Projet Jardinier – Symfony 6

Objectif

Développement d'une **application web** pour la société **Le P'tit Jardinier**, spécialisée dans l'entretien de jardins, notamment la **taille de haies**. L'objectif principal est de permettre aux utilisateurs de :

- visualiser différentes haies et catégories,
- créer un **devis personnalisé en ligne**,
- gérer les données de manière sécurisée via une interface Symfony.

Architecture technique

Modèle MVC (Model - View - Controller)

- **Model (Entity)** :
  Entités principales :
  - Haie (nom, hauteur, prix, etc.)
  - Categorie (type de haie)
    Utilisation d'**annotations Doctrine** pour gérer les relations et la persistance.
- **View (Twig)** :
  Les vues sont construites avec le moteur de templates **Twig** :
  - index.html.twig
  - devis.html.twig
  - modifier_haie/index.html.twig
- **Controller** :
  Contrôleurs Symfony pour gérer la logique métier :
  - HaieController pour la gestion des haies
  - DevisController pour le formulaire de simulation de devis
  - SecurityController pour l'authentification

Développement par étapes

Partie 1 : Contrôleur et Vue

- Mise en place du **routage** via annotations (@Route) dans les contrôleurs.
- Création de contrôleurs de test pour afficher les vues.
- Récupération de données via méthode **POST**.
- Stockage temporaire des informations via **SESSION Symfony** (par exemple : informations du devis).

Partie 2 : Modèles de données avec ORM

- Utilisation de **Doctrine ORM** :
    - Mapping des entités Haie et Categorie à la base MySQL.
    - Commandes CLI (make:entity, doctrine:migrations) pour créer et migrer les tables.

Partie 3 : Formulaires

- Utilisation du **FormBuilder Symfony** :
    - Génération de formulaires liés aux entités (HaieType, CategorieType, DevisType)
    - **Validation** des champs (annotations @Assert)
    - Soumission, affichage d'erreurs, et traitement des données.

Sécurité et authentification

- Mise en place du système de sécurité Symfony :
    - Authentification par email + mot de passe
    - Rôles définis (ROLE_USER, ROLE_ADMIN)
- Droits d'accès configurés dans security.yaml :
    - **Admin** : accès complet (CRUD)
    - **Utilisateur** : accès en lecture + simulation de devis

---

Technologies utilisées

Outil	Détail
Langages	PHP 8, HTML, CSS
Framework	Symfony 6

Outil	Détail
Base de données	MySQL
ORM	Doctrine
Templating	Twig
IDE	Visual Studio Code
Versioning	Git + GitHub
Serveur local	WampServer

Déploiement

- Déploiement **local** avec **WAMP** (Apache + MySQL)
- Base de données initialisée via migration (doctrine:migrations:migrate)
- Configuration dans .env (accès BDD)

Fonctionnalités principales

- Liste dynamique des haies (par catégorie)
- Consultation de tarifs
- Formulaire de devis avec enregistrement temporaire
- Authentification utilisateur / admin
- Interface d'administration CRUD complète

Liens utiles

- Documentation Symfony du projet