# Lycée La Tournelle CONTACT A RISQUE - ATTESTATION SUR L ...

# CAHIER DES CHARGES TECHNIQUE

## Projet : Cartes grises

# Sommaire

[1. Contexte du projet](#_heading=h.1d6mma2ci7xl)

[1.1. Présentation du projet](#_heading=h.wek20lqal41w)

[1.2. Date de rendu du projet](#_heading=h.hsbf2s9v0gpe)

[2. Besoins fonctionnels](#_heading=h.72adhlswac0n)

[3. Ressources nécessaires à la réalisation du projet](#_heading=h.lr47i5krzqwp)

[3.1. Ressources matérielles](#_heading=h.moe5ly27tmhr)

[3.2. Ressources logicielles](#_heading=h.cvyk6p742phf)

[4. Gestion du projet](#_heading=h.wj6e27o1xw4s)

[5. Conception du projet](#_heading=h.4jjvlmxvhcof)

[5.1. Le front-end](#_heading=h.b3pf1ig00kap)

[5.1.1. Wireframes](#_heading=h.xbdoysf41ddh)

[5.1.2. Maquettes](#_heading=h.taj1xh3jo3sr)

[5.1.3. Arborescences](#_heading=h.7cpah6l43vkf)

[5.2. Le back-end](#_heading=h.ri4n2nlk4x5y)

[5.2.1. Diagramme de cas d’utilisation](#_heading=h.4hfnlk8rv9if)

[5.2.2. Diagramme d’activités](#_heading=h.d1ay6n1m7wn9)

[5.2.3. Modèles Conceptuel de Données (MCD)](#_heading=h.mhbyfyrbmn9)

[5.2.4. Modèle Logique de Données (MLD)](#_heading=h.rr2kq3n12ss)

[5.2.5. Modèle Physique de Données (MPD)](#_heading=h.5jowy0y259aj)

[6. Technologies utilisées](#_heading=h.6j2nbpnxpppw)

[6.1. Langages de développement Web](#_heading=h.uhwlwqptkq2s)

[6.2. Base de données](#_heading=h.q8i476rr9oys)

[7. Sécurité](#_heading=h.15p9a6rgigad)

[7.1. Login et protection des pages administrateurs](#_heading=h.q04sq9cnrw82)

[7.2. Hachages des mots de passe avec Bcrypt](#_heading=h.ct1wz6q0jyd1)

[7.3. Protection contre les attaques XSS (Cross-Site Scripting)](#_heading=h.cimahn54kest)

[7.4. Protection contre les injections SQL](#_heading=h.cg22ajiezi4v)

# 

# Contexte du projet

## 1.1. Présentation du projet

Votre agence web a été sélectionnée par le comité d’organisation des jeux olympiques de Los Angeles 2028 pour développer une application web permettant aux organisateurs, aux médias et aux spectateurs de consulter des informations sur les sports, les calendriers des épreuves et les résultats des JO 2028.

Votre équipe et vous-même avez pour mission de proposer une solution qui répondra à la demande du client.

## 1.2. Date de rendu du projet

Le projet doit être rendu au plus tard le 09/01/2025.

# Besoins fonctionnels

Le site web devra avoir une partie accessible au public et une partie privée permettant de gérer les données.

Les données seront stockées dans une base de données relationnelle pour faciliter la gestion et la mise à jour des informations. Ces données peuvent être gérées directement via le site web à travers un espace administrateur.

