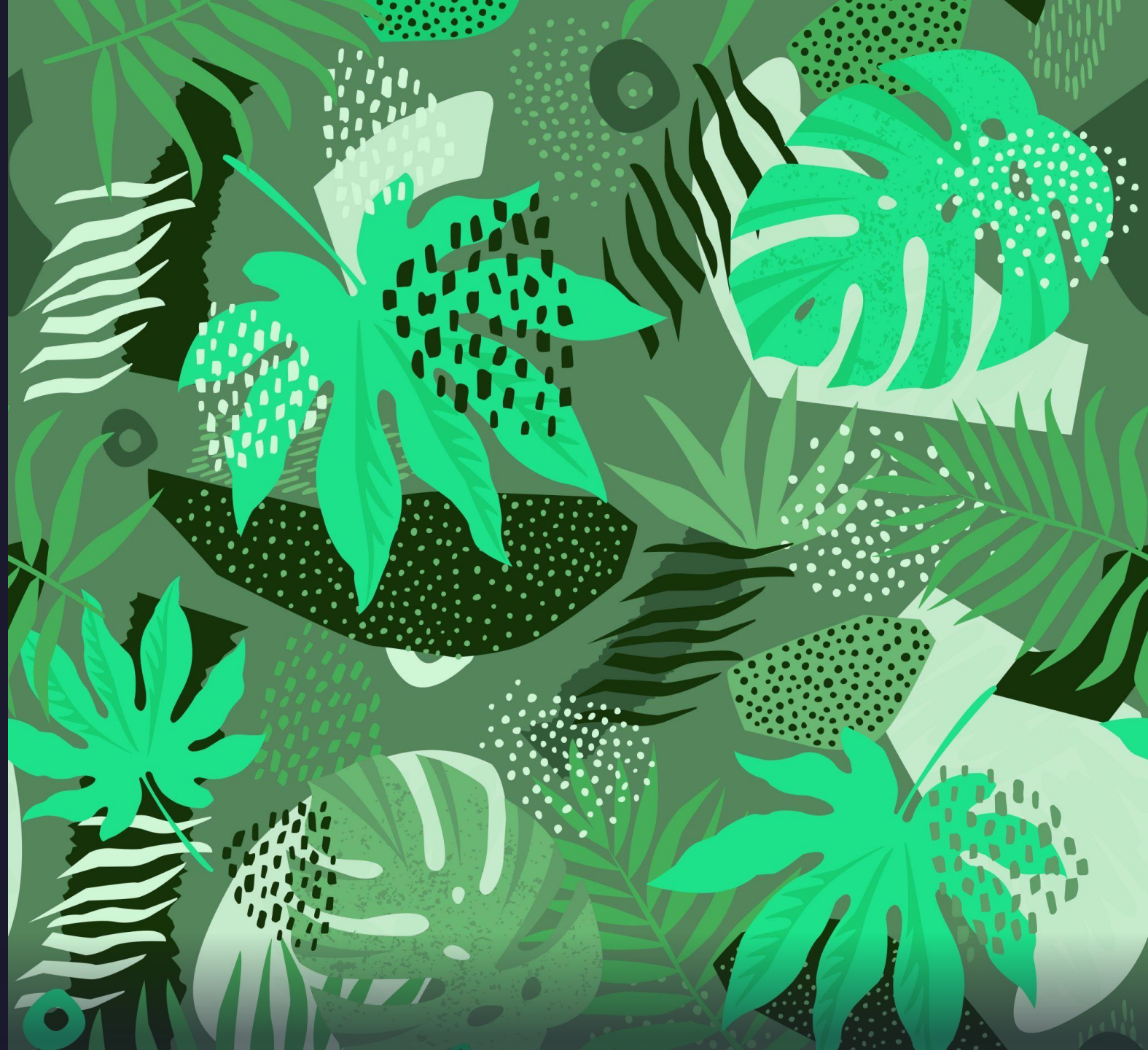
Cahier des charges technique

Projet : Jardins de Cocagne

Auteur : MARTIN Mathis

Date : 16 octobre 2025



Présentation du projet

- Le projet **Jardins de Cocagne** a pour but de **numériser la gestion des activités** d'un jardin solidaire.
  L'application doit faciliter la gestion :
- des **parcours d'insertion** des salariés,
- des **abonnements aux paniers de légumes**,
- et de la **logistique des livraisons**.
- Ce développement a pour ambition d'offrir un outil moderne, performant et simple d'utilisation, aussi bien pour les équipes internes que pour les adhérents.

