# Mon Chai — Dev Book (v1.0)
**Période couverte : 8 septembre → 31 octobre 2025 (Europe/Paris)**
**Auteur : Denis Berthelot (solo-dev)**
**Objectif : MVP production → ventes → DRM (région pilote), premiers retours en octobre**

---

## 0. Mode d’emploi (IA & Reprise rapide)
- **But de ce document** : permettre à un développeur ou une IA de reprendre le développement *sa
entre deux sessions.
- **Structure** : chaque section suit le format **Objectif → Spécs → Étapes → API/Modèles → UI
Acceptation → Dépendances → Livrables**.
- **Règle d’or** : à la fin de chaque sous-étape, **committer** avec un message **Conventional Co
et **mettre à jour** la checklist locale :
  - Exemple : `feat(api): add POST /vendanges with validation & tests (#45)`
- **Reprise express** (template) :
  1) *Dernier commit* : `<hash>` — `<message>`
  2) *Étape en cours* : `<nom>` → **bloquée ?** oui/non (si oui, préciser)
  3) *Prochaine micro■tâche* : `<action concrète en 1 phrase>`
  4) *Tests à (re)faire* : `<pytest -k ...>`
  5) *Notes rapides* : `<risques, TODO, décisions en suspens>`

> **Convention** : toutes les commandes shell sont préfixées par `$`. Les chemins sont relatifs à
repo.

---

## 1. Vision, portée, jalons
### 1.1 Vision
Créer le **SaaS viticole** le plus simple et fiable pour **suivre le flux chai** : *vendanges → c
assemblages/soutirages → mise en bouteille → ventes → exports DRM*.
- **Différenciant** : ergonomie, **mouvements auto (débits/crédits)**, **verrouillage par événeme
exports **DRM** prêts à déposer (région pilote).
- **Public** : domaines indépendants (1–20 ha), caves particulières, petites coopératives.

### 1.2 Portée MVP (tech & produit)
- **Back** : Django 5 + Django REST Framework (DRF), PostgreSQL 15.
- **Front** : Django templates + **HTMX** + TailwindCSS (MVP rapide), *API-first* pour réutilisat
(Next.js/React si besoin).
- **Fonctionnel** :
  - Parcelles, Vendanges, Cuves/Lots, Mouvements (double■entrée), Mise en bouteille, Stock.
  - Clients, Commandes, **Factures PDF**.
  - **Export DRM** : CSV/PDF pour **région pilote** (Loire — MVP).

### 1.3 Jalons datés (résumé)
- **J1** (9 sept 18:00) : Archi & schéma validés.
- **J2** (12 sept 18:00) : CRUD + règles **Mouvements** OK.
- **J3** (19 sept 18:00) : Chaîne vendanges→cuve→bouteille **opérationnelle**.
- **J4** (26 sept 18:00) : Stock & ventes + **facture PDF**.
- **J5** (1 oct 18:00) : **Export DRM** (Loire) v1.
- **J6** (3 oct 18:00) : Landing + waitlist + logo.
- **J7** (10 oct 18:00) : Démo vidéo + 1er outreach (≥5 domaines).
- **J8** (15 oct 18:00) : Pack juridique (ML/CGU/CGV/RGPD) v1.
- **J9** (24 oct 18:00) : Retours intégrés (v1.1).
- **J10** (31 oct 16:00) : Go/No■Go pilote payant.

---

## 2. Stack & conventions
### 2.1 Outils & versions
- **Python 3.12**, **Django 5.x**, **DRF 3.16+**, **PostgreSQL 15** (+ **PostGIS** optionnel pour
parcelles).
- **TailwindCSS 3**, **HTMX 1.9+**, **Alpine.js (optionnel)**.
- **Docker** & **docker-compose**, **Poetry** (ou pip-tools), **pre-commit**, **pytest**,

**black/flake8/isort**, **eslint/prettier** (si JS séparé).
- **Sentry** (backend), **OpenAPI/Swagger** auto pour l'API.

### 2.2 Structure repo (proposée)
```
monchai/
  backend/
    manage.py
    monchai/                # settings, urls
    apps/
      accounts/
      core/                 # chai: parcelles, vendanges, cuves, mouvements
      sales/                # clients, commandes, factures
      drm/                  # exports réglementaires
    templates/
    static/
    requirements/           # or pyproject.toml if Poetry
    tests/
  infra/
    docker/
      Dockerfile.api
      Dockerfile.worker
      docker-compose.yml
    k8s/ (plus tard)
  docs/
    DEV_BOOK.md
    API_SPEC.yaml
    ERD.png
  .github/workflows/
  Makefile
```