# Ressources nécessaires à la réalisation du projet

## 3.1. Ressources matérielles

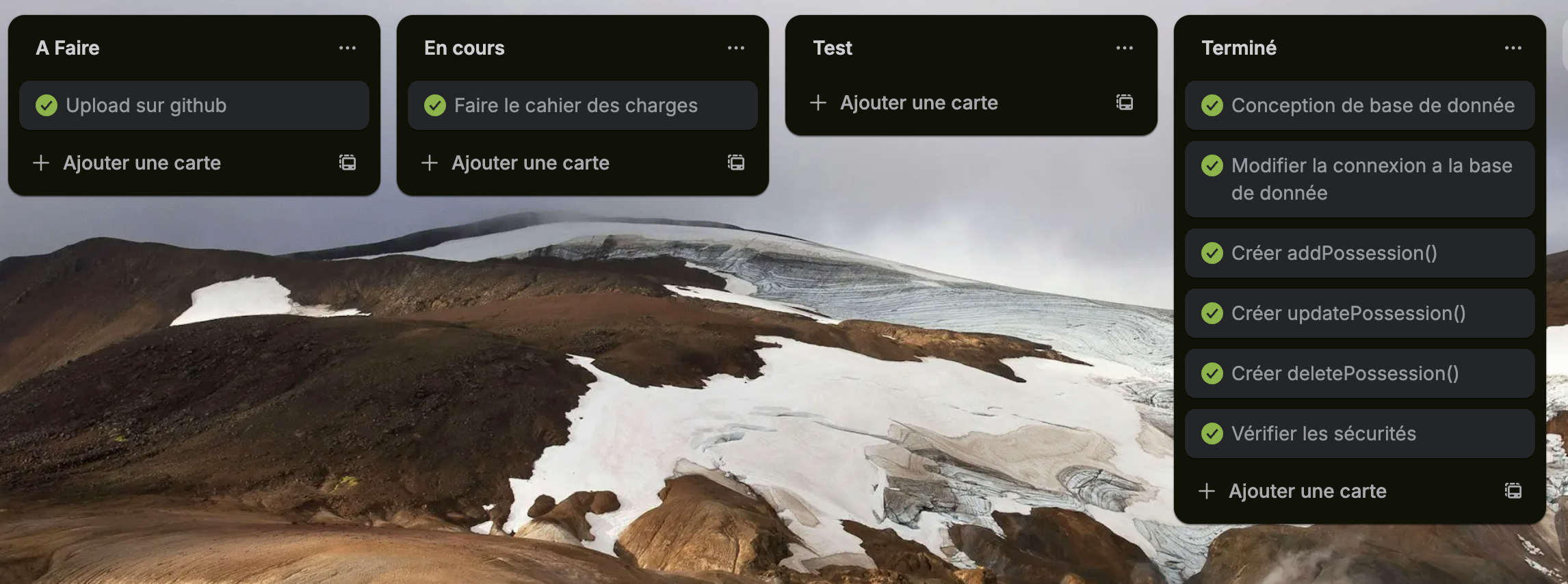
* PC
* Connexion internet
* Serveur local
* Périphérique de travail

## 3.2. Ressources logicielles

* Visual studio code
* Php
* Mysql
* Mamp
* Mocodo
* Java-script
* Html-CSS

# Gestion du projet

Pour réaliser le projet, nous utiliserons la méthode Agile Kanban. Nous utiliserons également l’outil de gestion de projet en ligne Trello.



# Conception du projet

## 5.1. Le front-end

### Une image contenant texte, capture d’écran, nombre, Police Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, ligne, diagramme Le contenu généré par l’IA peut être incorrect.5.1.1. Wireframes

Une image contenant texte, capture d’écran, Police, diagramme

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police, Rectangle

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

### 5.1.2. Maquettes

Une image contenant texte, capture d’écran, affichage, logiciel

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel, Police

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

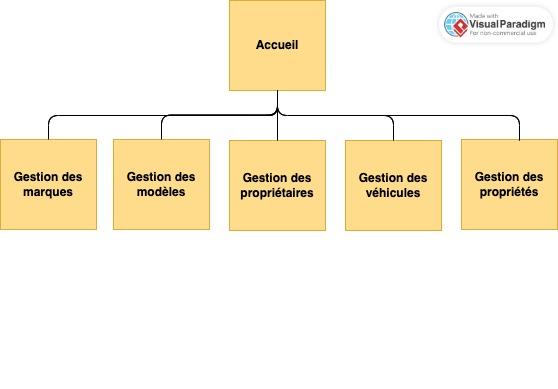
Une image contenant texte, capture d’écran, affichage, logiciel

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

### 5.1.3. Arborescences



## 5.2. Le back-end

### Une image contenant texte, capture d’écran, diagramme, Police Le contenu généré par l’IA peut être incorrect.5.2.1. Diagramme de cas d’utilisation

### 5.2.2. Diagramme d’activités

### 5.2.3. Modèles Conceptuel de Données (MCD)

### 

### 5.2.4. Modèle Logique de Données (MLD)

• MARQUE (id\_marque, nom\_marque)

Clé primaire :id\_marque

• MODELE (id\_modele, nom\_modele, id\_marque)

Clé primaire :id\_modele

Clé étrangère :id\_marque en référence à id\_marque de MARQUE

• VEHICULE (id\_vehicule, matricule, annee\_sortie, poids, puissance\_chevaux, puissance\_fiscale, id\_modele)

Clé primaire : id\_vehicule

Clé étrangère :id\_modele en référence à id\_modele de MODELE

• PROPRIETAIRE (id\_proprietaire, nom, prenom, adresse, cp, ville)

Clé primaire :id\_proprietaire

• POSSEDER (id\_proprietaire, id\_vehicule, date\_debut\_propriete, date\_fin\_propriete)

Clé primaire :id\_proprietaire, id\_vehicule

Clés étrangères :id\_proprietaire en référence à id\_proprietaire de PROPRIETAIRE id\_vehicule en référence à id\_vehicule de VEHICULE

### 5.2.5. Modèle Physique de Données (MPD)

### 

# Technologies utilisées

## 6.1. Langages de développement



* Java

## Une image contenant texte, Graphique, graphisme, Police Le contenu généré par l’IA peut être incorrect.6.2. Base de données

Une image contenant Graphique, Police, conception

Le contenu généré par l’IA peut être incorrect.

