AVANCEMENT DU PROJET S5 – WEB+MOBILE+GODOT

PARTIE MOBILE :

CONCEPTION DE LA BASE DE DONNÉES - PROJET MOBILE

Système de Gestion de Base de Données : PostgreSQL

Ce projet utilise une base de données relationnelle pour gérer les utilisateurs, leurs véhicules et les incidents (problèmes) signalés.

STRUCTURE DES TABLES :

1. Table 'users' (Utilisateurs)
   - id : Clé primaire (SERIAL)
   - email : Adresse email unique de l'utilisateur (VARCHAR, Unique, Non Null)
   - password : Mot de passe de l'utilisateur (VARCHAR, Non Null)
   - name : Nom complet de l'utilisateur (VARCHAR)

2. Table 'cars' (Véhicules)
   - id : Clé primaire (SERIAL)
   - make : Marque du véhicule (VARCHAR, Non Null)
   - model : Modèle du véhicule (VARCHAR, Non Null)
   - year : Année de fabrication (INTEGER)
   - owner_id : Clé étrangère vers users.id (INTEGER, Références users)

3. Table 'issues' (Problèmes/Incidents)
   - id : Clé primaire (SERIAL)
   - description : Description du problème (TEXT, Non Null)
   - severity : Niveau de gravité (VARCHAR)

- car_id : Clé étrangère vers cars.id (INTEGER, Références cars)

- status : État du problème (VARCHAR, Défaut: 'open')

- created_at : Date et heure de création (TIMESTAMP, Défaut: CURRENT_TIMESTAMP)

RELATIONS :

- Relation Utilisateur -> Voitures (1:N) : Un utilisateur peut posséder plusieurs véhicules. Chaque véhicule est lié à un utilisateur unique via le champ 'owner_id'.

- Relation Voiture -> Problèmes (1:N) : Un véhicule peut avoir plusieurs problèmes signalés. Chaque problème est lié à un véhicule spécifique via le champ 'car_id'.

DOCUMENTATION TECHNIQUE - SYSTÈME DE GESTION MOBILE

1. ARCHITECTURE TECHNIQUE

- Frontend : Ionic React

  - Framework : React.js avec les composants UI d'Ionic pour une expérience mobile native.

  - Build Tool : Vite.js pour une compilation rapide.

  - Native : Capacitor pour le déploiement sur Android/iOS.

- Backend : Node.js / Express

  - Serveur REST API pour le traitement des requêtes.

  - Gestion des CORS pour la communication sécurisée avec le frontend mobile.

- Base de Données : PostgreSQL

  - Base relationnelle robuste pour la gestion des données structurées.

(À modifier, passer de Vite.js à Vue.js)

2. LOGIQUE MÉTIER ET FONCTIONNALITÉS

A. Authentification (Login)

  - Objectif : Sécuriser l'accès aux fonctionnalités de l'application.

- Logique :

1. L'utilisateur saisit son email et son mot de passe.

2. Le frontend envoie une requête POST à l'API backend (/login).

3. Le backend interroge la table 'users' pour vérifier les identifiants.

4. En cas de succès, l'utilisateur est redirigé vers l'interface principale et ses informations de session sont stockées.

B. Gestion des Véhicules (Insertion)

- Objectif : Permettre au client d'enregistrer ses véhicules.

- Logique :

1. Formulaire de saisie (Marque, Modèle, Année).

2. Envoi des données via une requête POST à /cars.

3. Le système lie automatiquement le véhicule à l'utilisateur connecté via son ID (owner_id).

4. Les données sont persistées dans la table 'cars'.

C. Signalement d'Interventions (Report d'incident)

- Objectif : Déclarer un problème ou une demande d'intervention sur un véhicule.

- Logique :