### 2.3 Environnement & secrets
- `.env` (local) / variables env (staging/prod) :
  - `DJANGO_SECRET_KEY`, `DATABASE_URL`, `ALLOWED_HOSTS`, `SENTRY_DSN`, `STRIPE_API_KEY` (plus ta

### 2.4 Conventions
- **Conventional Commits** : `feat:`, `fix:`, `docs:`, `refactor:`, `test:`, `chore:` …
- **Branching** : `main` (prod), `develop` (staging), `feature/*` (tâche), `hotfix/*` (prod).
- **Migrations** : nom explicite, ex. `0003_add_mouvement_constraints.py`.

---

## 3. Modèle de données (ERD logique)
> **Principe clé** : tout mouvement de volume passe par un **journal de mouvements** avec **débit
**verrouillage** par événement validé.

### 3.1 Entités principales
- **Domaine** (1) — (N) **Parcelle**
- **Parcelle** (1) — (N) **Vendange** (date, volume brut, % perte possible)
- **Cuve** (1) — (N) **Lot** (lot = volume contenu dans une cuve à un instant T)
- **Mouvement** : journal (source → destination, volume, type, date, statut : draft/validé/verrou
  - types : `vendange_vers_cuve`, `inter_cuves`, `perte`, `mise_en_bouteille`, `vente_sortie_stoc
- **BouteilleLot** : résultat d'une **mise en bouteille** (lot_bouteilles, nb_bouteilles, contena
- **Produit** : mappage commercial d'un **BouteilleLot** (SKU, prix)
- **Stock** : **vue matérialisée** (ou requête) calculée à partir des mouvements validés
- **Client**, **Commande**, **LigneCommande**, **Facture**, **Paiement**
- **DRMExport** : artefacts (CSV/PDF) + log

### 3.2 Contraintes & règles
- Pas de **volume négatif**.
- **Conservation des volumes** : somme débits = somme crédits + pertes.
- **Verrouillage** : un mouvement validé lié à un événement **verrouille** sa modification (audit
- Un **Produit** ne peut référencer que des **lots bouteilles** existants et non épuisés.

- La suppression d’une **Parcelle** est interdite si **Vendanges** associées existent.

---

## 4. API (DRF) — contrats & exemples
> **Auth** par JWT (simplejwt) ou session (MVP : session + CSRF; si clients externes → JWT).

### 4.1 Auth & domaines
- `POST /api/auth/login` — body `{email, password}` → `200 OK` + session/JWT.
- `POST /api/auth/signup` — crée un domaine si premier user.
- `GET /api/me` — infos user + rôles (`admin_domaine`, `oenologue`, `operateur_chai`, `compta`).

### 4.2 Parcelles
- `POST /api/parcelles/`
  ```json
  { "nom": "Les Carrières", "cepage": "Melon B.", "surface_ha": 1.2, "lat":47.1, "lng":-1.5,
"annee_plantation": 2008 }
  ```
- `GET /api/parcelles/?search=Carri` — filtre simple.
- **Validation** : `surface_ha > 0`, coordonnées optionnelles.

### 4.3 Vendanges
- `POST /api/vendanges/`
  ```json
  { "parcelle_id": 12, "date": "2025-09-12", "volume_hl": 25.4, "dechets_hl": 0.8 }
  ```
- **Effet** : génère un **Mouvement** `vendange_vers_cuve` **draft** si cuve cible renseignée, si
planifier.

### 4.4 Cuves & Lots
- `POST /api/cuves/` — crée une cuve (capacité, matériau, nom interne).
- `POST /api/lots/` — crée un lot initial (rare; en général via mouvement).

### 4.5 Mouvements
- `POST /api/mouvements/`
  ```json
  {
    "type": "inter_cuves",
    "source_lot_id": 34,
    "destination_cuve_id": 7,
    "volume_hl": 5.0,
    "date": "2025-09-18",
    "commentaire": "Soutirage clair"
  }
  ```
- **Règles** :
  - `volume_hl > 0`
  - `volume_hl <= source_lot.volume_disponible_hl`
  - Si `type = mise_en_bouteille` → créer **BouteilleLot** (voir 4.6).

- `POST /api/mouvements/{id}/valider` → statut `validé` (+ verrou si lié à événement).
- `DELETE /api/mouvements/{id}` → autorisé seulement en `draft`.

