Cahier des charges technique

Cognet Matthéo / Erik Barba

15/10/2025

UNIVERSITÉ DE LORRAINE | IUT Saint-Dié-des-Vosges

Présentation d'équipe

Erik Becerra Barba

- *Rôle:* Responsable de la documentation, intégration frontend et appui technique backend
- *Responsabilités:*
  - Rédaction technique de cahiers des charges et présentations
  - Participation aux contrôleurs Symfony
  - Élaboration de la documentation fonctionnelle et technique
  - Définition des interfaces utilisateur et validation UX
  - Intégration des modules externes (calendrier)
  - Contribution aux tests fonctionnels et à la traçabilité des livrables

Matthéo Cognet

- *Rôle*: Développeur principal backend et architecte de la base de données
- *Responsabilités:*
  - Développement backend avec Symfony, Doctrine ORM et API REST
  - Modélisation relationnelle évolutive avec PostgreSQL
  - Conteneurisation avec Docker et configuration serveur
  - Mise en place de tests unitaires et gestion des branches Git
  - Conception de l’architecture technique.
  - Implémentation des modules de gestion (adhésions, abonnements, livraisons)
  - Intégration des modules externes (cartographie)
  - Suivi des sprints et revue de code

1. Introduction

a. Rappel du contexte

Dans les années 2000, un logiciel dit de « gestion des adhérents » a été conçu par un directeur de Jardin de Cocagne sur une base Microsoft Access. Ses fonctionnalités principales sont la gestion des adhérents, des abonnements aux paniers et des règlements associés, l'édition de feuilles de route pour la préparation et la livraison des paniers. Un export de données vers un logiciel de comptabilité est possible. Cet outil était encore utilisé en juin 2016 par 65% des Jardins de Cocagne. Il répond assez bien aux besoins de base en termes de gestion d'abonnement aux paniers, mais les formules commerciales et le contexte ont fortement évolué, nécessitant à présent une modernisation de l'outil (internet, automatisation des processus, ouverture des modes de ventes).

A ce jour 75% des Jardins de Cocagne ont manifesté un besoin urgent de développer un nouvel outil de gestion commerciale.

b. Objectif du projet

Développer une solution numérique moderne, évolutive et sécurisée permettant aux Jardins de Cocagne de gérer efficacement leurs activités commerciales: adhésions, abonnements, livraisons, paiements et itinéraires.

Le système devra être accessible via le web, s'intégrer avec des outils externes (cartographie, calendrier, comptabilité) et respecter les obligations légales en matière de protection des données.

2. Architecture générale du système

a. Schéma global de l'application

Exemples :

[Utilisateur] <-> [Frontoffice Web] <-> [API Symfony] <-> [Base de données PostgreSQL]

[Gestionnaire] <-> [Backoffice Web] <-> [API Symfony] <-> [Base de données PostgreSQL]

[Modules externes : Carte, Calendrier, Itinéraire] <-> [API Symfony]

Schéma en trois couches :

- Frontoffice (interface client/adhérent)
- Backoffice (interface gestionnaire)
- Base de données PostgreSQL
- API REST (pour communication entre front et back, et interconnexion avec d'autres outils)
- Serveur d'hébergement (cloud ou dédié, sécurisé)
- Modules externes : Carte interactive, calendrier, gestion des itinéraires

b. Architecture en couches

Couche présentation :

- Interfaces web responsives (Twig, Bootstrap, JS)
- Accès sécurisé (authentification, gestion des rôles)

Couche logique métier :

- Symfony (contrôleurs, services, entités)
- Gestion des règles métier (abonnements, livraisons, adhésions, commandes)

Couche accès aux données :

- Doctrine ORM pour PostgreSQL
- Modèle relationnel modulaire et évolutif

b. Hébergement sécurité modularité

Hébergement

- Un serveur de base de données PostgreSQL avec réplication et sauvegardes automatisées.
- Un stockage sécurisé pour les fichiers et journaux.

Sécurité

- Chiffrement de données sensible en transit (TLS 1.2+).
- Protection contre les vulnérabilités courantes (injections SQL, XSS, CSRF).

