LES JARDINS DE COCAGNE

Cahier Des Charges Technique

UNIVERSITÉ DE LORRAINE | IUT Saint-Dié

Téo VINCENT | Ewan GAILLIEGUE

15/10/2025

CONTEXTE ET OBJECTIFS

Contexte du projet

- Les jardins de cocagne veulent un site web moderne et intuitif
- valoriser la production locale et faciliter le lien avec les adhérents
- Le site doit proposer une expérience interactive et pratique (infos, adhésion, suivi).

Objectifs techniques

- Créer un site dynamique et responsif (React / Express / PostgreSQL).
- Intégrer une carte interactive pour localiser les points de distribution.
- Mettre en place un itinéraire dynamique (API de cartographie type Leaflet ou Google Maps).
- Ajouter un calendrier de livraison consultable et mis à jour en ligne.
- Garantir une interface simple pour la gestion interne (adhérents, paniers, tournées).

Contraintes principales

- Nouvelles technologies modernes
- Délais limités ⇒ solutions stables, open-source et faciles à maintenir.
- Respect du cahier des charges

Présentation de l'équipe

Composition de l'équipe

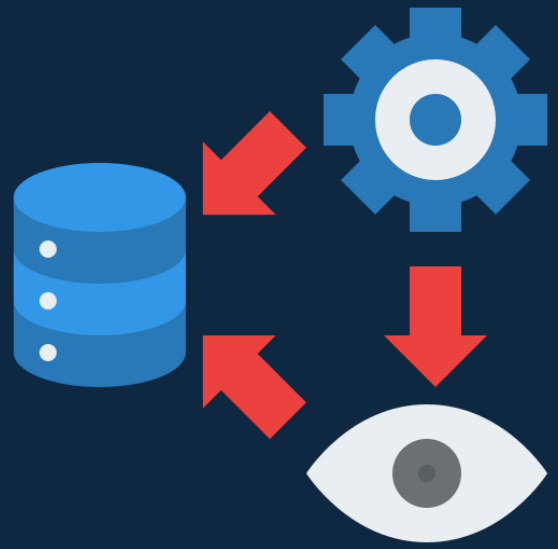
Ewan GAILLIEGUE

Etudiant en 3ème année de BUT INFO, Il est le lead développeur back-end, imagine la base de données et participe au développement front-end.

Téo VINCENT

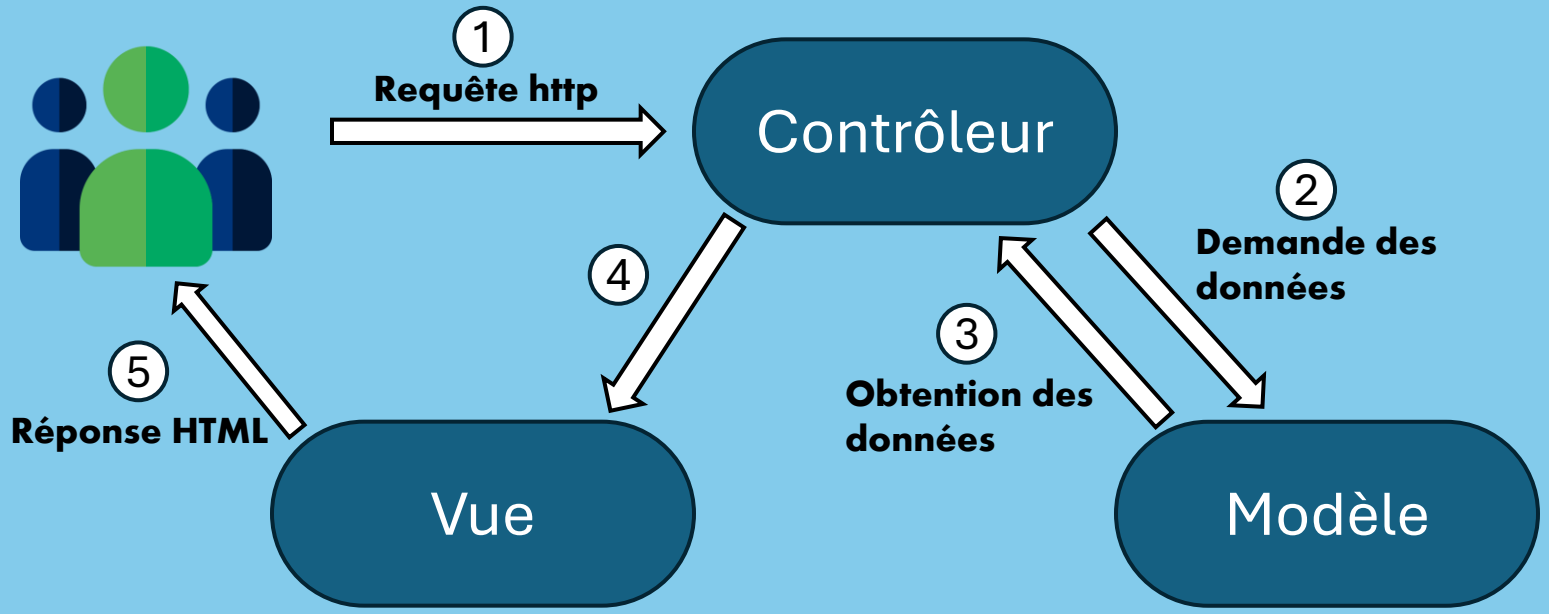
Etudiant en 3ème année de BUT INFO, Il est le lead développeur front-end, imagine le maquettage et participe au développement back-end

