Documentation Technique
Application de Gestion de Réparations

Introduction

Ce projet a été réalisé lors d'un stage au garage 974Car, situé à La Réunion et spécialisé dans l'entretien et la réparation de véhicules thermiques, hybrides et électriques. Afin d'optimiser sa gestion interne, l'entreprise souhaitait un outil numérique pour centraliser les données clients, véhicules, réparations et facturation.

SOMMAIRE

Prérequis

Le projet fonctionne exclusivement dans un environnement Docker, nécessitant uniquement Docker et Docker Compose, sans installation locale supplémentaire de PHP, PostgreSQL ou autres services.

Installation & Lancement avec Docker

ÉTAPES

```
### 1. Cloner le projet
git clone https://github.com/lmoutanin/97Car-symfony.git
cd 97Car-symfony

### Lancer les conteneurs
docker compose up -d --build
```

Cela lance :

- **PHP et APACHE**
- **PostgreSql 15**
- **Adminer**

Accéder aux services:

- **Application web :** http://localhost/
- **Adminer (interface de gestion PostgreSQL) :** http://localhost:8181/

Accéder aux services:

Service	Valeur
SGBD	PostgreSQL
Serveur	database
Utilisateur	user
Mot de passe	password
Base	97Car

📦 Installation des dépendances front-end

```
composer req easycorp/easyadmin-bundle:4.x-dev -W

apt update && apt install -y postgresql-client

mv assets/styles/app.css assets/styles/app.scss

curl -o /usr/local/bin/n
https://raw.githubusercontent.com/visionmedia/n/master/bin/n
chmod +x /usr/local/bin/n
n stable

npm install sass-loader@^16.0.1 sass --save-dev
npm install bootstrap @popperjs/core bs-custom-file-input --save-
dev
npm install
```

🚀 Compilation des assets

```
symfony run npm run dev
symfony run -d npm run watch
```

📩 Import du trigger SQL

Un fichier SQL est disponible dans assets/import/.
Pour activer le trigger de mise à jour automatique des montants des factures :

1. Rendez-vous sur Adminer : http://localhost:8181
2. Connectez-vous à la base 97Car
3. Importez manuellement le fichier SQL situé dans assets/import/

Packages PHP Symfony essentiels

Package	Description
symfony/framework-bundle	Framework Symfony principal
symfony/orm-pack	Pack pour installer Doctrine ORM
doctrine/doctrine-bundle	Intégration Doctrine dans Symfony
doctrine/doctrine-migrations-bundle	Gestion des migrations de base
symfony/security-bundle	Gestion de la sécurité, authentification, rôles
symfony/twig-bundle	**Moteur de templates Twig**
symfony/validator	Validation des données
symfony/form	Création et gestion des formulaires
symfony/maker-bundle	Génération de code utile (entités, contrôleurs...)
symfony/dotenv	Gestion des variables d'environnement (.env)

.ENV

sert à stocker des **variables d'environnement** pour **configurer une application**. On utiliser les **variables** du fichier **.env** dans **docker-compose.yml . Il faut changer les valeurs de user et de password pour sécuriser l'application**

```
POSTGRES_DB=97Car
POSTGRES_PASSWORD=password
POSTGRES_USER=user
POSTGRES_VERSION=16
```

STRUCTURE DU PROJET

```
Projet-974Car/
│
├── src/
│   ├── Controller/
│   │   ├── Admin/
│   │   │   ├── ClientCrudController.php
│   │   │   ├── DashboardController.php
│   │   │   ├── FactureCrudController.php
│   │   │   ├── TypeReparationCrudController.php
│   │   │   └── VoitureCrudController.php
│   │   └── SecurityController.php
│   │
│   ├── Entity/
│   │   ├── Admin.php
│   │   ├── Client.php
│   │   ├── Facture.php
│   │   ├── FactureTypeReparation.php
│   │   ├── TypeReparation.php
│   │   └── Voiture.php
│   │
│   └── Form/
│       ├── FactureType.php
│       └── FactureTypeReparationType.php
│
├── config/
│   └── ... (fichiers config Symfony)
│
├── public/
│   └── index.php (point d'entrée Symfony)
│
├── Dockerfile
├── docker-compose.yml
├── .env
├── composer.json
├── README.md
└── db_data/          # Volume de persistance des données PostgreSQL
```

Organisation du projet Symfony

/src
- **Controller** # Contrôleurs HTTP (logique de gestion des requêtes)
- **Entity** # Entités ORM liées aux tables PostgreSQL
- **Form** # Formulaires Symfony (gestion des inputs utilisateurs)
- **Repository** # Accès aux données spécifiques (requêtes personnalisées)
- **Security** # Gestion de la sécurité et authentification