API & Modules externes

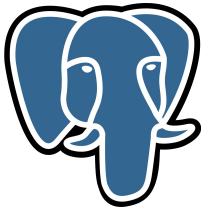
- Séparation entre frontend, backend, API et base de données
- Conteneurisation avec docker
- Intégration progressive de modules externes (cartographie, calendrier, itinéraires) via API REST.

3. Choix des technologies

Langage de développement

Frontend

Base de données



Cadriciel principal

Gestion de dépendances et versionning

Conteneurisation

4. Modélisation de la base de données

Structure

- **structure**: Définit l'entité principale.

Saisons et Calendrier

- **saison**: Périodes de livraison.
  - *Relation*: Structure
- **semain_non_livrable**: Semaines sans livraison.
  - *Relation:* Saison
- **jour_ferie**: Jours Fériés.
  - *Relation*: Saisons
- **jour_ferie_decalage**: Décalages liés aux jours fériés.
  - *Relation*: jour_ferie, tournee_livraisons

Logistique de Livraison

- **tournee_livraison**: Tournées de livraison.
  - *Relation*: structure_id
- **tournee_depot**: Association entre tournée et point de dépôt.
  - *Relation*: structure_id
- **point_depot**: Points de dépôt.
  - *Relation*: structure_id
- **point_depot_jour**: Jour d'ouverture des points de dépôt.
  - Relation: point_depo_idt

Clients et Adhesíons

- **client**: informations sur le client.
  - *Relation*: structure_id
- **adhesion_type**: Types d'adhésion.
  - *Relation*: structure_id
- **adhesion**: Adhésion active
  - *Relation*: client_id, adhesion_type_id

Produits et Composition

- **produit_type**: Catégories de produits.
- **produit**: Produit.
  - *Relation*:structuire_id, produit_type_id
- **produit_composition**: Produits composés.
  - *Relation*: produit_id, composant_id

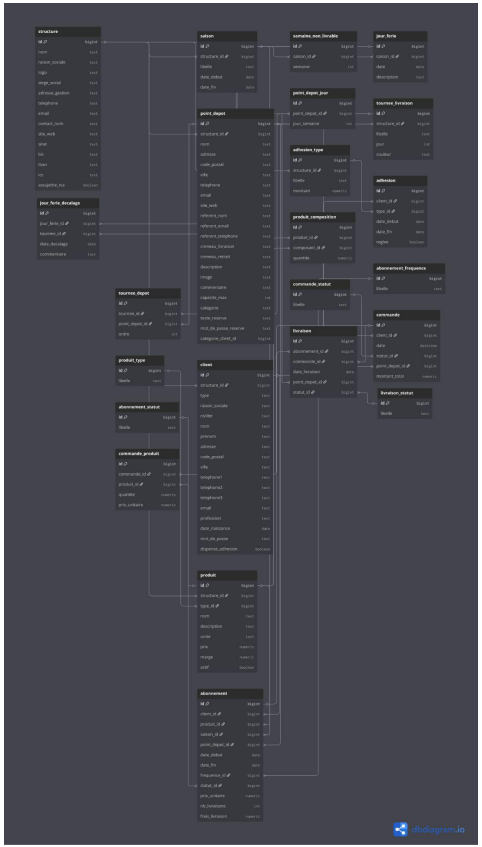
4. Modélisation de la base de données

Abonnements et Commandes

- **abonnement_frequence**: Fréquence et statut de l'abonnement.
- **abonnement**: Abonnement actif.
    - *Relation*: client_id, produit_id, saison_id, point_depot_id, frequence_id

Livraison

- **livraison_status**: Statut de la livraison.
- **livraison**: Enregistrement de la livraison.
    - *Relation*: abonnement_id, cammande_id, point_depot_id, livraison_statut_id



5. Organisation des Développements

Stratégie de gestion des branches Git

Branche main : version stable et prête pour mise en production.

Branche develop : intégration des fonctionnalités validées en cours de cycle.

Branches feature/ : développement de nouvelles fonctionnalités isolées.

Branches hotfix/ : corrections urgentes directement appliquées sur main.

Méthodologie de travail

Approche Agile Scrum

Itérations courtes avec objectifs définis (sprints de 2 à 3 semaines).

