de sécuriser un dispositif. En tant que développeur Java, vous aurez forcément à sécuriser à un moment ou un autre un formulaire de connexion ; nous nous entraînerons sur ce point au prochain chapitre.

Authentification et Autorisation

En vous connectant à votre espace personnel sur un site, l'application web vous **authentifie**, et garantit que vous êtes-vous. Lorsque vous consultez une page sur votre compte – vos relevés bancaires, par exemple – l'application vous **autorise** à accéder à cette page. Ces deux principes – l'authentification et l'autorisation – constituent le cœur de **Spring Security**. Si vous parvenez à sécuriser ces processus, vous avez parcouru la moitié du chemin.

En principe, vous vous authentifiez sur une application web grâce à votre nom d'utilisateur et votre mot de passe. Un nom d'utilisateur vous identifie. Le mot de passe est uniquement connu par vous. Il s'agit d'une **authentification à facteur unique**, car votre identification ne relève que d'une seule information (votre mot de passe).

Parfois, il peut vous être demandé de fournir des éléments supplémentaires, comme des empreintes, une preuve physique ou un badge. Il s'agit alors d'une **authentification à facteurs multiples**, car vous devez, d'une part, connaître votre mot de passe, et d'autre part, prouver que vous êtes bien la personne que vous prétendez être.

Sélectionnez une option parmi les multiples formes d'authentification

Nous venons de voir qu'il existe différentes manières de s'authentifier :

- L'authentification à facteur unique, qui permet de se connecter grâce à un mot de passe ;

- Et l'authentification à facteurs multiples, qui requiert la livraison de plusieurs éléments pour prouver votre identité.

Eh bien, sachez que votre navigateur peut également gérer l'authentification de différentes manières :

- L'authentification **par session** ;
- L'authentification **par jeton** (ou *token*, en anglais).

En résumé

- **L'authentification** permet de s'assurer que l'utilisateur dispose des bonnes informations pour prouver qu'il est bien la personne qu'il prétend être.
- **L'autorisation** permet de s'assurer que l'utilisateur authentifié se rend uniquement sur les pages qu'il est autorisé à consulter.
- **L'authentification par session** utilise des cookies pour stocker les informations de l'utilisateur de la session. Ces cookies sont eux-mêmes stockés dans le navigateur de l'utilisateur, et sur le serveur d'autorisation.
- **L'authentification par token** utilise un JWT transmis par le serveur d'autorisation. Ce JWT est utilisé pour valider la connexion de l'utilisateur, et peut être stocké directement dans le navigateur, mais il est préférable qu'il soit stocké dans un token sécurisé du navigateur.

Configurez une page de connexion customisée avec Spring Security

Démarrez par le formulaire Spring Security par défaut, en utilisant la méthode `formLogin()` .

java

```
1 @Override
2 public void configure(HttpSecurity http) throws Exception {
3     http
4         .formLogin();
5 }
```

Spring Security fournit des filtres qui peuvent être utilisés pour authentifier différents types de rôles. Définissez les deux rôles suivants dans votre app Spring Boot :

1. Utilisateur.
2. Administrateur.

Une fois qu'ils sont authentifiés, Spring Security se charge d'octroyer les autorisations appropriées, et contrôle l'accès aux données selon les rôles.

En premier lieu, ajoutez la méthode `authorizeRequests()` pour définir les rôles.

Ensuite, ajoutez la méthode `antMatchers()` pour définir l'association des rôles `USER` (utilisateur) et `ADMIN` (administrateur) avec des pages.

Cela permettra d'assurer l'étape d'autorisation pour les pages qui sont à la disposition des utilisateurs. Le rôle `ADMIN` se voit attribuer une page d'accueil spécifique, `/admin`, mais a également accès à `/user`. Les utilisateurs ont accès à la page d'accueil `/user`, mais ne devraient pas avoir accès à la page d'accueil `/admin`. Ajoutez `anyRequest().authenticated()` pour vous permettre d'utiliser le formulaire ci-dessous pour l'authentification.