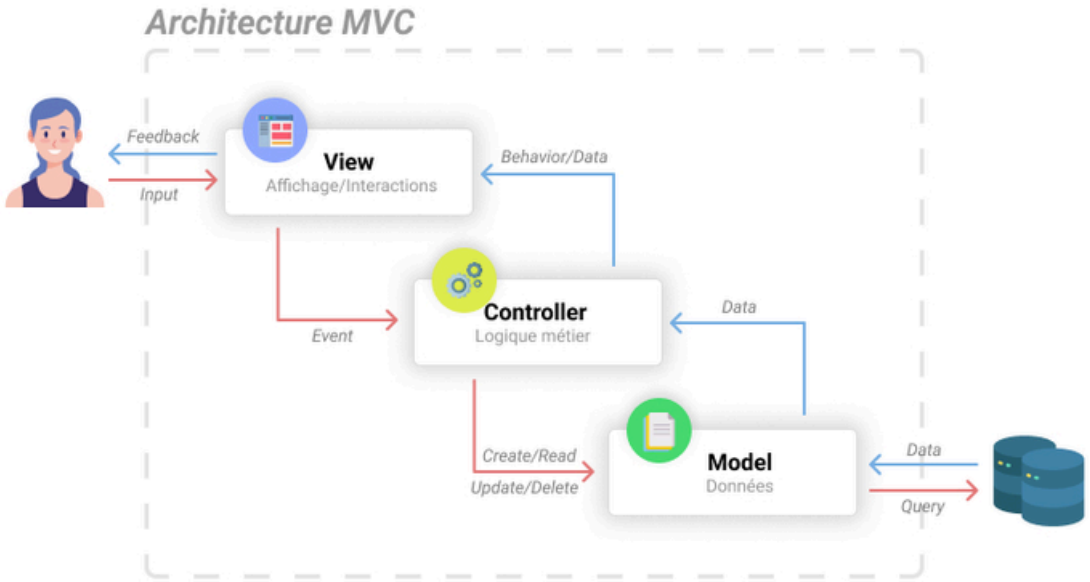
/config # Configuration des bundles, routes, services
/templates # Vues Twig pour affichage frontend
/public # Fichiers publics accessibles (CSS, JS, images)

ARCHITECTURE GLOBALE

DESCRIPTION GÉNÉRALE

Le projet de gestion du garage 974Car est une application web développée en PHP avec le framework Symfony, utilisant une architecture MVC (Modèle-Vue-Contrôleur). L'application suit une architecture client-serveur classique, avec une API backend qui gère la logique métier et les données, et une interface utilisateur accessible via un navigateur web.
La communication entre les composants est assurée via HTTP, avec des échanges de données en JSON ou HTML.

