CAHIER DES CHARGES

PROJET
Carte grise

DOSSIER REDIGER PAR:

[KOPOIN Bélinda]

Date: 05/12/2024

lien Github:
https://github.com/kokobelly29/Carte-grise-kopoin-Belinda

SOMMAIRE

1. Contexte
    1.1. Les cartes grises: présentation
    1.2. Objectifs
    1.3. Date de rendu du projet
    1.4. Besoins fonctionnels
2. Besoins fonctionnels
3. Ressources nécessaires à la réalisation du projet
    3.1. Ressources logiciels
    3.2. Ressources matérielles
    3.3. Languages nécessaires
    3.4. Sources documentaires
4. Gestion du projet
5. Conception du projet
    5.1. Le front-end
        5.1.1. Wireframes
        5.1.2. Maquettes
        5.1.3. Arborescence (sitemap)
    5.2. Le back-end
        5.2.1. Diagramme de cas d'utilisation
        5.2.2. Diagramme d'activités
        5.2.3. Diagramme de classe
        5.2.4. Dictionnaire
        5.2.5. Modèles Conceptuel de Données (MCD)
        5.2.6. Modèle Logique de Données (MLD)
        5.2.7. Modèle Physique de Données (MPD)
6. Sécurité
    6.1. Protection contre les injections SQL
    6.2. Protection contre les attaques XSS (Cross-Site Scripting)

1. Contexte

1.1. Les cartes grises : présentation

Une carte grise est un document permettant d'identifier un véhicule et également de donner différentes informations concernant son propriétaire, mais aussi les caractéristiques du véhicule. La carte grise est, en définition, une carte d'identité de véhicule. (source: autodemarches.fr)

1.2. Objectifs

Vous occupez actuellement le poste de concepteur et développeur au sein de la Direction des systèmes d'information de la préfecture de votre département. La responsable du service des cartes grises souhaiterait faire évoluer leur application métier. Cependant, aucun document de conception n'est disponible. Votre travail consiste donc à travailler sur l'élaboration de documents de conception de l'application actuelle en vue de faciliter la réflexion autour de son évolution.

1.3. Date de rendu du projet

Le projet sera rendu au plus tard le 05/12/2024.

2. Besoins fonctionnels

L'application métier de gestion des cartes grises devra présenter :

> des marques de voitures
> des modèles de voitures
> les propriétaires
> les véhicules
> les propriétés
> compatibilité avec les appareils (Device compatible)

3. Ressources nécessaires à la réalisation du projet

3.1. Ressources logicielles

Les plateformes et logiciels nécessaires au développement de l'application métier sont:

> l'IDE (environnement de développement) Visual Studio Code

> GitHub (plateforme de développement collaboratif)

> Apache (serveur web contenu dans MAMP)

> MySQL (base de donnée relationnelle / Système de gestion de la base de données relationnelle contenue dans MAMP).

> Trello (outil de gestion de projet)

> Visual paradigme online (outils de conception UML et arborescence)

> Figma (plateforme de maquettage)

> Mocodo (Conception bdd)

j'utiliserais des sources documentaires tels que, https://tinyurl.com/bdf8kasb

3.2. Ressources matérielles

Les matériaux nécessaires à la réalisation de la mission sont un ordinateur fixe et portable (écran, souris, unité centrale). Ces ordinateurs seront reliés à Internet par câble ou Wi-Fi.

3.3. Languages nécessaires

Pour la réalisation de l'application métier, nous emploierons différents langages tels que: Java et sql

Java est un langage de programmation polyvalent et hautement transférable, utilisé pour développer une large gamme d'applications sur différentes plateformes et types d'appareils. Ainsi dans le cadre de l'évolution de l'application métier du service des cartes grises, ce langage serait parfaitement adapté.

Pour stocker, gérer ou récupérer des informations en rapport avec les cartes grises; il nous sera impératif d'avoir une base de données. Elle contiendra :

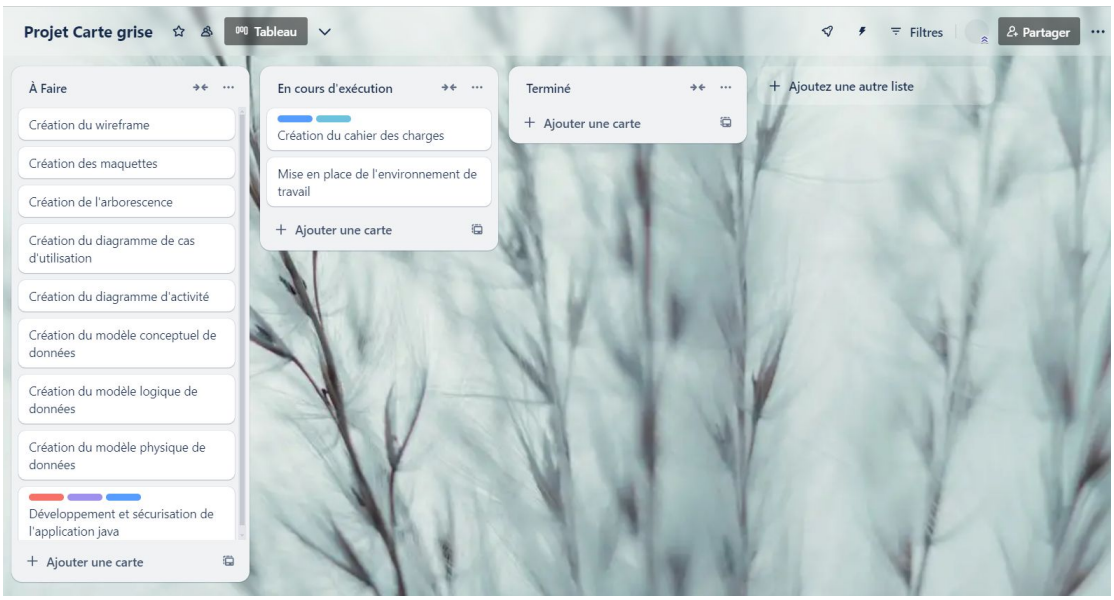
> les données des propriétaires
> les données des véhicules
> les données sur les propriétés
> les données sur les marques
> les données sur les modèles

3.4. Sources documentaires

Pour la réalisation de la mission, nous utiliserons des sources documentaires tels que le site de **Java**.