Architecture Globale

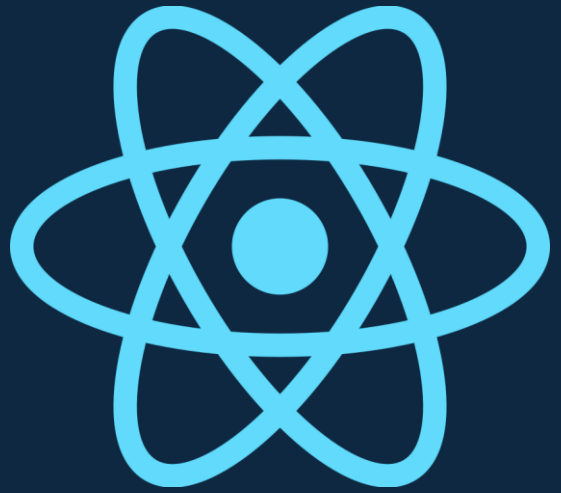


Architecture **MVC**

- Structure modulaire facilitant la maintenance et l'évolution du projet.
- Séparation claire entre les données, la logique métier et l'affichage.



Front-end



Technologie retenue : React

- React.js est une bibliothèque JavaScript open source développée par Meta.
- Elle permet de construire des interfaces utilisateur dynamiques et réactives à partir de composants réutilisables.
- Utilisée dans de nombreux projets web modernes (Airbnb, Netflix, Spotify).

Pourquoi ce choix?

- Permet une interface fluide et interactive pour les adhérents et administrateurs (paniers, calendrier, carte).
- Facilite la modularité : chaque section (adhésion, dépôt, panier) est un composant isolé.
- Compatible avec React-Leaflet pour la carte interactive et Axios pour la communication avec l'API Express.js.
- Parfaitement intégrable dans une architecture MVC côté front-end. Large communauté et documentation — un atout pour un projet d'étude à petite équipe

Avantages

Performance	Rendu rapide grâce au Virtual DOM
Réutilisabilité	Composants modulaires, maintenance facilitée
Interopérabilité	Compatible avec la plupart des API et frameworks
Écosystème riche	Nombreuses librairies (React Router, Formik, Zustand...)
Support communautaire	Forte base d'utilisateurs et mises à jour régulières

Inconvénients

Courbe d'apprentissage	Syntaxe JSX et logique des hooks à maîtriser
Mises à jour fréquentes	Nécessite une veille technique régulière
Non-framework complet	Nécessite d'ajouter des outils externes (routing, state management)

Back-end

Technologie retenue : Node.Js + Express.Js

- Environnement JavaScript côté serveur, léger et performant.
- **Express.js** : framework minimaliste facilitant la création d'API REST.
- Structure organisée selon le **modèle MVC** (Contrôleurs – Modèles – Routes).

Pourquoi ce choix?