Procédez comme suit, en ajoutant l'extrait de code à votre méthode `configure()` qui gère les requêtes HTTP.

```
1 @Override
2 public void configure(HttpSecurity http) throws Exception {
3     http
4         .authorizeRequests()
5         .antMatchers("/admin").hasRole("ADMIN")
6         .antMatchers("/user").hasRole("USER")
7         .anyRequest().authenticated()
8         .and()
9         .formLogin();
10 }
```

Utilisez l'annotation `AuthenticationManagerBuilder` pour assigner les rôles d'utilisateur et d'administrateur.

vous utiliserez aussi `auth.inMemoryAuthentication`. Cela signifie que les identifiants créés seront stockés dans la mémoire, plutôt que dans un jeton ou dans une base de données.

Voici la librairie pour **`AuthenticationManagerBuilder`** :

```
import
org.springframework.security.config.annotation.authentication.builders.Authenti
cationManagerBuilder
```

java

```
1 @Override
2 protected void configure(AuthenticationManagerBuilder auth) throws Exception
3     auth.inMemoryAuthentication()
4         .withUser("springuser").password(passwordEncoder().encode("spring123")).roles("USER")
5         .and()
6         .withUser("springadmin").password(passwordEncoder().encode("admin123")).roles("ADMIN",
7         "USER");
7 }
```


Ajoutez le code suivant à votre classe **SpringSecurityConfig** :

java

```
1 @Bean
2 public PasswordEncoder passwordEncoder() {
3     return new BCryptPasswordEncoder();
4 }
```

Ensuite, vous allez pouvoir mettre en place votre contrôleur REST, qui se chargera de créer vos pages d'accueil. Pour garantir son fonctionnement, assurez-vous de disposer d'une version à jour de Maven, et d'avoir défini le Spring Boot Maven plugin en tant que dépendance. Eclipse rend cette procédure très facile :

- Référez-vous au fichier `pom.xml` qui contient vos dépendances.
- Faites un clic droit sur `pom.xml` -> *Maven* -> *Update project*.
- Vous devez également ajouter le Spring Boot Maven plugin pour un support web supplémentaire.
- Faites un clic droit sur `pom.xml` -> *Maven* -> *Add Plugin*.
- Tapez `spring-boot-maven-plugin`.

Ensuite, créez une méthode différente pour chaque rôle, en utilisant les classes

`@RolesAllowed` et `@RequestMapping` pour associer l'URL au rôle.

java

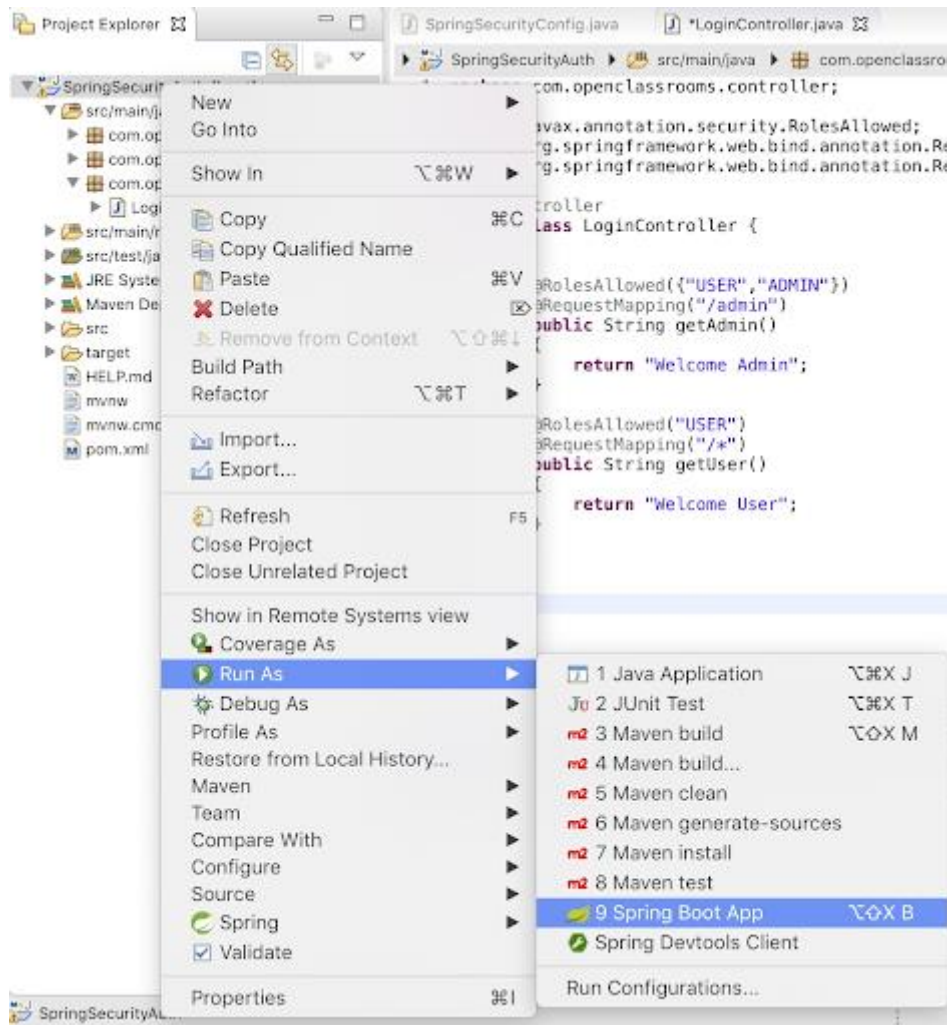
```
1 package com.test.controller;
2
3 import javax.annotation.security.RolesAllowed;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 public class LoginController
9
10 {
11     @RolesAllowed("USER")
12     @RequestMapping("/")
13     public String getUser()
14     {
15         return "Welcome User";
16     }
17
18     @RolesAllowed({"USER","ADMIN"})
19     @RequestMapping("/admin")
20     public String getAdmin()
21     {
22         return "Welcome Admin";
23     }
24 }
```