4. Gestion du projet

Pour réaliser le projet, nous utiliserons la méthode **Agile Kanban**. Afin de pouvoir se répartir le volume de travail plus facilement. Nous utiliserons comme outil de gestion de projet en ligne **Trello**.



en savoir plus (https://tinyurl.com/2s49mtrv)

Nous travaillerons également sur GitHub, une plateforme de développement collaboratif.

5. Conception du projet

Pour mener à bien ce projet, nous sommes passés par ces différentes étapes:

5.1. Le front-end

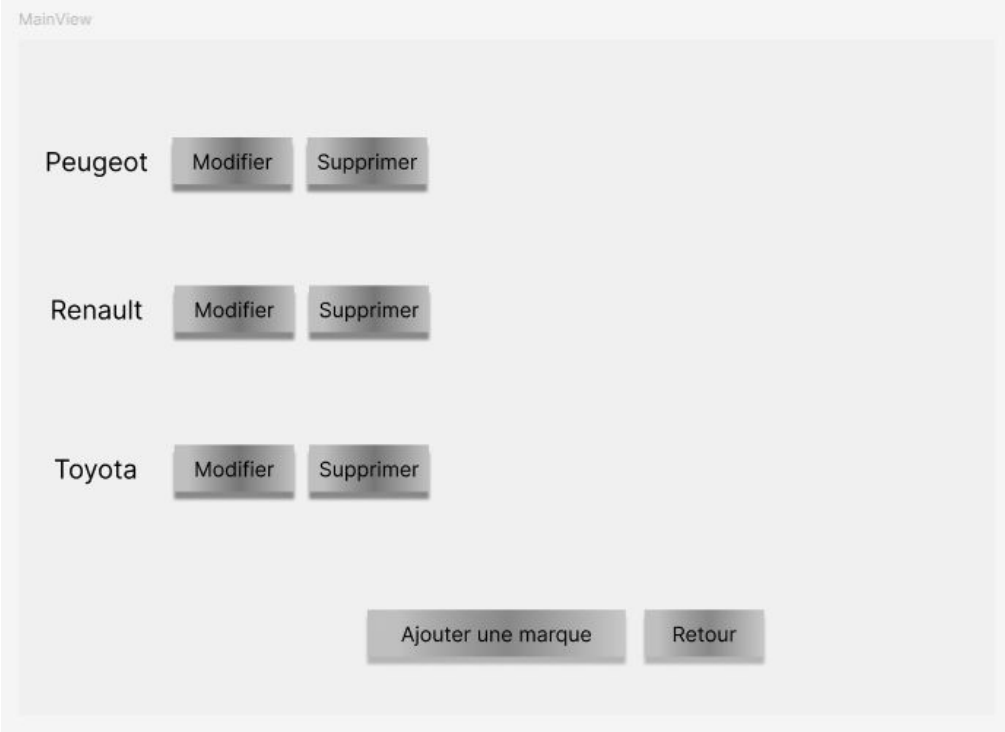
Vous trouverez les aspects visuels de l'application Java avec lesquels les salariés interagissent ci-contre:

5.1.1. Wireframes

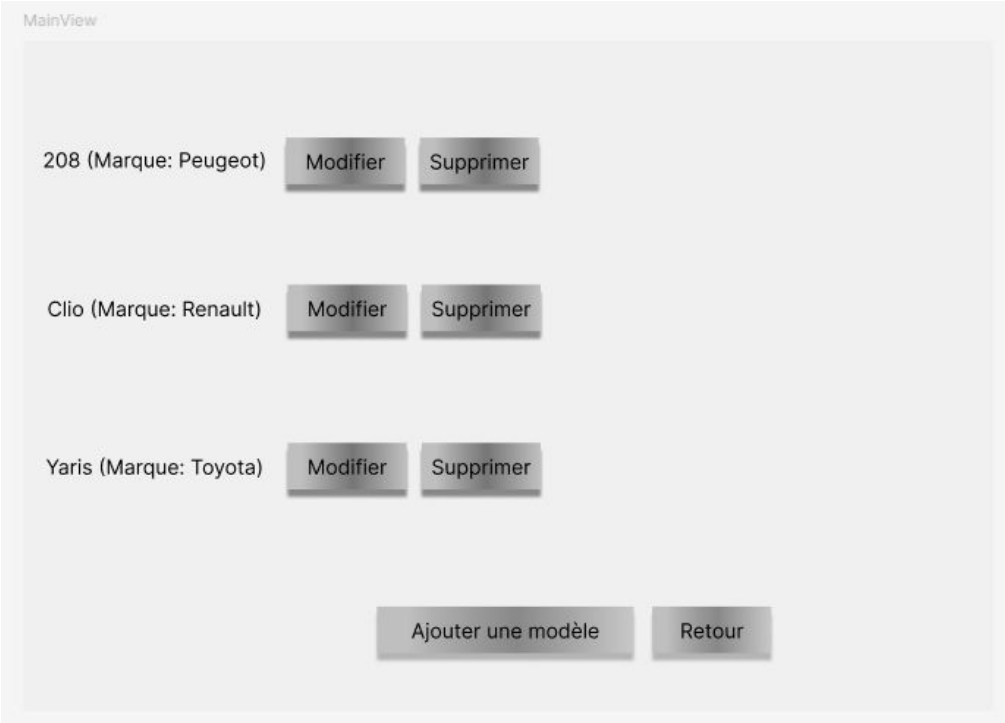
Vous pouvez consulter quelques wireframes ci-dessous:

Wireframe carte grise - page dédié à l'accueil

Wireframe carte grise - page dédié à la gestion des marques

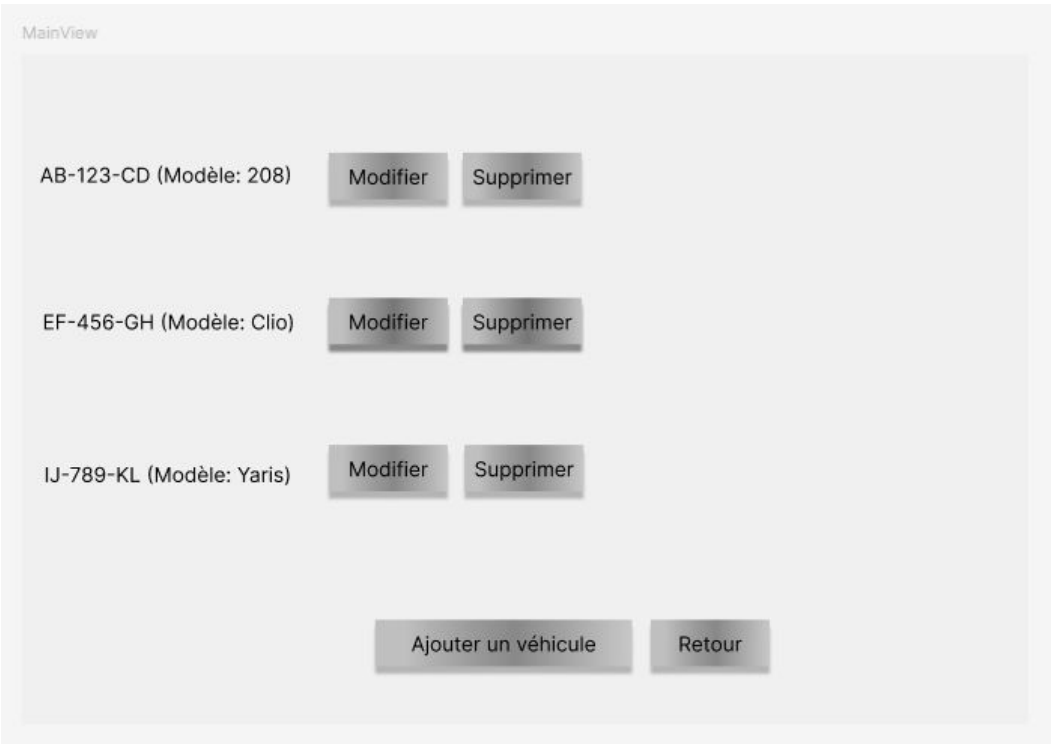


Wireframe carte grise - page dédié à la gestion des modèles de marques



Wireframe carte grise - page dédié à la gestion des propriétaires

Wireframe carte grise - page dédié à la gestion des véhicules

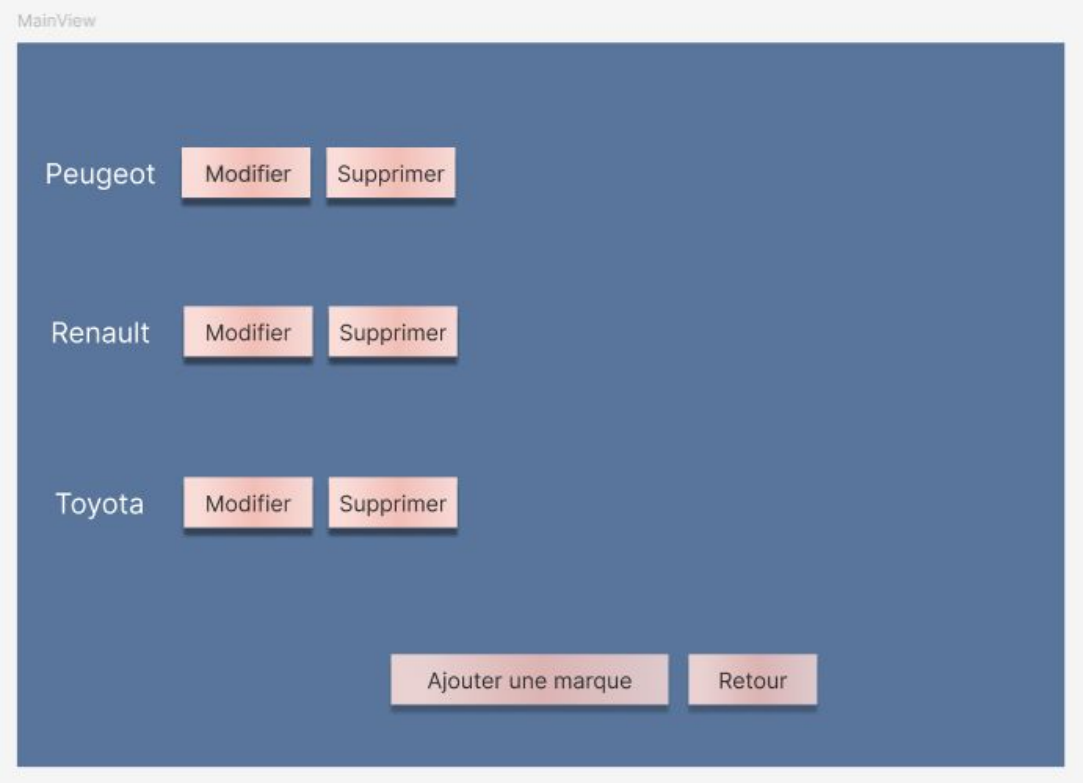