1. Récupération préalable de la liste des véhicules enregistrés via une requête GET (/cars).

2. Sélection du véhicule concerné par l'utilisateur.

3. Saisie de la description du problème et choix de la sévérité (ex: Faible, Haute).

4. Envoi d'une requête POST à /issues.

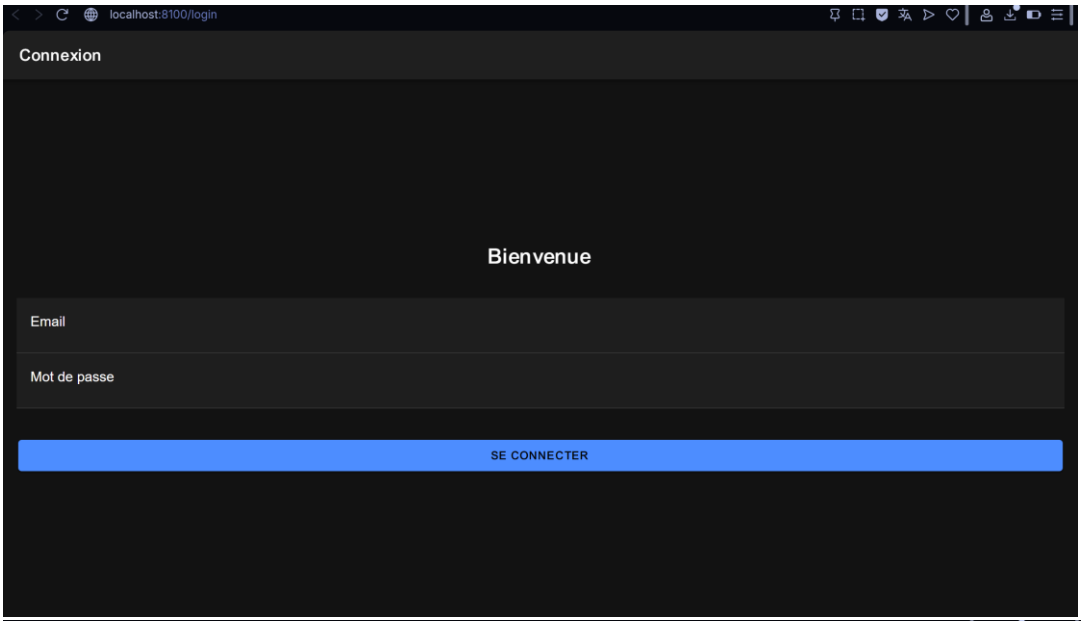
5. Création d'une entrée dans la table 'issues' liée au car_id sélectionné.

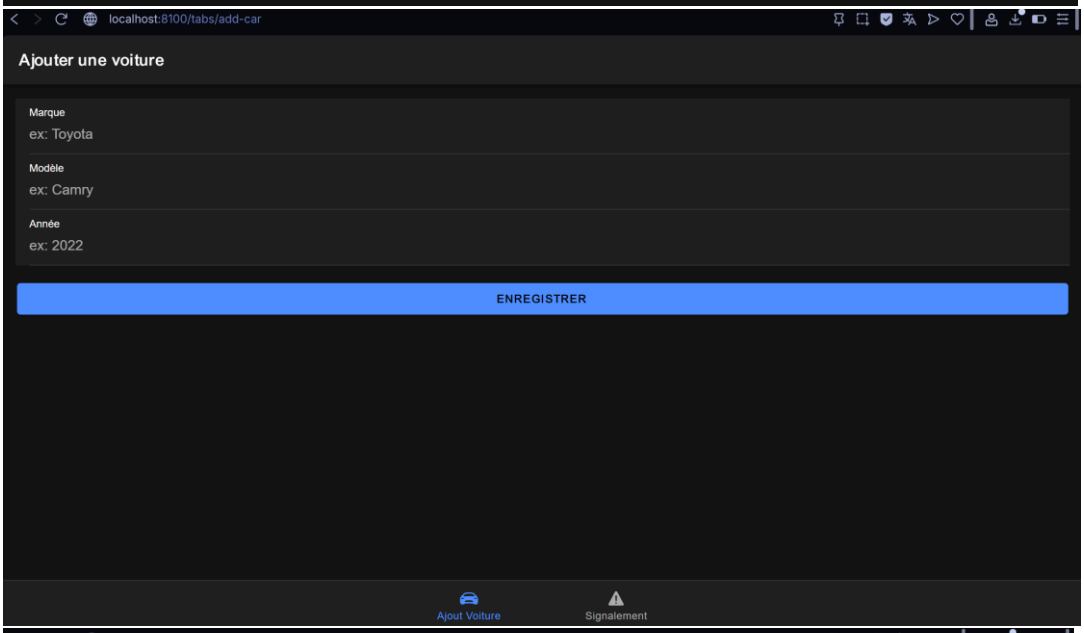
6. Statut initialisé par défaut à 'open' (ouvert) pour permettre le suivi par les techniciens.