Ensuite mettez en place votre Serveur Web

Lorsque vous lancez votre app indépendante, vous pouvez retrouver votre page de connexion en utilisant **Apache Tomcat** en serveur web. Aucune configuration n'est

nécessaire ; tout s'effectue automatiquement à la compilation et à l'exécution du code.

- **Étape 1 :** sur Eclipse, rendez-vous sur votre dossier dans votre Project Explorer qui affiche le nom de votre app. La mention [boot] apparaît à côté, entre crochets. Cliquez sur *Run As -> Spring Boot App*.

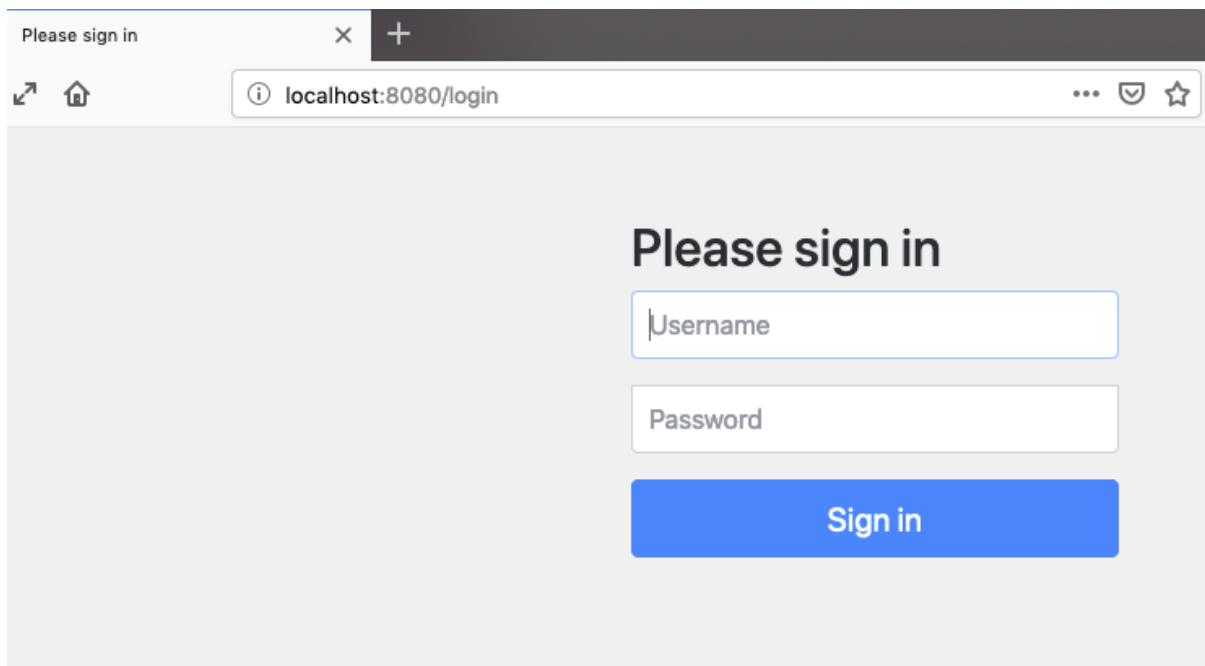


Exécutez SpringSecurityAuth

Vous constatez que votre code se compile dans votre console. Une fois qu'il est complété, vous pouvez ouvrir votre application web sur votre navigateur.

- **Étape 2 :** Ouvrez votre navigateur favori et tapez cette URL : <http://localhost:8080>.