Revue de code systématique avant intégration.

Tests unitaires et fonctionnels intégrés au cycle.

Outils de suivi

Trello pour gestion des tâches et suivi de backlog.

Tableaux Kanban pour visualiser l'avancement (To Do, In Progress, Done).

6. Support des Applications

Compatibilité Multi-plateforme

Sécurité des accès

Sécurité et accès

Authentification par identifiant/mot de passe chiffré (hashage, salage)

Gestion des rôles (administrateur, contributeur, utilisateur simple) avec droits différenciés

Sessions sécurisées avec expiration et renouvellement de jeton

Connexions chiffrées (HTTPS/TLS 1.2+) pour toutes les communications

Protection native contre attaques XSS, CSRF et injections SQL via Symfony

Journalisation des accès et tentatives de connexion pour suivi de sécurité

7. Solutions pour la Carte et l'Itinéraire

	Leaflet.js	Google Maps JavaScript API
Licence	Open source	Propriétaire – Payant selon usage
Personnalisation	Très flexible avec plugins et styles CSS	Personnalisation limitée aux options Google
Dépendance externe	Faible – fonctionne avec tuiles OSM ou locales	Forte – dépendance aux serveurs Google
Fonctionnalités	Cartes, marqueurs, couches, popups	Cartes, itinéraires, géolocalisation, Street View
API d'itinéraire	Intégration via OpenRouteService ou OSRM	Native avec Directions API
Performance mobile	Optimisé pour web et mobile	Très performant, mais plus lourd
Quota et limites	Aucun quota imposé	Limites de requêtes par jour/mois
RGPD et hébergement	Compatible avec hébergement local sécurisé	Données hébergées par Google

	OpenRouteService (ORS)	Google Directions API
Licence	Open source	Propriétaire – Payant selon usage
Types de profils	Voiture, vélo, piéton, fauteuil roulant, transport adapté	Voiture, vélo, piéton, transport public
Analyse spatiale	Oui – matrices d'accessibilité, géocodage direct/inverse	Limité – géocodage séparé, pas d'analyse spatiale
Personnalisation	Très flexible, paramètres détaillés, formats GeoJSON	Moins flexible, formats et options prédéfinis
Quota gratuit	2 000 requêtes/jour (clé gratuite)	500 requêtes/jour (clé gratuite)
hébergement	Hébergement européen – conformité RGPD	Hébergement Google – hors UE
Intégration avec Leaflet	Native via plugins et formats compatibles	Possible mais nécessite adaptation
Documentation	Complète, open source	Très complète, développeurs commerciau

7. Solutions pour la Carte et l'Itinéraire

Utilisation de Leaflet.js carte open source web

openroute service

Openrouteservice est une API permettant de calculer des itinéraires

Contraintes techniques

Nécessite un fournisseur de tuiles compatible (ex. OpenStreetMap).

Optimisé pour web et mobile, limité en fonctionnalités natives 3D.

Usage dans le projet

Base cartographique interactive.

Interface utilisateur pour affichage des itinéraires générés par l'API de routage.

Fonctionnalités principales

Chargement et affichage de tuiles cartographiques (OSM, cartes personnalisées).

Gestion des couches (layers), marqueurs, polygones, popups.

Contrôles interactifs (zoom, géolocalisation, échelle).

Extensible via plugins (heatmaps, clusters, dessin).

openroute service

Contraintes techniques

Requiert clé d'API (quota selon plan choisi).

Limitation du nombre de requêtes par minute/jour.

Temps de réponse dépendant de la complexité de la requête.

Usage dans le projet

Fourniture des données de navigation.

Génération d'itinéraires intégrés dans Leaflet.js.

Fonctionnalités principales

Calcul d'itinéraires multi-profils (voiture, piéton, vélo, transports adaptés).

Isochrones (zones atteignables en X minutes ou kilomètres).

Géocodage direct et inverse.

Analyse spatiale (accessibilité, matrices d'itinéraires).

8. Gestion des Calendrier

FullCalendrier.js

Affichage des livraisons et événements.

Création et modification d'évènements.