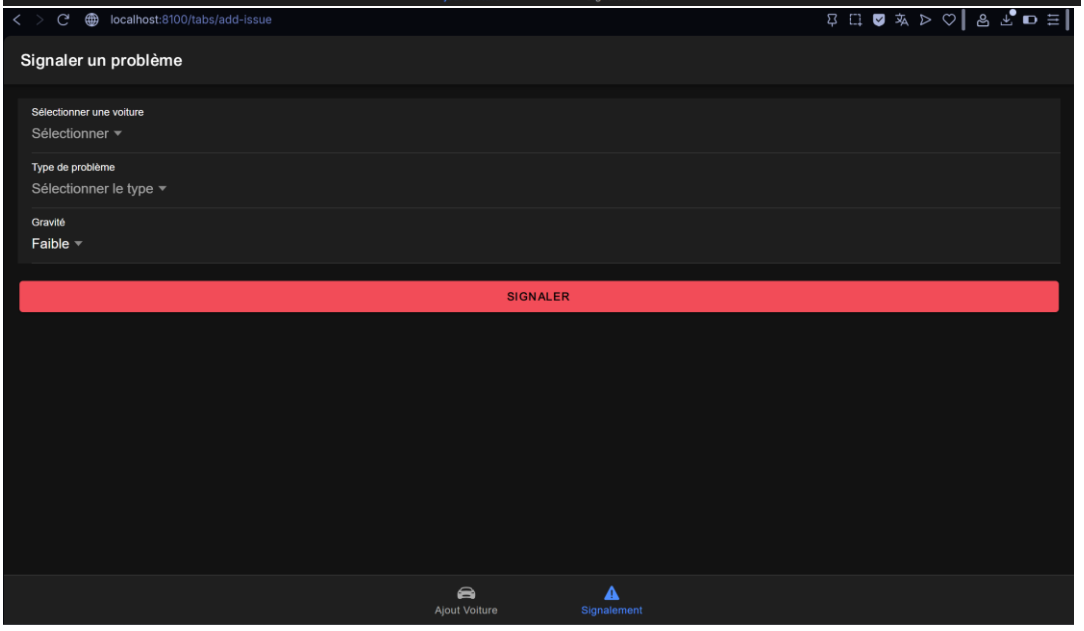
3. SÉCURITÉ ET PERSISTANCE

- Les données sont stockées de manière centralisée dans PostgreSQL.

- La séparation claire entre le frontend (React) et le backend (Express) permet une maintenance isolée et une scalabilité du système.


localhost:8100/login
Connexion
Bienvenue
Email
Mot de passe
SE CONNECTER


localhost:8100/tabs/add-car
Ajouter une voiture
Marque
ex: Toyota
Modèle
ex: Camry
Année
ex: 2022
ENREGISTRER
Ajout Voiture
Signalement


localhost:8100/tabs/add-issue
Signaler un problème
Sélectionner une voiture
Sélectionner
Type de problème
Sélectionner le type
Gravité
Faible
SIGNALER
Ajout Voiture
Signalement

PARTIE WEB :

Le système utilise une base de données relationnelle (PostgreSQL) avec l'architecture suivante :

A. Modèle Conceptuel (Tables principales) :

- users : Gère l'authentification et les rôles (admin/user).

- clients : Informations personnelles des clients (nom, prénom, email, téléphone, adresse).

- voitures : Véhicules enregistrés, liés à un client (marque, modèle, immatriculation, année, couleur).

- interventions : Catalogue des services disponibles (vidange, freins, etc.) avec prix et durée estimée.

- reparations : Dossiers de réparation liés à une voiture, incluant le statut (en_attente, en_cours, terminee) et les dates.

B. Relations :

- Un Client possède plusieurs Voitures (1:N).

- Une Voiture peut avoir plusieurs Réparations (1:N).

- Une Réparation peut inclure plusieurs Interventions (N:N) via une table de jointure 'reparation_intervention'.

C. Règles de Gestion implémentées en base :

- Unicité de l'email pour les utilisateurs et clients.

- Unicité de l'immatriculation pour les voitures.

- Calcul automatique du montant total des réparations basé sur la somme des interventions liées.

2. DOCUMENTATION TECHNIQUE

Architecture : Application Web découplée (Frontend / Backend API).