### 4.6 Mise en bouteille
- `POST /api/mise_en_bouteille/`
  ```json
  {
    "source_lot_id": 34,
    "nb_bouteilles": 2500,
    "contenance_ml": 750,
    "taux_perte_hl": 0.2,
    "date": "2025-09-19"
  }
  ```

- **Effet** : mouvement `mise_en_bouteille` (débit lot → crédit stock bouteille), crée **Bouteil

### 4.7 Produits & Stock
- `POST /api/produits/`
  ```json
  {
    "nom": "Muscadet S&M 2024 - Les Carrières",
    "bouteille_lot_id": 4,
    "sku": "MC24-CAR-750",
    "prix_ttc_eur": 8.90,
    "tva_pct": 20
  }
  ```
- `GET /api/stock/produits/` — agrégé depuis mouvements validés.

### 4.8 Ventes & Factures
- `POST /api/clients/`
- `POST /api/commandes/`
  ```json
  {
    "client_id": 9,
    "lignes": [
      {"produit_id": 3, "quantite": 60, "prix_unitaire_ttc_eur": 8.50}
    ],
    "date": "2025-09-26"
  }
  ```
- **Effet** : mouvement `vente_sortie_stock` (débit stock bouteille).
- `POST /api/factures/{commande_id}/pdf` → génère PDF (numérotation légale).

### 4.9 DRM Export (Loire — MVP)
- `GET /api/drm/export?mois=2025-10` → génère **CSV** + **PDF** (artefacts stockés et historisés)
- **Champs** (brouillon) : *période, appellation, volume entrées/sorties, pertes, stocks début/fi
lots*.

---

## 5. UI (HTMX + Tailwind) — parcours clés
### 5.1 Dashboard (post■login)
- Cartes : **Stock bouteilles**, **Ventes du mois**, **Alerte faible stock**, **Tâches en attente
- Boutons rapides : *Ajouter vendange*, *Créer mouvement*, *Mise en bouteille*, *Nouvelle command

### 5.2 Parcelles & Vendanges
- **Parcelles** : liste + carte (option PostGIS) / CRUD minimal.
- **Vendanges (wizard)** : sélectionner **Parcelle** → saisir **date + volume** → (option) **aff
immédiate.

### 5.3 Cuves/Lots & Mouvements
- **Vue cuverie** : table des **Lots** avec **volume disponible**, actions : `Soutirage`, `Assemb
`Perte`, `Mise en bouteille`.
- **Historique mouvements** : filtrable par période, export CSV.

### 5.4 Mise en bouteille & Produits
- **Formulaire** : nb bouteilles, contenance, pertes, étiquetage (info).
- **Création produit** à partir d’un **BouteilleLot**.
- **Stock produits** : *sku, nom, dispo, réservés, vendus*.

### 5.5 Ventes & Facturation
- **Commande** : client → lignes → total → **Générer facture PDF** (numéro lncrmt, mentions obl
- **Liste factures** : statut (*brouillon, émise, payée*), export.

### 5.6 DRM
- Écran **Export DRM** : choix période, région = *Loire (MVP)*, téléchargement CSV/PDF, trace d’a

---

## 6. Sécurité & RBAC (rôles)
- **Admin domaine** : tout (paramètres, utilisateurs).
- **Œnologue** : données chai + DRM export.
- **Opérateur chai** : vendanges, mouvements, mise en bouteille (pas de suppression validés).
- **Compta** : clients, commandes, factures, exports.
- **Règles** :
  - Modification/suppression impossible sur **mouvements validés** (audit).
  - Accès cloisonné par **domaine** (tenant simple via `domain_id` sur toutes les entités).

---

## 7. Tests (unitaires & e2e minimal)
### 7.1 Outils
- `pytest`, `pytest-django`, `factory_boy`, `model_bakery`, `httpx` (si tests API), `pytest-cov`.

### 7.2 Stratégie
- **Unitaires** : modèles (contraintes), services (règles volumes), API (status codes, payload).
- **Intégration** : parcours chai complet (vendange→cuve→bouteille), ventes/facture, DRM export.