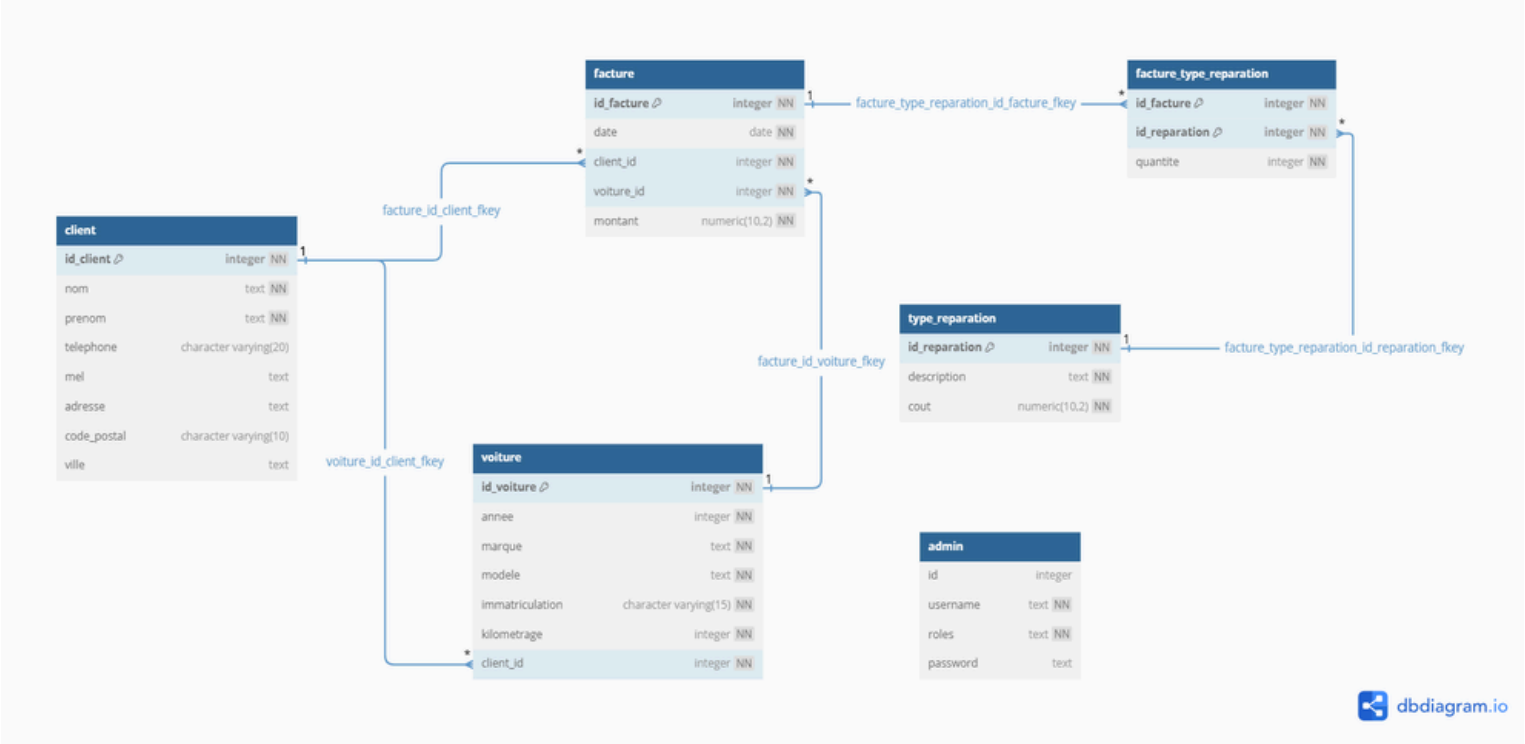
SCHÉMA D'ARCHITECTURE



- **Navigateur / Client :** Interface utilisateur web développée avec Twig (moteur de templates Symfony).
- **Serveur web / Backend :** Application Symfony qui traite les requêtes, exécute la logique métier, et communique avec la base de données.
- **Base de données PostgreSQL :** Stockage persistant des données (clients, voitures, factures, réparations).
- **Docker Compose :** Conteneurisation de l'application et de la base de données, garantissant un environnement cohérent et facilement déployable.

BASE DE DONNÉES

La base de données utilisée est PostgreSQL. Elle est conçue selon un modèle relationnel permettant de gérer efficacement les données clients, véhicules, réparations, factures, ainsi que les utilisateurs administrateurs.
Le modèle relationnel comprend les tables suivantes :



CONTENEURISATION AVEC DOCKER

L'application est déployée dans un environnement Docker pour assurer la portabilité et la facilité d'installation :

- Un conteneur exécute le serveur PHP avec Symfony.
- Un conteneur distinct exécute PostgreSQL, avec un volume Docker (db_data) pour la persistance des données.
- Docker Compose orchestre ces conteneurs, facilite le démarrage, l'arrêt, et la gestion des services.

Fonctionnalités principales

L'application de gestion du garage 974Car intègre plusieurs fonctionnalités essentielles destinées à faciliter la gestion quotidienne des clients, véhicules, réparations et factures. Ces fonctionnalités couvrent l'ensemble des besoins exprimés dans le cahier des charges, avec une interface utilisateur intuitive et un système sécurisé.

1 Gestion des clients

Création : Ajout de nouveaux clients avec leurs coordonnées complètes (nom, prénom, téléphone, adresse, email).
Modification : Mise à jour des informations client.
Suppression : Suppression des clients, avec gestion des relations (voitures et factures associées).
Consultation : Visualisation des fiches clients avec historique des véhicules et factures.

2 Gestion des véhicules

Ajout : Enregistrement des voitures avec marque, modèle, immatriculation, kilométrage et association à un client.
Modification : Mise à jour des données véhicules.
Suppression : Suppression d'un véhicule sans supprimer le client.
Consultation : Affichage détaillé du véhicule et de ses réparations.

3 Gestion des réparations

Référentiel : Création et gestion d'un catalogue des types de réparations avec coût unitaire.
Association : Possibilité d'ajouter des réparations à une facture avec quantité.

4 Gestion des factures

Création : Génération de factures liées à un client et un véhicule, avec sélection des réparations effectuées.
Calcul automatique : Le montant total de la facture est calculé automatiquement à partir des réparations associées.
Modification & suppression : Gestion complète du cycle de vie des factures.
Consultation : Visualisation des factures détaillées avec historique.

5 Authentification et gestion des rôles

Sécurisation : Système de connexion basé sur Symfony Security.
Rôles utilisateurs : Gestion des droits d'accès avec au moins deux rôles principaux :
Admin : Accès complet à toutes les fonctionnalités.
Utilisateur standard : Accès limité selon les permissions définies.
Protection des routes : Accès aux pages restreint en fonction des rôles.

4.6 Tableau de bord (Dashboard)

Synthèse : Résumé des indicateurs clés (nombre de clients, voitures, factures).
Statistiques : Visualisation simple des totaux et données importantes pour le suivi de l'activité.

Diagramme de cas d'utilisation

Le diagramme ci-dessous présente les cas d'utilisation principaux de l'application, illustrant les interactions entre les utilisateurs et le système.