A. Backend (API) :

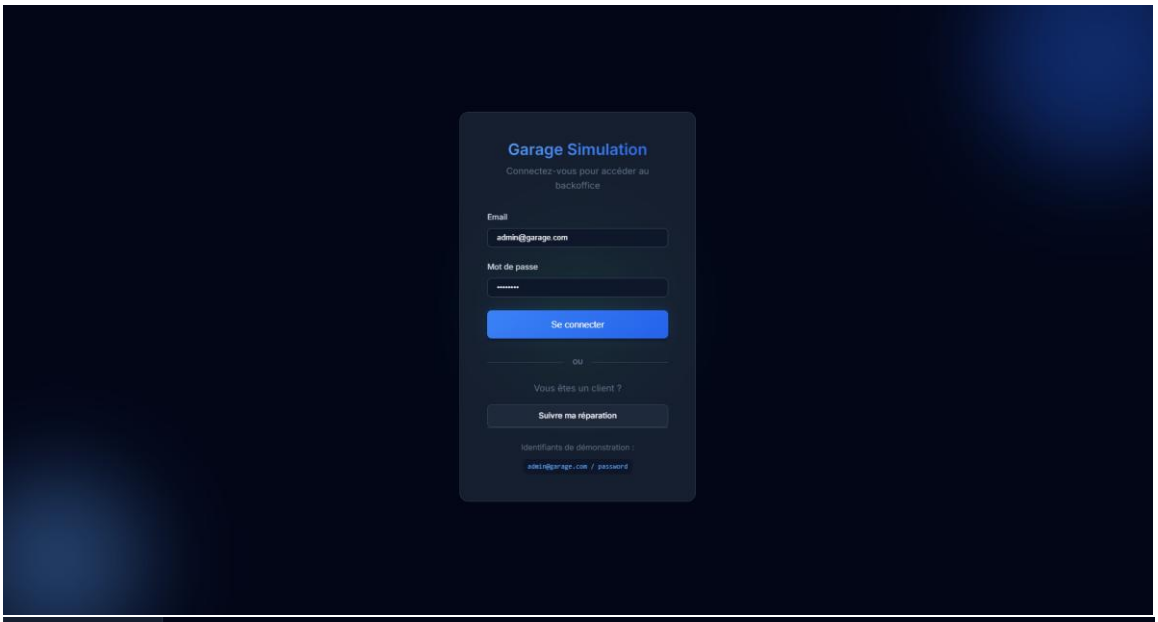
- Framework : Laravel 10+
- Authentification : Laravel Sanctum (Token-based)
- Base de données : PostgreSQL
- Fonctionnalités clés :
  - Contrôle de la capacité du garage (Max 2 réparations en cours simultanément).
  - API RESTful pour tous les CRUD.
  - Calculs statistiques centralisés.

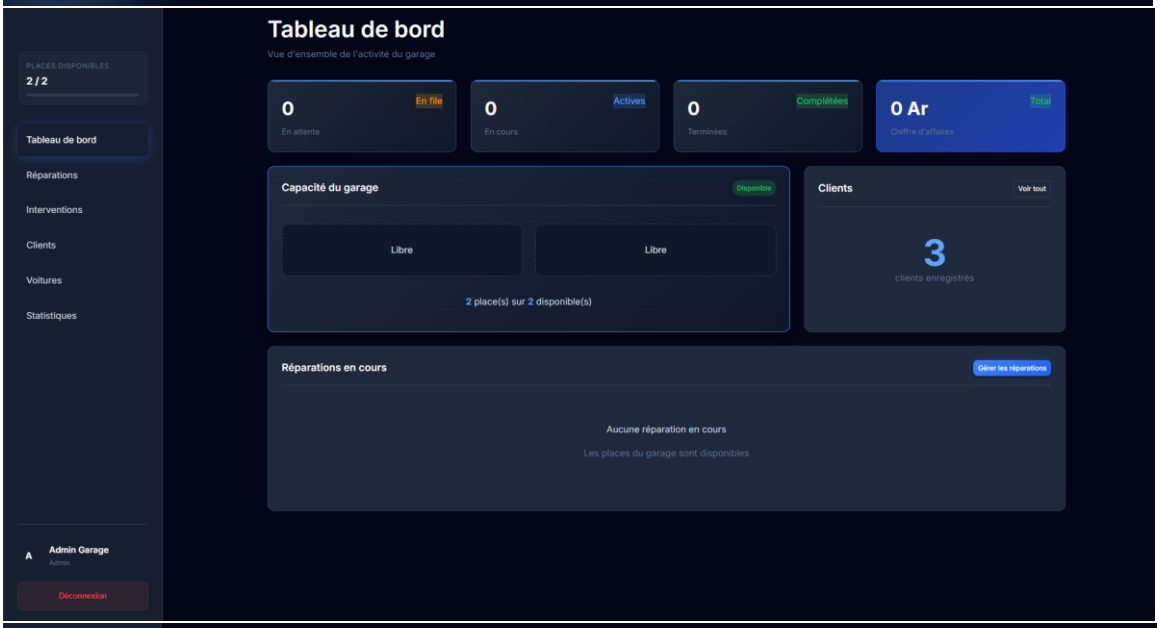
B. Frontend :