- Permet une **communication fluide** entre le front (React) et la base de données.
- Grande **flexibilité** et compatibilité avec de nombreux modules npm.
- Idéal pour des projets web modernes et évolutifs.

Exemple de structure serveur

```
backend/
├── server.js
├── routes/
│   └── livraisonRoutes.js
├── controllers/
│   └── livraisonController.js
├── models/
│   └── livraisonModel.js
└── config/
    └── db.js
```

Base de Données

PostgreSQL

Base de données relationnelle – PostgreSQL

PostgreSQL est le système de gestion de base de données relationnelle choisi pour le projet *Les Jardins de Cocagne*.
Il offre stabilité, performance et une grande flexibilité pour la modélisation de données complexes (adhésions, dépôts, livraisons, paniers, facturation...).

Fiabilité et sécurité

Transactions ACID, gestion fine des permissions, sauvegardes intégrées.
Adapté à la gestion des données sensibles (adhérents, paiements).

Performance et évolutivité

Capable de gérer des milliers de lignes sans perte de performance.
Indexation avancée, vues matérialisées et requêtes complexes optimisées.

Richesse fonctionnelle

- Types personnalisés (ENUM, UUID) pour une structure claire
- Fonctions et déclencheurs (triggers) pour automatiser les règles de gestion
- Extensions utiles
  - uuid-ossp pour les identifiants uniques
  - pgcrypto pour la sécurité
  - PostGIS (optionnel) pour la géolocalisation des dépôts

Structure de la Base de Données

Structure – Base de données relationnelle

Objectifs

- Organiser de manière cohérente toutes les informations du site : adhérents, adhésions, paniers, dépôts, livraisons, production, facturation.
- Garantir la cohérence des données et faciliter les échanges avec l'API Express.js.

Domaine	Tables principales	Rôle
Adhésions	`foyer, adherent, adhesion`	Gérer les familles, adhérents et abonnements
Dépôts	`depot, creneau, foyer_depot_pref`	Localiser les points de retrait et leurs créneaux
Paniers & livraisons	`livraison, panier, contenu_panier`	Planifier et suivre les distributions
Production	`parcelle, culture, semis, recolte, produit`	Suivre la production maraîchère
Facturation	`facture, paiement`	Gérer les paiements et les factures des adhérents
Bénévolat (optionnel)	`volunteer_task, volunteer_signup`	Gérer les chantiers participatifs

Relations clé

- Un foyer peut avoir plusieurs adhésions (1—N)
- Une adhésion génère plusieurs paniers sur une saison
- Un panier est associé à un dépôt et à un contenu panier
- Les produits proviennent de cultures liées à des parcelles
- Une adhésion donne lieu à une ou plusieurs factures / paiements

Déploiement

Déploiement – Docker

Objectif du déploiement

- Garantir un environnement stable, reproductible et portable
- Simplifier la mise en ligne du site des Jardins de Cocagne
- Assurer une cohérence entre développement et production

Choix technologique: Docker

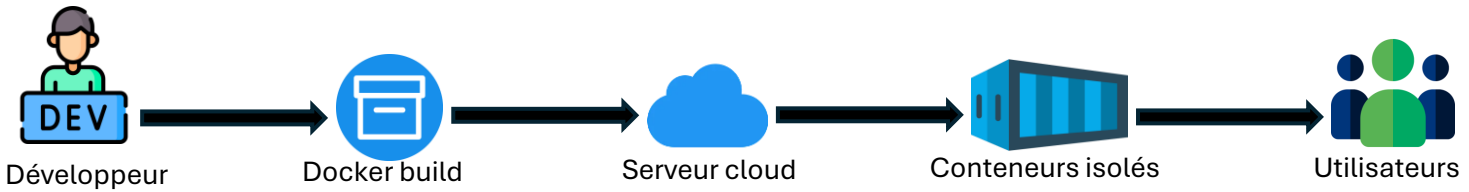
- Conteneurisation du backend (Express.js), du frontend (React) et de la base (PostgreSQL)
- Isolation des services : chaque composant tourne dans son propre conteneur
- Compatible avec tous les environnements (local, cloud, serveur Linux)

Avantages clés

- Rapidité de déploiement
  - Un seul fichier docker-compose.yml pour tout lancer
- Stabilité et reproductibilité
  - Zéro conflit de dépendances (Node, Postgres…)
- Maintenance simplifiée
  - Mises à jour par simple rebuild de l'image