Synchronisation bidirectionnelle avec le backend..

Notifications automatiques.

Usage dans le projet

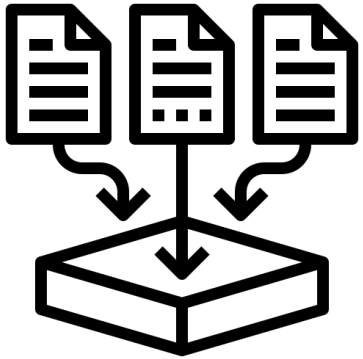
Intégré dans Twig

API REST pour lecture/écriture des événements

Accès restreint par rôles

9. Protection des données et RGPD

Collecte et conservation :

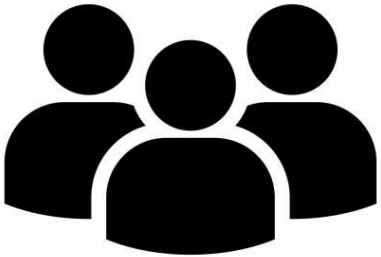


Données limitées au strict nécessaire (adhérents, abonnements, paiements).

Durée de conservation définie selon obligations légales.

Suppression ou anonymisation automatique après expiration.

Droits des utilisateurs :



Accès, rectification, suppression et portabilité des données.

Interface utilisateur pour gérer les consentements.

Procédure documentée pour exercer ses droits.

Transparence :

Politique de confidentialité claire et accessible.

Information explicite lors de la collecte des données (finalité, usage).

Traçabilité et conformité :

Journalisation des opérations sur les données personnelles.

Registre interne des traitements.

Évaluation d’impact (DPIA) si traitement sensible.

Responsabilité organisationnelle :

Référent RGPD désigné.

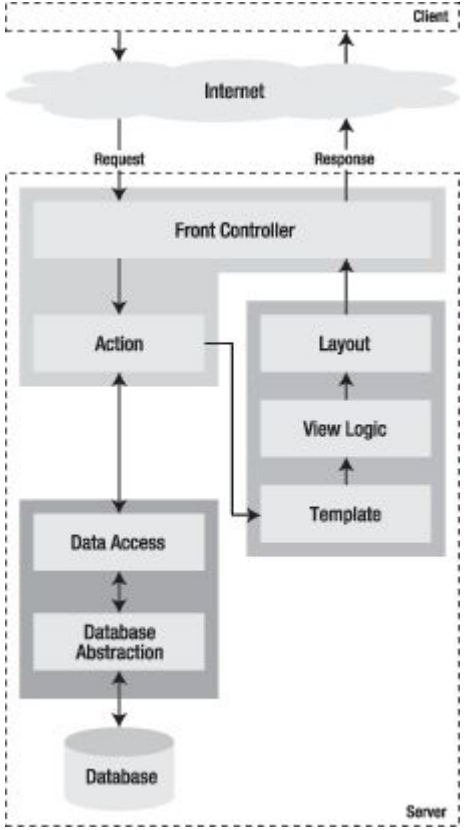
Processus de gestion des incidents et violations de données.

10. Qualité de code et tests

Normes et bonnes pratiques :

- Respect des standards PSR pour PHP/Symfony.
- Revue de code systématique avant intégration.
- Documentation du code et des API.

11. Schémas et illustration



Subscription et Livraison

Client → Frontoffice → API Symfony → Base de données
└──> Crée/modifie abonnement
└──> Sélectionne point de dépôt
└──> Choisit fréquence et produit

API Symfony → Génère planning de livraison
→ Enregistrer dans DB
→ Envoie confirmation au client

Jour de livraison:
Backoffice → API → Vérifie statut
→ Génère feuille de route
→ Affiche itinéraire (ORS + Leaflet)

12. Conclusion

Ce document définit les spécifications techniques pour le développement dúne nouvelle plateforme de gestion commerciale adaptée aux besoins actuels des Jardins de Cocagne, La solution proposée repose sur les technologies modernes, garantit la sécurité des données, facilite l’utilisation des outils externes et encourage une gestion agile et collaborative. Sa mise en œuvre permettra d’améliorer l’efficacité et l'expérience des utilisateurs.