Si vous voulez changer le port que vous utilisez en un autre, comme 8090, vous pouvez ajouter `server.port=8090` dans le fichier **application.properties** > **src/main/resources**.



Maintenant, connectez-vous en tant qu'user et voyez ce qu'il se passe. Le formulaire de connexion de Spring Security par défaut

Yes, ça fonctionne !

10.2 Spring boot sécurité avec OAuth 2.0

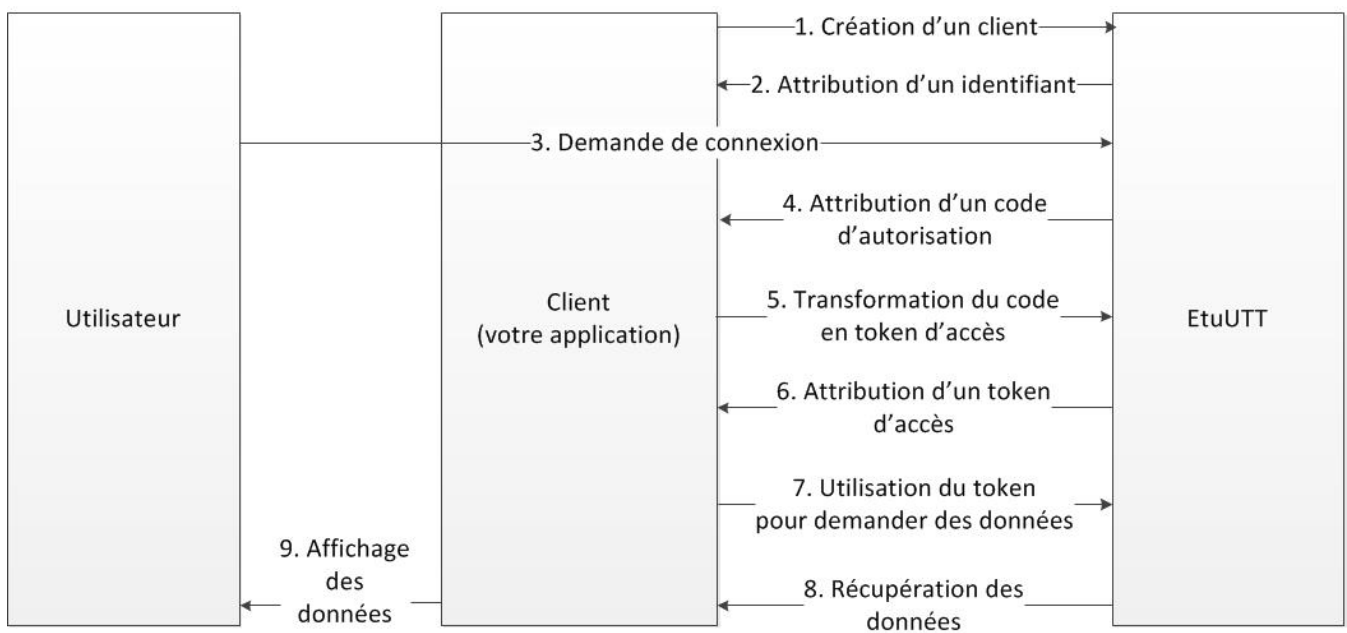
Généralement dans les entreprises de e-commerce, des clients se connectent et effectuent des achats en utilisant leurs données personnelles. Et La majorité des violations de données prennent pour cible les petites entreprises, car elles ne dédient généralement pas un budget important aux mesures de protection de données .