Environnement cible

- Serveur Linux (Ubuntu) ou hébergement Docker Cloud
- Services : web -> Front-end React ; api -> back-end Express.js ;
  db -> PostgreSQL



Sécurité et contrainte technique

Sécurité et contrainte technique

- Authentification sécurisée via JWT (JSON Web Token) pour les utilisateurs et administrateurs
- Gestion des rôles : adhérent, salarié, administrateur
- Mots de passe hachés avec bcrypt avant stockage dans la base PostgreSQL Sessions protégées par HTTPS et tokens stockés de manière sécurisée
- Possibilité d'extension future : connexion via OAuth2 (Google, FranceConnect)

Sauvegarde et intégrité des données

- Sauvegardes quotidiennes automatisées de la base PostgreSQL (pg_dump)
- Conservation des sauvegardes sur 20 jours
- Données persistées dans des volumes Docker (ou base cloud managée)
- Contrôle d'intégrité : clés primaires/étrangères, contraintes de cohérence
- Politique de rétention et anonymisation conforme au RGPD

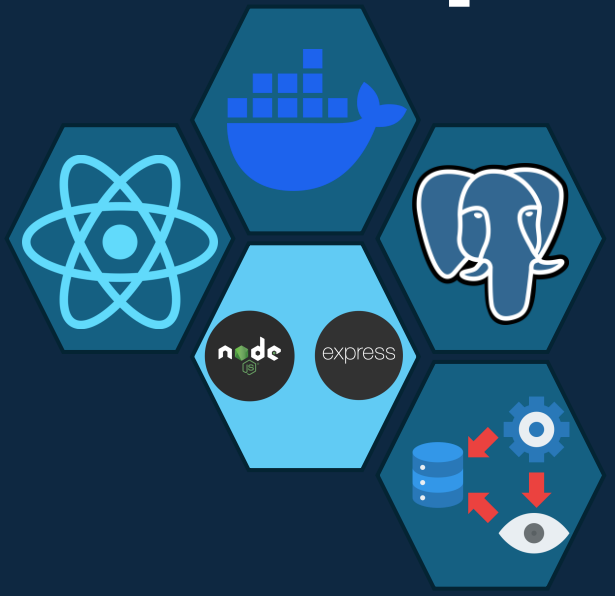
Standards web et bonnes pratiques

- Respect des normes OWASP Top 10 (injection, XSS, CSRF, etc.)
- Communication chiffrée (HTTPS) pour toutes les interactions client–serveur
- Validation des entrées côté client et serveur Structure du code conforme au modèle MVC (séparation des responsabilités)
- Tests automatisés (unitaires & API) intégrés au pipeline CI/CD

Contraintes techniques

- PostgreSQL pour la base, Express.js pour l'API, React pour le front
- Suivi de performance et alertes via outils de monitoring (Grafana, UptimeRobot)
- Politique de mise à jour mensuelle (packages, dépendances, images Docker)

Récapitulatif Technique



Domaine	Technologie retenue	Rôle principale
Front-end	React.js	Interface utilisateur dynamique, composant réutilisables, design responsive
Back-end	Express.js (Node.js)	API REST, gestion des routes, authentification et logique métier
Base de données	PostgreSQL	Stockage relationnel, gestion des adhésions, paniers, livraisons, production
Carte & itinéraires	Leaflet.js + API OpenStreetMap	Affichage des dépôts, itinéraires de livraison
Déploiement & hébergement	Docker	Conteneurisation, cohérence entre dev et prod, hébergement cloud
CI/CD	GitHub Actions	Tests, build et déploiements automatisés
Sauvegarde & Monitoring		Sauvegardes automatisées, supervision du service
Authentification & sécurité	JWT + bcrypt + HTTPS	Protection des accès, mots de passe hachés, communication sécurisée

Points forts

- **Architecture claire (MVC)** → séparation Front / Back / DB
- Technologies de développement **modernes et bien documentés**
- **Maintenance facilitée** grâce à Docker et CI/CD
- **Scalabilité** possible (microservices, cloud)
- **Expérience utilisateur fluide et interactive** via React + carte dynamique

