suivi et gestion de ruchers

Plateforme de gestion apicole moderne

Timour S. Dorian J. Corentin G. Melissa A.O.

Février 2026

Ynov Campus

Sommaire

1. Contexte et objectifs
2. L'équipe
3. Étapes de développement
4. Technologies
5. Architecture
6. Fonctionnalités clés
7. Problèmes et solutions
8. Choix techniques
9. Tests et déploiement
10. Bilan et perspectives

Contexte et objectifs

L'apiculture manque d'outils numériques

Problème

- Carnets papier, perte d'historique
- Aucune traçabilité sanitaire
- Pas de vue d'ensemble sur le rucher
- Risque de vol sans surveillance

Notre solution

- Plateforme web moderne
- Ruchers, ruches, reines, élevage
- IoT et anti-vol GPS

Personas

Jean, 42 ans — Pro

200+ ruches, élevage + production, terrain, GPS, IoT

Marc, 55 ans — Amateur

15 ruches, production de miel, interface simple, rappels saisonniers

L’équipe

Une équipe de 4 profils complémentaires

Timour S.

Chef de projet

Coordination

Dorian J.

Frontend

Next.js, Apollo

Corentin G.

Backend

Django, Hasura

Melissa A.O.

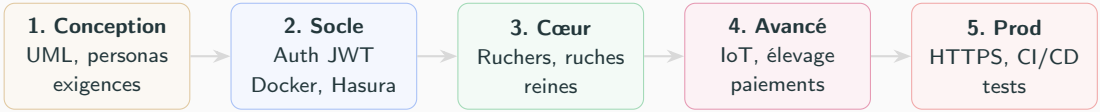
DevOps

Docker, Traefik

GitHub Discord Réunions hebdo

Étapes de développement

5 phases, de la conception à la production



Gestion de projet et CI/CD

GitHub Actions

- Pipeline CI automatisé sur chaque push
- Lint + tests backend (pytest)
- Lint + tests frontend (Vitest)
- Analyse SonarCloud intégrée
- Build Docker validé avant merge

Organisation

- Branches : `main` / `develop` / features
- Pull requests + code review
- Issues GitHub pour le suivi
- Réunions hebdomadaires
- Discord pour la communication

Technologies

Une stack moderne et modulaire

Backend

- Django 5 (Python)
- PostgreSQL 15
- Hasura GraphQL v2.36
- Gunicorn
- Traccar (GPS)

Frontend

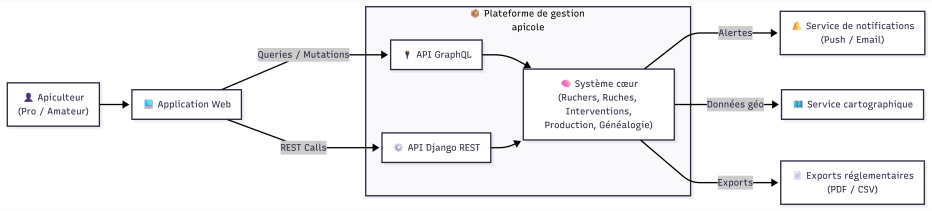
- Next.js 14 (App Router)
- TypeScript
- Apollo Client
- Tailwind CSS
- shadcn/ui

DevOps

- Docker Compose
- Traefik + Let's Encrypt
- Stripe (paiements)
- GitHub Actions
- SonarCloud

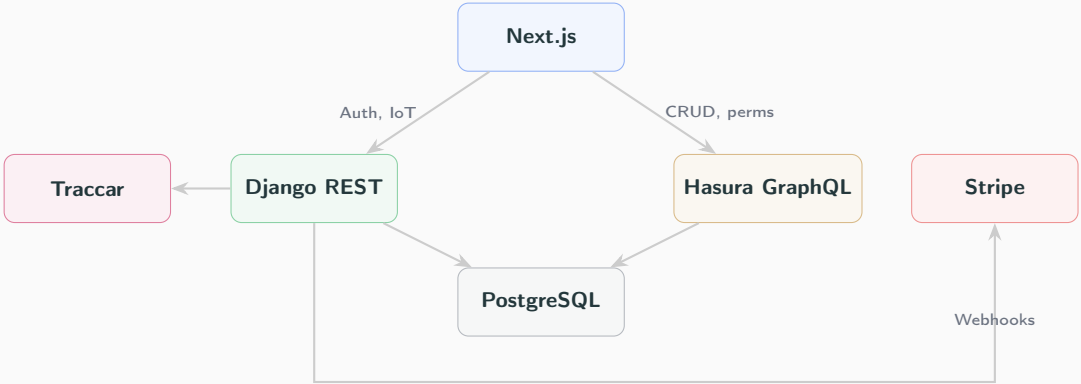
Architecture

Vue d'ensemble du système



Acteurs externes, capteurs IoT et services connectés à la plateforme.