Heureusement, il existe une solution à ce problème : **OAuth** !

OAuth est un protocole d'autorisation permettant une connexion sécurisée, en utilisant des tokens d'encodage sans état pour sécuriser les sessions des utilisateurs sur une application web.

En clair, OAuth 2.0 permet à vos clients de créer un compte sur votre application web en se connectant sur un compte appartenant à une société vérifiée, comme Google, Facebook, Twitter, Okta ou GitHub. Vous vous êtes probablement déjà enregistré sur un site permettant de créer un profil via votre compte Google ou Facebook.

Les étapes de fonctionnement d'un processus classique OAUTH sont :



1. Vous déclarez votre application auprès d'EtuUTT en créant un client et en listant les droits dont vous avez besoin dans votre application (à quoi vous voulez accéder) ;
2. EtuUTT vous attribue alors un **client id** et un **client secret** qui vous serviront pour utiliser l'API ;
3. Au moment où l'utilisateur veut se connecter en utilisant votre application, vous le redirigez vers une adresse spécifique d'EtuUTT (que nous verrons plus en détail par la suite), et EtuUTT demandera à l'utilisateur connecté actuellement si il souhaite donner ses informations à votre application ;
4. Si l'utilisateur accepte, il est redirigé vers votre application avec un **code d'autorisation** ;
5. Ce **code d'autorisation** vous permet alors de demander un **token d'accès** qui sera votre clé pour utiliser concrètement l'API ;
6. Un **token d'accès** et un **refresh token** vous sont attribué : le premier vous sert à accéder à l'API et n'est valide qu'une heure, le deuxième vous permet de régénérer à volonté des tokens d'accès et est valide un mois ;
7. Vous dialoguez avec l'API grâce au **token d'accès** en lui demandant des données ...
8. ... qu'elle vous fournit si vous y avez accès ...
9. ... et que vous pouvez donc fournir à l'utilisateur ;

Exemple de connexion avec GitHub

- S'enregistrer sur GitHub pour obtenir un nom d'utilisateur et un mot de passe, c'est ce qui permettra de se connecter au serveur d'autorisation de GitHub avec Aouth 2.0. Pour cela se rendre sur [page d'enregistrement GitHub OAuth 2.0](#) .

Register a new OAuth application

Application name *

OAuth2OpenClassrooms

Something users will recognize and trust.

Homepage URL *

http://localhost:8080

The full URL to your application homepage.

Application description

course example

This is displayed to all users of your application.

Authorization callback URL *

http://localhost:8080/login/oauth2/code/github

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Register application

Cancel

Si tout fonctionne, vous obtiendrez vos propres identité client et client secret. Mettez-les de côté pour plus tard.

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

List this application in the Marketplace

6 users

Client ID

0f62e46d40e21a12d860

Client Secret

f8128d36f28d375182f2222b679dff8f541d3d37

Revoke all user tokens

Reset client secret

Application logo



Drag & drop

Upload new logo

You can also drag and drop a picture from your computer.

Application name *

OAuth2OpenClassrooms

Something users will recognize and trust.

Homepage URL *

http://localhost:8080/github

The full URL to your application homepage.

Application description

course example

This is displayed to all users of your application.

Authorization callback URL *

http://localhost:8080/login/oauth2/code/github

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Update application

Delete application

Voici les

liens pour [Google](#) et [Facebook](#).