* MySQL
* MAMP

# Sécurité

## 7.1. Login et protection des pages administrateurs

### Définition de la solution :

1. **LOGIN ADMIN**

* **Authentification** : Vérifier identité admin avant accès.  
  **Composants** : Formulaire → Vérif BCrypt → Création session.

1. **PAGES ADMIN**

* **Interfaces sécurisées** : pages réservées aux gestionnaires.  
  **Exemples** : Dashboard, gestion utilisateurs, logs, configuration.

1. **PROTECTION**

* **Sécurisation accès** aux fonctionnalités sensibles.

Extraits de code

|  |
| --- |
| **CONFIRMATION AVANT ACTION :**  int confirm = JOptionPane.showConfirmDialog(ProprietaireView.this,  "Supprimer ce propriétaire ?", "Confirmer", JOptionPane.YES\_NO\_OPTION);  **GESTION DES ERREURS :**  try {  new ProprietaireView(proprietaireController).showWindow();  }  catch (Exception ex) {  showError("Propriétaire", ex);  }  **PROTECTIONS APPLIQUÉES :**   * Confirmation utilisateur avant suppression * Gestion des exceptions pour accès aux vues * Encapsulation des actions sensibles |

## 7.2. Hachages des mots de passe avec Bcrypt

### Définition de la solution :

**Bcrypt est un algorithme de hachage conçu exclusivement pour sécuriser les mots de passe.**

Il transforme un mot de passe en une chaîne illisible et impossible à inverser. Sa particularité : il est volontairement lent pour résister aux attaques par force brute.

**POURQUOI BCRYPT ?**

* **Sécurisé** : Salt intégré, résiste aux attaques par tables arc-en-ciel
* **Adaptatif** : On peut augmenter la difficulté avec le temps
* **Standard** : Recommandé par les experts en cybersécurité

Extraits de code

|  |
| --- |
| **PROTECTION CONTRE LES DOUBLONS :**  public static boolean exists(String matricule, Integer excludeId) {  String sql = "SELECT COUNT(\*) FROM VEHICULE WHERE matricule = ?";  if (excludeId != null) sql += " AND id\_vehicule != ?";    try (Connection conn = DBConnection.getConnection();  PreparedStatement ps = conn.prepareStatement(sql)) {    ps.setString(1, matricule);  if (excludeId != null) ps.setInt(2, excludeId);    try (ResultSet rs = ps.executeQuery()) {  if (rs.next()) return rs.getInt(1) > 0;  }  } catch (SQLException e) {  System.err.println("Erreur exists : " + e.getMessage());  }  return false;  }  **VÉRIFICATION DES RELATIONS AVANT SUPPRESSION :**  try (PreparedStatement psCheck = conn.prepareStatement(checkSql)) {  psCheck.setInt(1, id);  try (ResultSet rs = psCheck.executeQuery()) {  if (rs.next() && rs.getInt(1) > 0) {  conn.rollback(); // impossible de supprimer si véhicule lié à une possession  return false;  }  }  }  **PRINCIPES DE SÉCURITÉ APPLIQUÉS :**   * Validation d'unicité : Empêche les doublons de matricule * Transactions : Rollback en cas d'erreur ou de contrainte * Requêtes préparées : Protection contre l'injection SQL * Vérification des dépendances : Empêche la suppression de données liées   Objectif : Maintenir l'intégrité des données et prévenir les erreurs applicatives. |

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

### Définition de la solution :

**XSS (Cross-Site Scripting)** : Attaque où un pirate injecte du code JavaScript malveillant dans une page web, exécuté par les navigateurs des autres visiteurs.

**Impact** : Vol de sessions, redirection, vol de données, défiguration de site.

**2 TYPES PRINCIPAUX**

**1. XSS Stocké (Persistant) :** Code malveillant stocké en base (commentaires, articles), affecte tous les utilisateurs.