Wireframe carte grise - page dédié à la gestion des propriétés

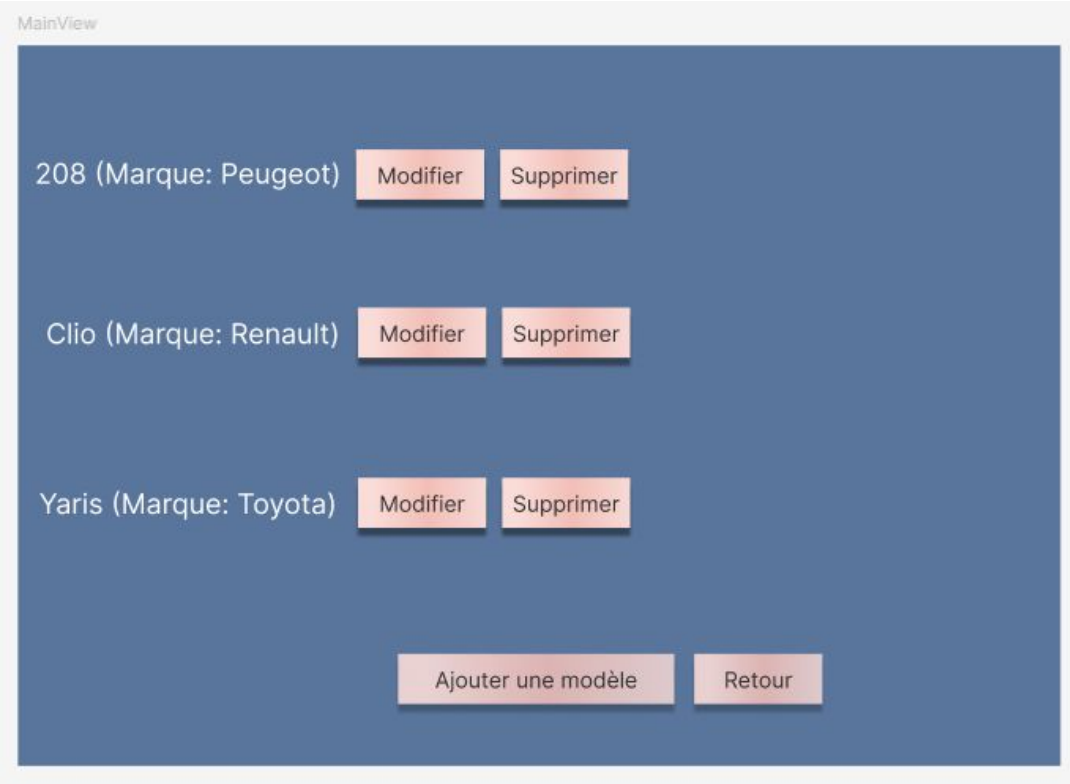
5.1.2. Maquettes

Maquette carte grise - page dédié à l'accueil

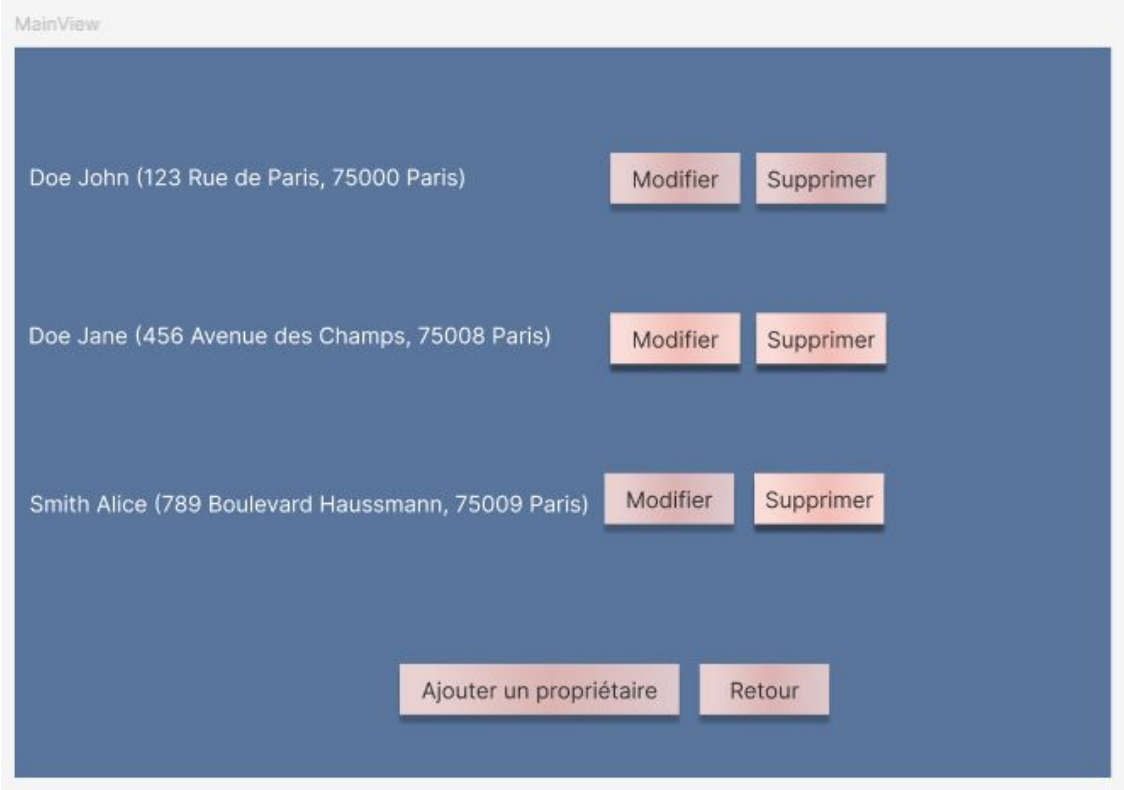
Maquette carte grise - page dédié à la gestion des marques



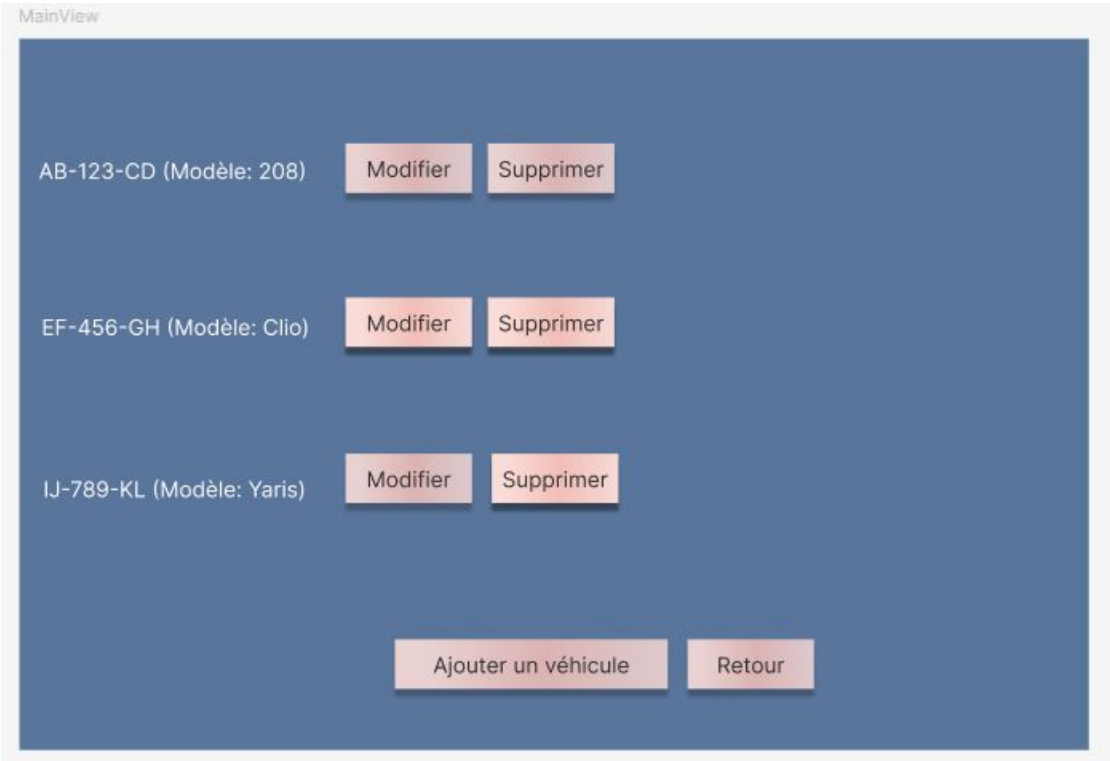
Maquette carte grise - page dédié à la gestion des modèles de marques