Présentation de l’étudiant

- Je m’appelle **Mathis Martin**, étudiant en **BUT Informatique**.
  Je réalise ce projet dans le cadre de ma formation, avec pour objectif d’approfondir mes compétences en **développement web full-stack** et en **gestion de projet**.

- J’interviens sur l’ensemble du projet :

- conception,

- développement du Back-End et du Front-End,

- intégration de la base de données,

- mise en place de l’environnement Docker,

- et déploiement final.

Architecture globale

- Le projet repose sur une **architecture web moderne** organisée autour de trois couches principales :
- **Back-End** : logique métier et gestion des données,
- **Front-End** : interface utilisateur et interaction,
- **Base de données** : stockage structuré et sécurisé des informations.
- L’ensemble est conteneurisé à l’aide de **Docker**, pour garantir un environnement stable et homogène.

- Choix techniques

- 1. Back-End
- Framework : Symfony (PHP)
- **Comparaison :**
- **Laravel** : framework moderne, mais plus complexe à configurer et moins adapté à une application de gestion avec une logique métier forte.
- **Symfony** : très bien documenté, structuré et largement utilisé dans le milieu professionnel.
- **Choix retenu : Symfony 7 avec PHP 8.2**
  ➡️ **Ce choix garantit la stabilité**, la **maintenabilité** et la **performance** du Back-End, tout en permettant une bonne séparation entre les couches logiques.
- Symfony inclut des outils facilitant la **gestion des rôles utilisateurs** et le respect du **RGPD** pour la protection des données personnelles.

- ORM : Doctrine

- J’ai choisi **Doctrine ORM** pour gérer les données sous forme d’objets PHP.
  Il remplace l’écriture manuelle des requêtes SQL et assure la **sécurité des interactions** avec la base.
- **Avantages principaux :**
- Protection contre les **injections SQL**.
- Gestion automatique des **migrations** lors des modifications de structure.
- Gain de temps pour le développement Back-End.
- Doctrine permet d’écrire des requêtes plus lisibles et plus sûres, directement en PHP.

- 2. Front-End

- Technologies utilisées : HTML5 / CSS3 / Twig
- **Comparaison :**
- **ReactJS** aurait permis une interface plus dynamique, mais la mise en place aurait été plus complexe et moins cohérente avec Symfony.
- **Twig (intégré à Symfony)** offre un rendu serveur simple et efficace.
- **Choix retenu : Twig avec HTML5 et CSS3**
  ➡️ **Cela me permet d'avoir une séparation claire** entre la logique et l'affichage, tout en assurant une interface **responsive** et compatible avec tous les supports (mobile, tablette, ordinateur).
- Les balises sémantiques HTML5 et les propriétés avancées CSS3 assurent une navigation fluide et un affichage professionnel.

Cartographie : Leaflet.js

- Pour afficher la carte des **dépôts et points de retrait**, j'ai retenu **Leaflet**, une bibliothèque JavaScript **open source et légère**.
- **Comparaison :**
- **Google Maps API** : plus complète, mais payante à grande échelle.
- **Leaflet** : gratuite, rapide et simple à intégrer dans Twig.
- **Choix retenu : Leaflet.js**
  ➡️ **Ce choix offre un excellent compromis entre performance, simplicité et personnalisation**.
  Les coordonnées GPS stockées dans la base MySQL sont injectées dynamiquement dans la carte via Doctrine.