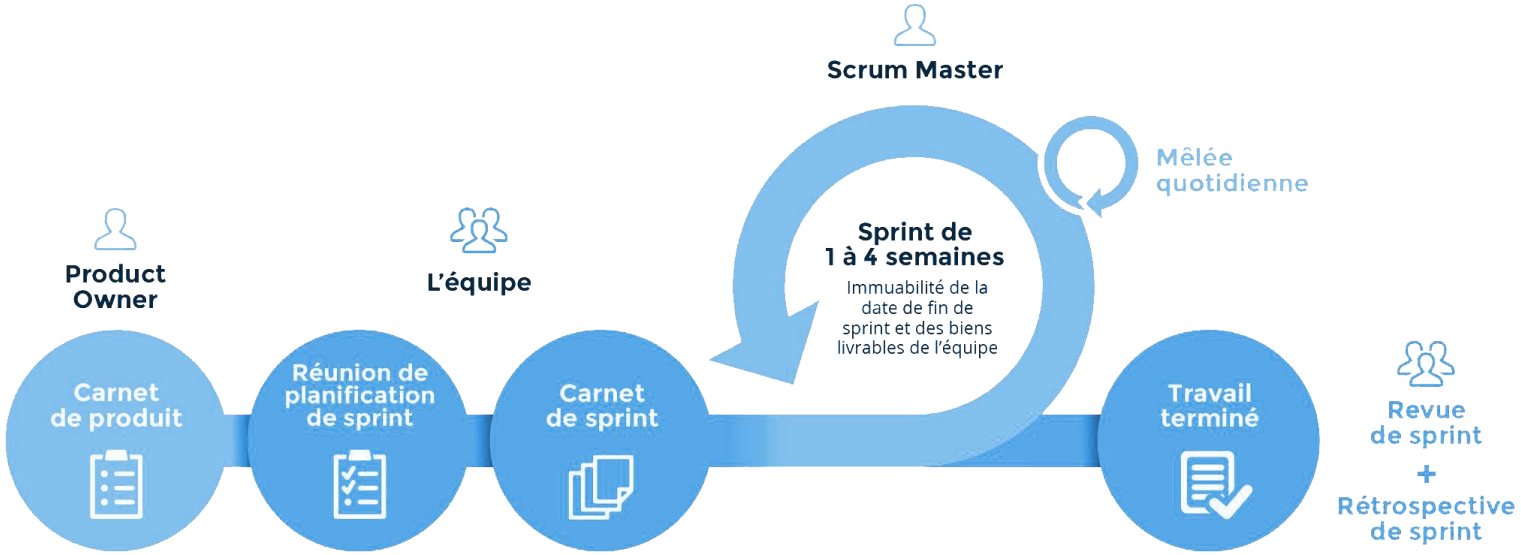
Organisation et Gestion de projet

Méthode agile – Scrum

Approche Agile choisie pour sa flexibilité et son adaptation aux besoins du client.
Découpage du projet en sprints courts (1 à 2 semaines).

À chaque sprint :
- objectifs précis,
- livrables fonctionnels,
- revue et ajustement avec le client.

Avantages :
- meilleure **réactivité** face aux changements,
- **communication continue** dans l'équipe,
- intégration progressive des fonctionnalités (adhésion, carte, calendrier…).



Organisation et Gestion de projet

Outils – Planification et Gestion

TRELLO	- Suivi des tâches par sprint (tableau Kanban)	- Simple, collaboratif, visuel
GanttProject	- Planification temporelle globale avec un diagramme de Gantt	- Vue chronologique claire, dépendances visibles
Git / GitHub	- Gestion de versions, suivi du code, CI/CD	- Collaboration, traçabilité, intégration avec Trello
Discord	- Communication et synchronisation rapide	- Travail d'équipe fluide, décisions en temps réel
Figma / Canva	- Maquettage visuel et wireframes	- Interface intuitive, collaboration en direct
Google Drive	- Documentation et centralisation des fichiers	- Partage rapide, historique des modifications

LES JARDINS DE COCAGNE

Cahier Des Charges Technique

UNIVERSITÉ DE LORRAINE | IUT Saint-Dié

Téo VINCENT | Ewan GAILLIEGUE

15/10/2025