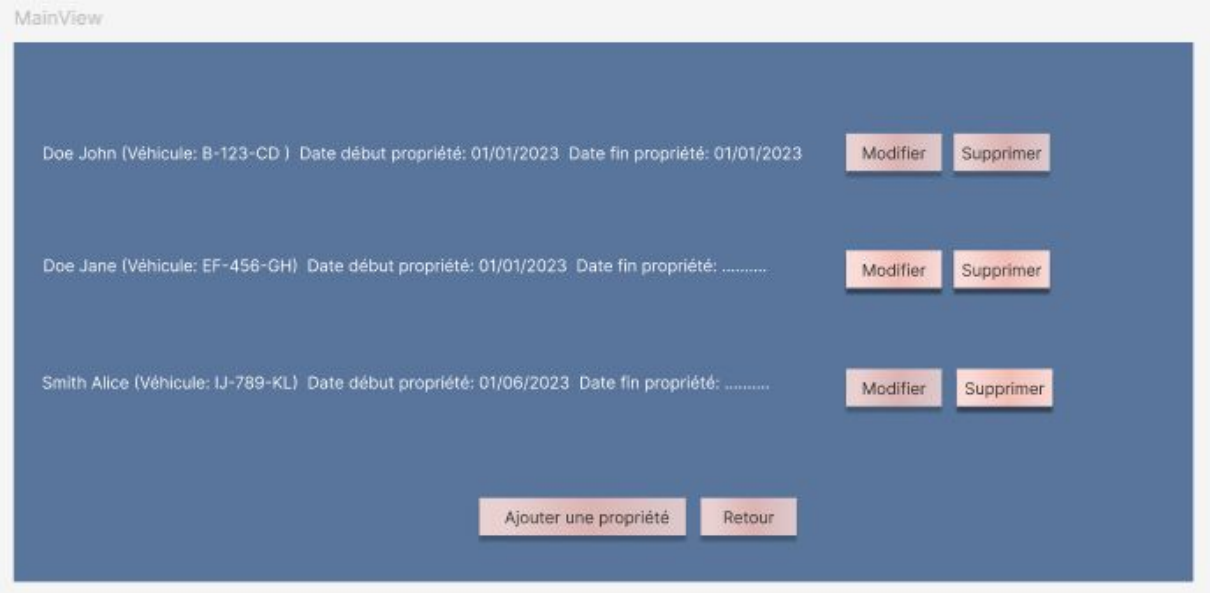
Maquette carte grise - page dédié à la gestion des propriétaires



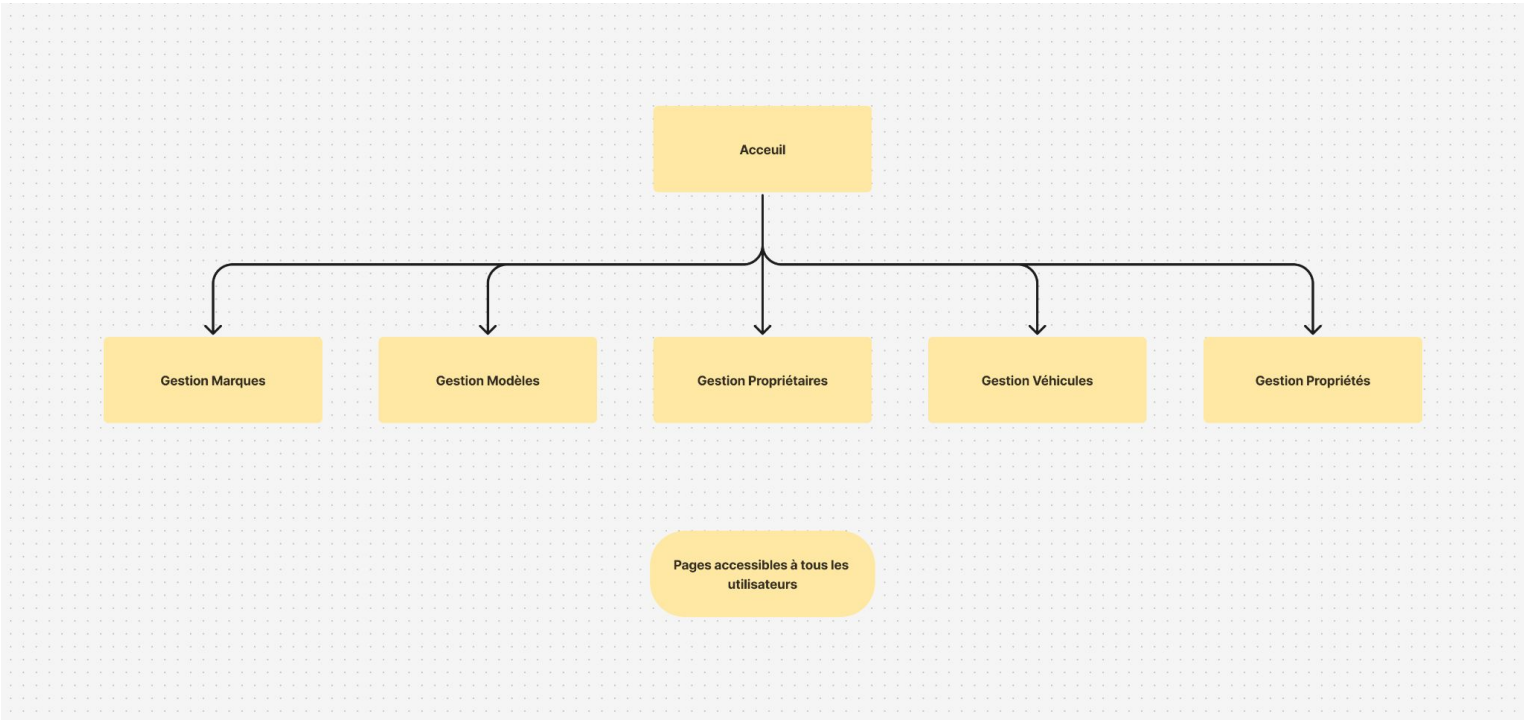
Maquette carte grise - page dédié à la gestion des propriétaires