- 3. Base de données

- Système : MySQL 8.0
- J’ai choisi **MySQL 8.0** pour sa fiabilité et sa bonne intégration avec PHP/Symfony.
  Il assure la **cohérence**, la **sécurité** et la **rapidité** des échanges.
- **Comparaison :**
- **PostgreSQL** : plus puissant pour les calculs complexes, mais plus lourd à configurer.
- **MySQL** : plus léger, plus intuitif pour un environnement web.
- **Choix retenu : MySQL 8.0**
  ➡️ **Parfaitement adapté à une application web de gestion.**
- La base est encodée en **UTF-8mb4**, et les données personnelles sont stockées dans le respect du **RGPD**.

- Outil de gestion : phpMyAdmin

- Pour la gestion de la base de données, j’utilise **phpMyAdmin**, une interface web pratique qui me permet de visualiser et modifier les tables facilement.
  Il est intégré dans **Docker** pour être accessible localement dès le lancement du projet.

• Environnement de développement : Docker

- J'utilise **Docker** pour uniformiser l'environnement de développement.
  Chaque service (PHP, MySQL, Nginx, phpMyAdmin) est isolé dans son propre conteneur, ce qui rend le projet plus fiable et plus facile à déployer.
- **Avantages :**
- Même configuration en local et en production,
- Déploiement rapide avec `docker-compose up`,
- Débogage facilité,
- Respect strict des versions des outils.
- **Choix retenu :**
  ➡️ Environnement Docker composé de conteneurs :
- `php-fpm` pour exécuter Symfony,
- `mysql` pour la base de données,
- `nginx` pour servir les pages web,
- `phpmyadmin` pour l'administration des données.

- Serveur web : Nginx

- J’ai choisi **Nginx** comme serveur web principal.
  Il est réputé pour sa capacité à gérer un grand nombre de connexions simultanées tout en restant léger.
- **Comparaison :**
- **Apache** : plus simple à configurer, mais moins performant en charge élevée.
- **Nginx** : plus rapide, plus moderne, et parfaitement adapté à Docker.
- **Choix retenu : Nginx**
  ➡️ **Il gère la distribution des fichiers statiques (HTML, CSS, JS, images) et redirige les requêtes dynamiques vers PHP-FPM.**
  Il ajoute également des en-têtes de sécurité HTTP pour renforcer la protection du site.

- Solution d’hébergement

- Le projet est prévu pour être **hébergé sur un serveur VPS** (Virtual Private Server) afin de garder un contrôle complet sur l’environnement.
  Le déploiement se fera à l’aide de **Docker Compose**, ce qui garantit une configuration identique à celle du développement.
- **Comparaison :**
- **Hébergement mutualisé** : moins cher, mais peu flexible.
- **VPS** : plus de liberté, meilleure sécurité, et configuration personnalisée.
- **Choix retenu : VPS hébergé via Docker.**

- Solution de tests

- Pour assurer la qualité du projet, j'utilise plusieurs types de tests :
- **Tests unitaires** avec **PHPUnit** pour vérifier la logique interne du code Symfony,
- **Tests fonctionnels** pour valider les formulaires et les routes,
- **Tests manuels** pour vérifier le bon affichage sur différents navigateurs et supports.
- Un script d'exécution automatisé est prévu dans Docker pour lancer les tests localement avant chaque mise à jour du projet.

- Représentation du Jardin de Cocagne

- Pour proposer une expérience plus immersive, j'ai intégré une **représentation 3D du jardin** développée avec **Unity**.
  Cette simulation, compilée en **WebGL**, est directement intégrée dans la page web.

- Elle permet aux adhérents et partenaires d'explorer virtuellement les parcelles de culture et d'avoir une meilleure vision du fonctionnement du jardin.
  Des éléments interactifs pourront être ajoutés ultérieurement.

Conclusion

- Ce projet me permet de concevoir une **application web complète**, moderne et adaptée aux besoins du Jardin de Cocagne.

- L’utilisation d’un environnement conteneurisé et de technologies reconnues (Symfony, MySQL, Nginx, Docker) garantit une solution **performante, évolutive et professionnelle**.