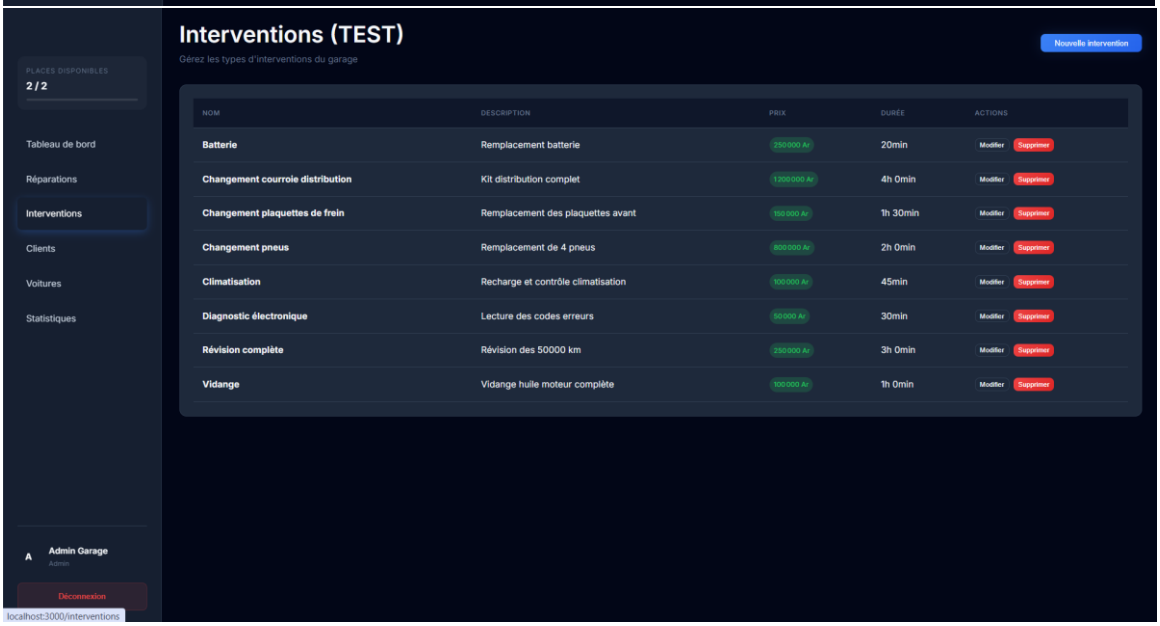
- Framework : Vue.js 3 (Composition API)
- Design : CSS

C. Déploiement :

- Environnement conteneurisé avec Docker et Docker Compose.


Garage Simulation
Connectez-vous pour accéder au backoffice
Email
admin@garage.com
Mot de passe
••••••••
Se connecter
ou
Vous êtes un client ?
Suivre ma réparation
Identifiants de démonstration :
admin@garage.com / password


PLACES DISPONIBLES
2 / 2
Tableau de bord
Réparations
Interventions
Clients
Voitures
Statistiques
Admin Garage
Admin
Déconnexion
Tableau de bord
Vue d'ensemble de l'activité du garage
0
En file
En attente
0
Actives
En cours
0
Complétées
Terminées
0 Ar
Total
Chiffre d'affaires
Capacité du garage
Disponible
Libre
Libre
2 place(s) sur 2 disponible(s)
Clients
Voir tout
3
clients enregistrés
Réparations en cours
Gérer les réparations
Aucune réparation en cours
Les places du garage sont disponibles


PLACES DISPONIBLES
2 / 2
Tableau de bord
Réparations
Interventions
Clients
Voitures
Statistiques
Admin Garage
Admin
Déconnexion
Interventions (TEST)
Gérez les types d'interventions du garage
Nouvelle intervention
NOM
DESCRIPTION
PRIX
DURÉE
ACTIONS
Batterie
Remplacement batterie
250 000 Ar
20min
Modifier
Supprimer
Changement courroie distribution
Kit distribution complet
1 200 000 Ar
4h 0min
Modifier
Supprimer
Changement plaquettes de frein
Remplacement des plaquettes avant
150 000 Ar
1h 30min
Modifier
Supprimer
Changement pneus
Remplacement de 4 pneus
800 000 Ar
2h 0min
Modifier
Supprimer
Climatisation
Recharge et contrôle climatisation
100 000 Ar
45min
Modifier
Supprimer
Diagnostic électronique
Lecture des codes erreurs
50 000 Ar
30min
Modifier
Supprimer
Révision complète
Révision des 50000 km
250 000 Ar
3h 0min
Modifier
Supprimer
Vidange
Vidange huile moteur complète
100 000 Ar
1h 0min
Modifier
Supprimer
localhost:3000/interventions