Maquette carte grise - page dédié à la gestion des propriétaires

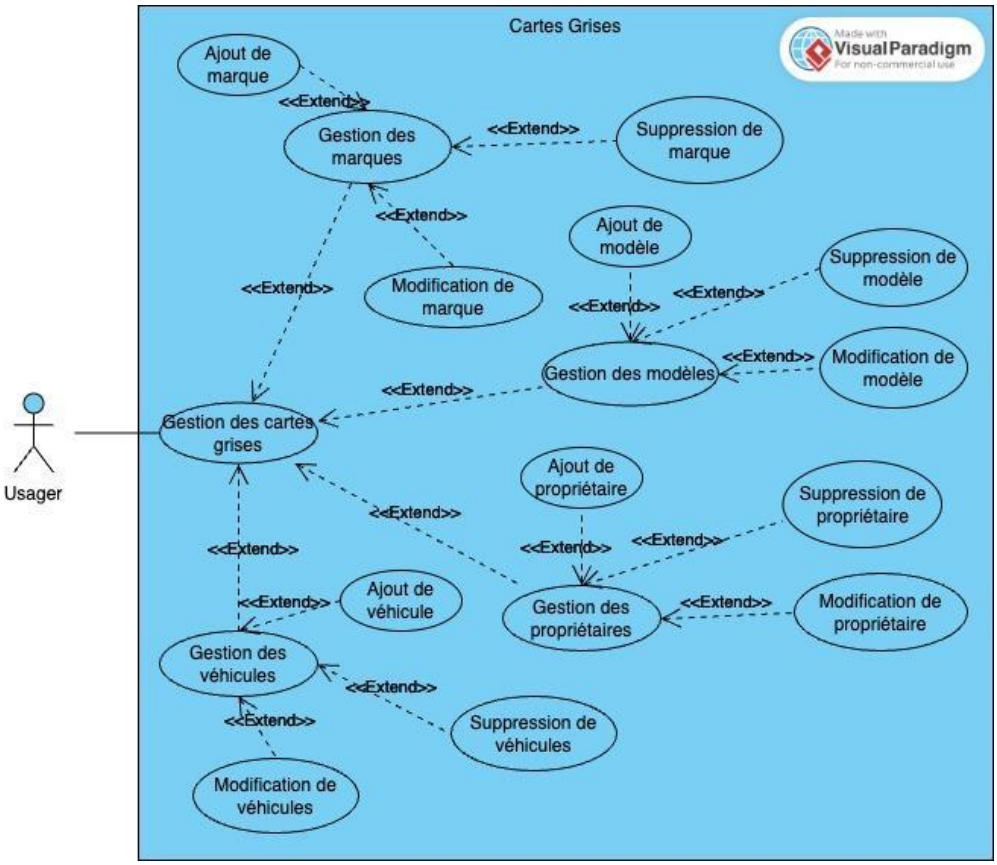


5.1.3. Arborescence (sitemap)



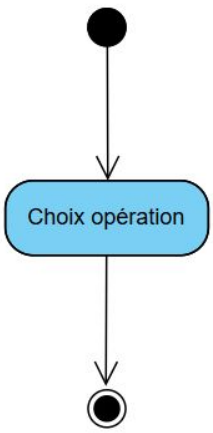
5.2. Le back-end

5.2.1. Diagramme de cas d'utilisation



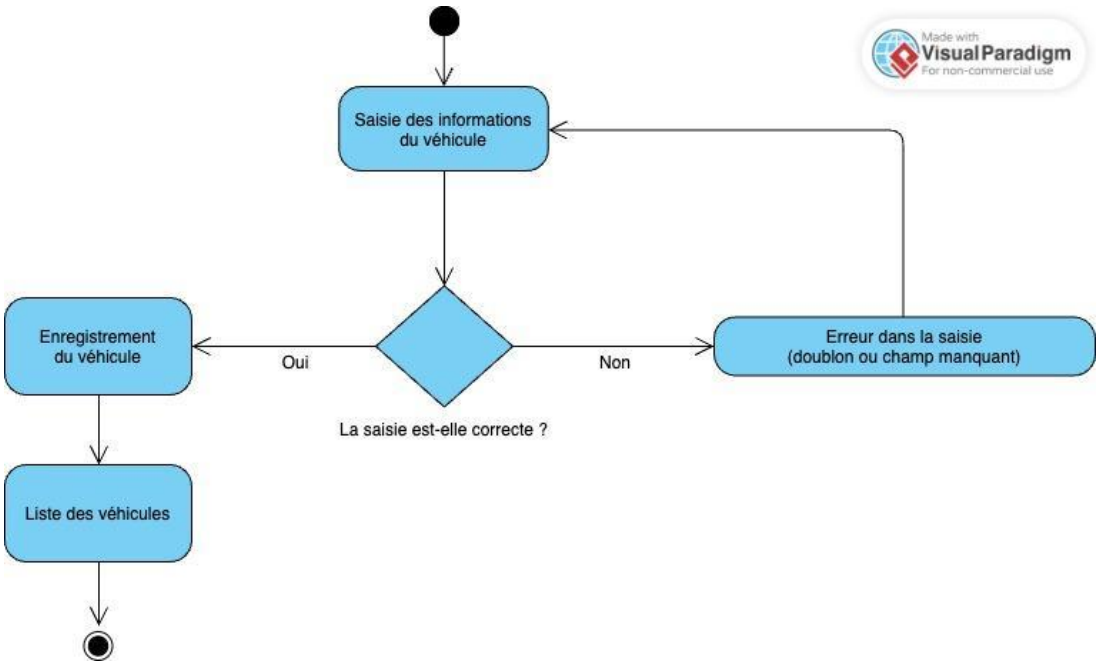
5.2.2. Diagramme d'activité

Diagramme d'activité pour la page d'accueil

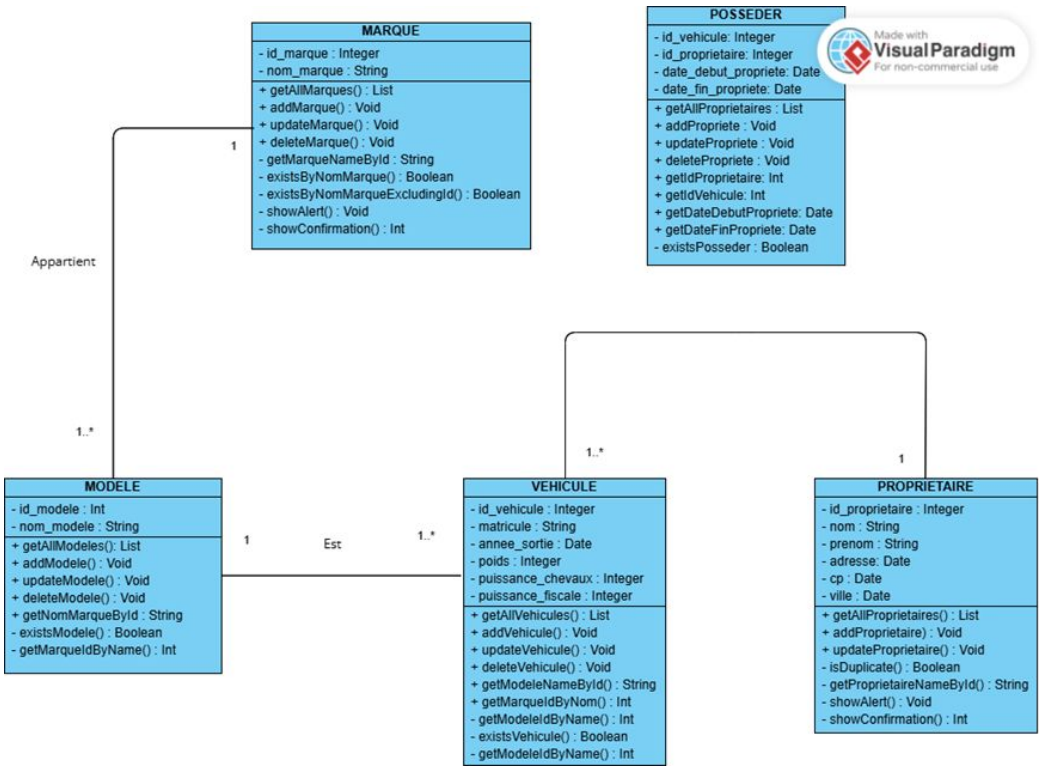


5.2.2. Diagramme d'activité

Diagramme d'activité pour la page ajout de véhicule



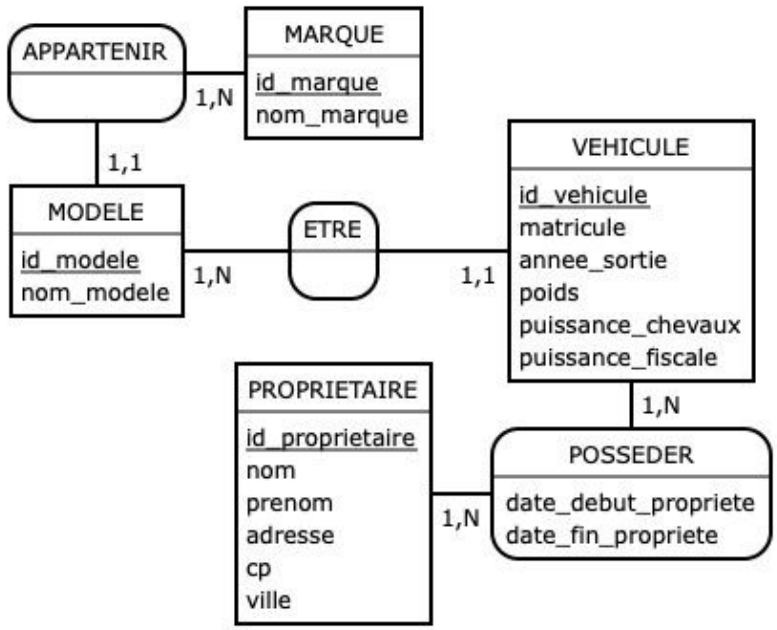
5.2.3. Diagramme de classe



5.2.4. Dictionnaire

Table	Nom de l'attribut	Type	Description
MARQUE	id_marque	INTEGER(4)	Identifiant unique de la marque
MARQUE	nom_marque	VARCHAR(255)	Nom de la marque
MODELE	id_modele	INTEGER(4)	Identifiant unique du modèle
MODELE	nom_modele	VARCHAR(255)	Nom du modèle
MODELE	Id_marque	INTEGER(4)	Identifiant de la marque (clé étrangère)
POSSEDER	id_vehicule	INTEGER(4)	Identifiant du véhicule (clé étrangère)