**2. XSS Réfléchi (Non-persistant) :** Code injecté via URL ou formulaire, affecte immédiatement l'utilisateur.

Extraits de code

|  |
| --- |
| **VALIDATION DES ENTREES DANS Vehicule.java :**  /\*\* Vérifie si un véhicule existe (par matricule) \*/  public static boolean exists(String matricule, Integer excludeId) {  String sql = "SELECT COUNT(\*) FROM VEHICULE WHERE matricule = ?";  if (excludeId != null) sql += " AND id\_vehicule != ?";  try (Connection conn = DBConnection.getConnection();  PreparedStatement ps = conn.prepareStatement(sql)) {    ps.setString(1, matricule); // ← PREPARED STATEMENT protège injection SQL  if (excludeId != null) ps.setInt(2, excludeId);    try (ResultSet rs = ps.executeQuery()) {  if (rs.next()) return rs.getInt(1) > 0;  }  } catch (SQLException e) {  System.err.println("Erreur exists : " + e.getMessage());  }  return false;  }  **AFFICHAGE SÉCURISÉ DANS ProprietaireView.java :**  // Ligne ~90 : Affichage des données dans le tableau  tableModel.addRow(new Object[]{  p.getIdProprietaire(),  p.getNom(), // Données brutes mais affichées dans JTable sécurisée  p.getPrenom(), // JTable échappe automatiquement le contenu  p.getAdresse(),  p.getCp(),  p.getVille(),  "Modifier", // Textes statiques sécurisés  "Supprimer"  });  **CONFIRMATION D'ACTION (Protection indirecte) :**  // Dans ButtonEditor class - ligne ~175  int confirm = JOptionPane.showConfirmDialog(ProprietaireView.this,  "Supprimer ce propriétaire ?", // ← Message fixe, sécurisé  "Confirmer",  JOptionPane.YES\_NO\_OPTION);  **MESURES ANTI-XSS DANS le CODE :**   * **Requêtes préparées** : PreparedStatement empêche l'injection SQL * **JTable sécurisée** : Échappement automatique des données * **Messages fixes** : Textes constants dans les dialogues * **Validation métier** : Vérification des doublons, formats |

## 7.4. Protection contre les injections SQL

### Définition de la solution :

**Injection SQL :** Attaque de sécurité qui permet à un utilisateur malveillant d'interférer avec les requêtes SQL qu'une application exécute sur sa base de données.

Comment ça marche ?  
L'attaquant insère du code SQL malveillant dans des champs de saisie (formulaires, URL, etc.) qui est ensuite exécuté par la base de données lorsqu'il est concaténé sans protection dans une requête SQL.

**Conséquences possibles :**

* Accès non autorisé aux données
* Modification/suppression de données
* Élévation de privilèges
* Exécution de commandes système

Extraits de code

|  |
| --- |
| **REQUÊTES PRÉPARÉES DANS Vehicule.java :**  // Ligne ~72 : Ajout d'un véhicule avec PreparedStatement  public static boolean addVehicule(String matricule, int annee, double poids,  int chevaux, int fiscale, int idModele) {    String sql = "INSERT INTO VEHICULE (matricule, annee\_sortie, poids, " +  "puissance\_chevaux, puissance\_fiscale, id\_modele) " +  "VALUES (?, ?, ?, ?, ?, ?)"; // ← Placeholders  try (Connection conn = DBConnection.getConnection();  PreparedStatement ps = conn.prepareStatement(sql)) {    // Assignation sécurisée des paramètres  ps.setString(1, matricule);  ps.setInt(2, annee);  ps.setDouble(3, poids);  ps.setInt(4, chevaux);  ps.setInt(5, fiscale);  ps.setInt(6, idModele);  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.err.println("Erreur addVehicule : " + e.getMessage());  return false;  }  }  **REQUÊTE AVEC PARAMÈTRE CONDITIONNEL :**  // Ligne ~54 : Méthode exists() avec paramètre optionnel  public static boolean exists(String matricule, Integer excludeId) {  String sql = "SELECT COUNT(\*) FROM VEHICULE WHERE matricule = ?";  if (excludeId != null) sql += " AND id\_vehicule != ?"; // ← Construction sécurisée  try (Connection conn = DBConnection.getConnection();  PreparedStatement ps = conn.prepareStatement(sql)) {    ps.setString(1, matricule);  if (excludeId != null) ps.setInt(2, excludeId); // ← Paramètre conditionnel    try (ResultSet rs = ps.executeQuery()) {  if (rs.next()) return rs.getInt(1) > 0;  }  } catch (SQLException e) {  System.err.println("Erreur exists : " + e.getMessage());  }  return false;  }  **TECHNIQUES UTILISÉES DANS VOTRE CODE :**   * **PreparedStatement** : Séparation code/données avec ? * **Typage fort** : setString(), setInt(), setDouble() * **Validation métier** : Vérification existence avant action * **Gestion transactions** : commit()/rollback() pour intégrité * **Logs d'erreur** : Journalisation sans exposition de détails sensibles |