➤ Définir GitHub en tant que page de connexion OAuth 2.0

```
*application.properties  ⓘ
1 spring.security.oauth2.client.registration.github.client-id=0f62e46d40e21a12d860
2 spring.security.oauth2.client.registration.github.client-secret=f8128d36f28d375182f2222b679dff8f541d3d37|
3
```

➤ Configurer OAuth 2.0 dans la page de connexion

```
1 @Override
2 public void configure(HttpSecurity http) throws Exception {
3     http
4         .authorizeRequests()
5         .antMatchers("/admin").hasRole("ADMIN")
6         .antMatchers("/user").hasRole("USER")
7         .anyRequest().authenticated()
8         .and()
9         .formLogin()
10        .and()
11        .oauth2Login();
12 }
```

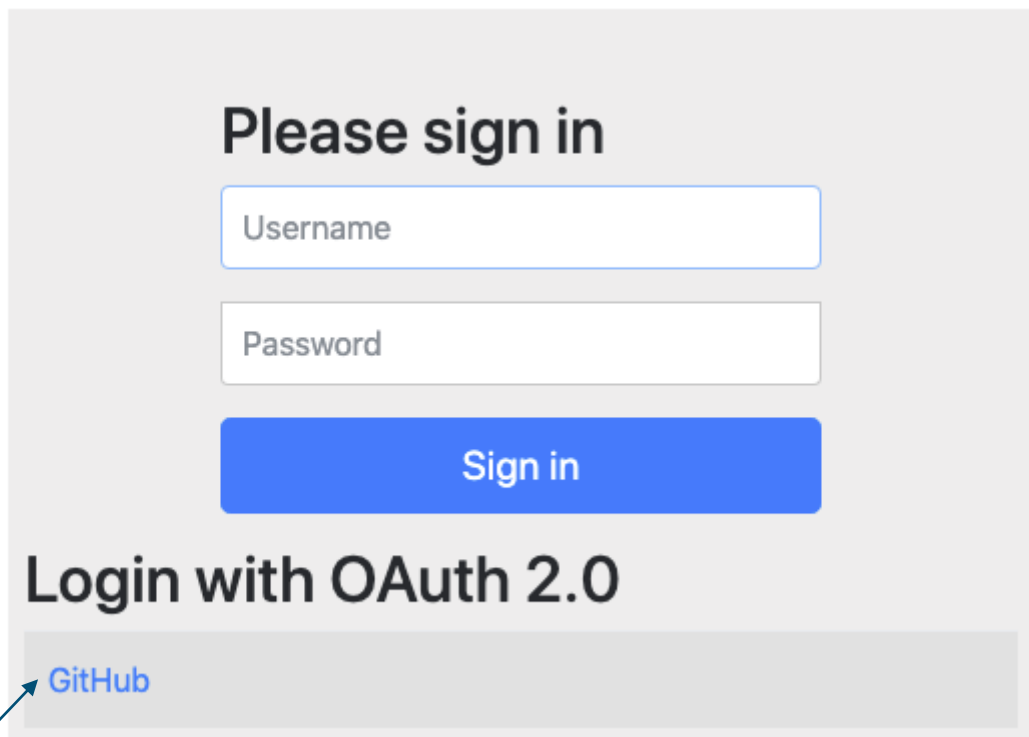
Ajoutons une page d'accueil pour GitHub à votre contrôleur, à l'aide de la méthode

`getGithub()` :

java

```
1 @RestController
2 public class LoginController{
3
4     @RolesAllowed("USER")
5     @RequestMapping("/")
6     public String getUser()
7     {
8         return "Welcome User";
9     }
10
11     @RolesAllowed({"USER","ADMIN"})
12     @RequestMapping("/admin")
13     public String getAdmin()
14     {
15         return "Welcome Admin";
16     }
17
18     @RequestMapping("/")
19     public String getGithub()
20     {
21         return "Welcome Github user!";
22     }
23 }
```

Ensuite, allez sur votre navigateur et tapez localhost:8080 (dépend de notre configuration).



Please sign in

Username

Password

Sign in

Login with OAuth 2.0

GitHub

Cela a bien fonctionné !

11. Référence :

- <https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml#:~:text=Le%20diagramme%20de%20classes%20standard%20est%20compos%C3%A9%20de,liste.%20Chaque%20op%C3%A9ration%20occupe%20sa%20propre%20ligne.%20>
- www.ibm.com/cloud/learn/java-spring-boot
- <https://openclassrooms.com/fr/courses/7137776-securisez-votre-application-web-avec-spring-security/7275496-utilisez-spring-security-dans-votre-application-spring-boot>
- <https://etuutt.readthedocs.io/fr/latest/api/introduction/oauth.html>