POSSEDER	id_proprietaire	INTEGER(4)	Identifiant du propriétaire (clé étrangère)
POSSEDER	date_debut_proprietaire	DATE	Date de début de possession du véhicule
POSSEDER	date_fin_propriete	DATE	Date de fin de possession du véhicule
PROPRIETAIRE	Id_proprietaire	INTEGER(4)	Identifiant unique du propriétaire
PROPRIETAIRE	nom	VARCHAR(255)	Nom du propriétaire
PROPRIETAIRE	prenom	VARCHAR(255)	Prénom du propriétaire
PROPRIETAIRE	adresse	VARCHAR(255)	Adresse du propriétaire
PROPRIETAIRE	cp	VARCHAR(10)	Code Postal du propriétaire
PROPRIETAIRE	ville	VARCHAR(255)	Ville du propriétaire
VEHICULE	id_vehicule	INTEGER(4)	Identifiant unique du véhicule
VEHICULE	matricule	VARCHAR(50)	Matricule du véhicule
VEHICULE	annee_sortie	DATE	Date de sortie du véhicule
VEHICULE	poids	VARCHAR(255)	Poids du véhicule
VEHICULE	puissance_chevaux	VARCHAR(11)	Puissance cheval du véhicule
VEHICULE	puissance_fiscale	VARCHAR(11)	Puissance fiscale du véhicule
VEHICULE	id_modele	INTEGER(4)	Identifiant du modèle (clé étrangère)