Django gère la logique, Hasura le CRUD



Django = logique métier | Hasura = CRUD auto + permissions | PostgreSQL = source unique

Fonctionnalités clés

Gestion complète des ruchers et ruches

Ruchers & Ruches

- CRUD complet via GraphQL
- Carte GPS interactive (Leaflet)
- Immatriculation unique (`X1234567`)
- 6 types, 5 races, 11 maladies
- Création en masse
- Transhumance GPS

Interventions

- 7 types (Visite, Traitement, Récolte...)
- Intervention groupée multi-ruches
- Notifications automatiques :
  - Rappel visite (30j)
  - Rappel traitement
  - Calendrier saisonnier
  - Alerte sanitaire

Élevage de reines : du greffage à la vente

Gestion des reines

- Code couleur international
- Lignée génétique
- Note de douceur (1–10)
- 6 statuts (Fécondée → Vendue)
- Profil Éleveur dédié

Module élevage

- Racles (cadre + cupules)
- Pré-remplissage auto des reines
- Cycle de 7 tâches automatiques :
  - J0 Greffage → J28 Mise en vente
- Verrouillage séquentiel
- Kanban de suivi

IoT, anti-vol GPS et modèle économique

IoT & Anti-vol

- 7 types de capteurs
- Intégration Traccar (open-source)
- Alerte GPS anti-vol :
  - Position de référence
  - Seuil configurable (100m)
  - Calcul Haversine
  - Email d'alerte automatique

Multi-tenant & Stripe

- Multi-entreprise (N :N)
- Invitations par email
- 3 rôles (permissions Hasura)

Freemium / Premium

Free 3 ruchers, 1 capteur

Premium Illimité

Webhooks Stripe pour la facturation

Problèmes et solutions

4 problèmes majeurs résolus en cours de route

	Problème	Solution
	Metadata Hasura perdu en prod	Volume Docker persistant
	Suppression racle : erreur FK	Mutation cascade en 4 étapes
	Secrets dans docker-compose	Fichiers `.env`
	Spam d'alertes GPS	Rate-limiting (`gpsLastAlertAt`)

Choix techniques

Ce qui a marché et ce qui reste difficile

Succès

- **Hasura** : CRUD auto + permissions
  $\Rightarrow$ gain de temps majeur
- **shadcn/ui** + **Tailwind** : UI rapide
- **Docker** : env reproductible
- **Traefik** : HTTPS automatique
- **Traccar** : GPS clé en main
- **TypeScript** : bugs détectés tôt

Difficultés

- **Django** + **Hasura** :
  double source de permissions
- **Metadata Hasura** :
  drift fréquent
- **Logique métier** :
  reste côté Django

Tests et déploiement

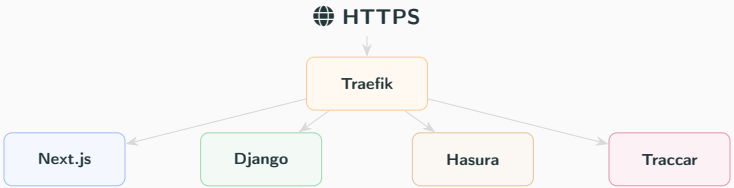
Qualité du code et pipeline CI/CD

Tests & Qualité

- **Backend** : pytest (unitaire + intégration)
- **Frontend** : Vitest + Testing Library
- SonarCloud : analyse continue
- GitHub Actions : CI/CD
- Swagger UI : doc API

Production

- 6 conteneurs Docker
- Traefik reverse proxy
- HTTPS Let's Encrypt
- DuckDNS (DNS dynamique)
- Volumes persistants (DB, metadata)
- Gunicorn (3 workers)



Démo en direct

Bilan et perspectives

79% des exigences implémentées, 11 bonus

Module	✔	∼	✘
Gestion générale	6	0	0
Suivi sanitaire	4	1	2
Traçabilité	5	0	2
Production	1	3	1
Actions groupées	3	0	0
Élevage reines	7	3	3
IoT et sécurité	5	2	0
Total / 51	**31**	**9**	**11**

79% implémenté

11 bonus hors cahier

- Multi-entreprise
- Invitations par email
- Freemium / Premium (Stripe)
- Notifications automatiques
- Calendrier saisonnier
- Vérification email
- Reset mot de passe
- Création en masse
- Infra production Docker/HTTPS
- SonarCloud + tests
- Documentation API Swagger

Roadmap : 7 évolutions priorisées

Prio.	Fonctionnalité	Effort
P1	Quarantaine virtuelle (ruches malades)	Faible
P2	Dashboard de production (analytics)	Moyen
P3	Historique de ponte des reines	Moyen
P4	Workflow de remérage	Moyen
P5	Tableau de bord Élite (classement)	Élevé
P6	Généalogie des essaims	Élevé
P7	Application mobile offline-first	Élevé

MVP Livré

39	15+	30+	79%	11
tables	endpoints	opérations	exigences	bonus

Base solide, multi-tenant, prête pour la V2.

❤ Merci pour votre attention !

Des questions ?