5.2.5. Modèles Conceptuel de Données (MCD)



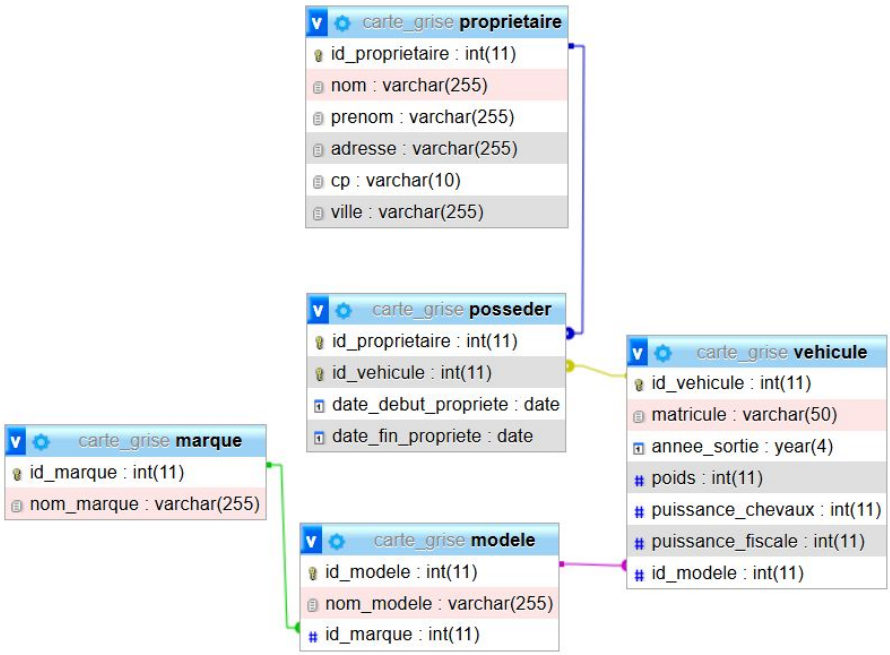
5.2.6. Modèle Logique de Données (MLD)

MARQUE (id_marque, nom_marque)
MODELE (id_modele, nom_modele, *#id_marque*)
POSSEDER (*#id_vehicule*, *#id_proprietaire*, date_debut_propriete, date_fin_propriete)
PROPRIETAIRE (id_proprietaire, nom, prenom, adresse, cp, ville)
VEHICULE (id_vehicule, matricule, annee_sortie, poids, puissance_chevaux, puissance_fiscale, *#id_modele*)

(MLD version BTS)

MARQUE (id_marque, nom_marque)
Clef primaire : id_marque
MODELE (id_modele, nom_modele, #id_marque)
Clef primaire : id_modele
Clefs étrangères : id_marque en référence à MARQUE (id_marque)
POSSEDER (#id_vehicule, #id_proprietaire, date_debut_proprietaire, date_fin_propriete)
Clef primaire : id_vehicule et id_proprietaire
Clefs étrangères: id_vehicule en référence à VEHICULE (id_vehicule)
id_proprietaire en référence à PROPRIETAIRE (id_proprietaire)
PROPRIETAIRE (id_proprietaire, nom, prenom, adresse, cp, ville)
Clef primaire : id_proprietaire
VEHICULE (id_vehicule, matricule, annee_sortie, poids, puissance_chevaux, puissance_fiscale, #id_modele)
Clef primaire : id_vehicule
Clefs étrangères: id_modele en référence à MODELE (id_modele)

5.2.7. Modèle Physique de Données (MPD)



6. Sécurité

6.1. Protection contre les injections SQL

Une attaque par injection SQL consiste à récupérer des données sensibles ou à caractère personnel stockées sur le serveur SQL. Ainsi, il sera nécessaire de rechercher les paramètres vulnérables (données caractères personnelles) pour les protéger. Pour cela nous avons décidé d'utiliser:

> *des instructions préparées*

```
<// Connect to the database.
Connection conn = DriverManager.getConnection(URL, USER, PASS);

// Construct the SQL statement we want to run, specifying the parameter.
String sql = "SELECT * FROM users WHERE email = ?";

// Generate a prepared statement with the placeholder parameter.
PreparedStatement stmt = conn.prepareStatement(sql);

// Bind email value into the statement at parameter index 1.
stmt.setString(1, email);

// Run the query...
ResultSet results = stmt.executeQuery(sql);

while (results.next())
{
   // ...do something with the data returned.
}
```

(source hacksplaining)

7.2. Protection contre les attaques XSS (Cross-Site Scripting)

Cross-site scripting (XSS) est une faille de sécurité qui permet à un attaquant d'injecter dans un site web un code client malveillant. Ce code est exécuté par les victimes et permet aux attaquants de contourner les contrôles d'accès et d'usurper l'identité des utilisateurs (source MDN Web Docs). Les méthodes de prévention XSS les plus courantes et les plus utiles sont :

> *le filtrage*.

Ceci implique de vérifier que les données d'entrée correspondent à ce qui est attendu (texte brut, chiffres, adresses e-mail, etc.).

> l'*échappement*.

Ceci consiste à transformer les caractères spéciaux en équivalents HTML ou JavaScript sûrs (exemple:< devient < > devient > & devient &amp et " devient ")