### 7.3 Exemples de tests (pseudo■pytest)
```python
def test_mouvement_inter_cuves_ne_peut_pas_depasser_volume_source(db):
    lot_src = make_lot(volume_hl=10)
    cuve_dst = make_cuve()
    with pytest.raises(ValidationError):
        mouvements.create_inter_cuves(lot_src.id, cuve_dst.id, volume_hl=11)

def test_mise_en_bouteille_cree_bouteillelot_et_met_a_jour_stock(db):
    lot_src = make_lot(volume_hl=15)
    bl = mise_en_bouteille.exec(lot_src.id, nb_bouteilles=2000, contenance_ml=750, pertes_hl=0.2)
    assert bl.nb_bouteilles == 2000
    assert stock.produit_disponible(bl.produit_id) >= 2000
```

---

## 8. CI/CD, déploiement & monitoring
### 8.1 CI (GitHub Actions minimal)
- Jobs : `lint`, `tests`, `build`, `deploy-staging`.
- Cache pip/poetry, `pytest --maxfail=1 --disable-warnings`.
- Artefacts : **coverage.xml**, **API_SPEC.yaml**.

### 8.2 Déploiement
- Cibles : **Railway** ou **Render** (back + Postgres).
- **Commandes** :
  - Migrations : `$ python manage.py migrate`
  - Collect static : `$ python manage.py collectstatic --noinput`
  - Créer superuser (staging) : `$ python manage.py createsuperuser`

### 8.3 Monitoring
- **Sentry** (Django integration).
- Logs **structurés JSON** (niveau `INFO` en prod).
- Ping healthcheck : `/healthz` (renvoie `{status:"ok", db:true, version:"..."}`).

---

## 9. Juridique & conformité (MVP)
- **Mentions légales**, **CGU/CGV**, **Politique RGPD** (base légale, DPA, sous■traitants, droits
- **Facturation** : numérotation, mentions obligatoires (SIRET, TVA, adresse, conditions de paiem
- **DRM (Loire – MVP)** : confirmer champs obligatoires, mapping interne → export CSV/PDF.
- **Cookies/analytics** : bannière simple, mesure d’audience non■invasive (MVP).

---

## 10. Chronologie détaillée (avec micro■étapes)

### Semaine du **8 sept**
#### [J1] Archi & schéma données (deadline 9 sept 18:00)
**Objectif** : poser le repo, les apps Django, le schéma initial, les conventions.
**Étapes** :
1. Repo + README + licence.
2. Django project `monchai`, apps : `accounts`, `core`, `sales`, `drm`.
3. Paramètres `settings.py` (split base/dev/prod), `DATABASE_URL`.
4. Modèles squelette (vides) + migrations initiales.
5. Fichier `docs/ERD.md` + **ERD (texte)** + TODO image.
6. **Pre-commit** (`black`, `isort`, `flake8`) + **pytest** config.
7. CI lint/tests (fail fast).
**Acceptation** : `pytest` vert (0 tests OK), `manage.py migrate` OK, CI passe.

#### [J2] Mouvements de cave (CRUD + règles) (deadline 12 sept 18:00)
**Objectif** : implémenter **Mouvement** + services règles volumes.
**Étapes** :
1. Modèles : `Cuve`, `Lot`, `Mouvement` (types, status, audit).
2. Services : `mouvements.create_*`, `valider`, `annuler` (draft only).
3. Validations : volumes, conservation, verrous.
4. API DRF : `POST/GET/PATCH/DELETE /api/mouvements/`, `POST /valider`.
5. Tests unitaires & d'intégration.
**Acceptation** : cas `inter_cuves`, `perte`, `vendange_vers_cuve` validés; impossible de modifie
mouvement validé.

### Semaine du **15 sept**
#### [J3] Chaîne vendanges → cuve → bouteille (deadline 19 sept 18:00)
**Objectif** : **parcours chai complet** jusqu'à création de **BouteilleLot**.
**Étapes** :
1. Vendanges : modèle + API + wizard HTMX.
2. Mise en bouteille : service + API + UI (form).
3. Mise à jour stock via mouvements validés.
4. Dashboard : cards volumes clés.
**Acceptation** : scénario e2e passe : *créer vendange → mouvement vers cuve → mise en bouteille
bouteilles > 0*.

### Semaine du **22 sept**
#### [J4] Stock & ventes + facture PDF (deadline 26 sept 18:00)
**Objectif** : **ventes** reliées au **stock** + génération **facture PDF**.
**Étapes** :
1. Modèles `Client`, `Commande`, `LigneCommande`, `Facture`.
2. Mouvement `vente_sortie_stock` sur validation commande.
3. Génération PDF (numéro, mentions légales).
4. Listes & filtres UI.
**Acceptation** : créer une commande diminue le stock; facture PDF téléchargeable et conforme.

### Semaine du **29 sept**
#### [J5] Export DRM Loire v1 (deadline 1 oct 18:00)
**Objectif** : export **CSV + PDF** conforme *brouillon* pour Loire.
**Étapes** :
1. Mapper champs internes → DRM (période, entrées, sorties, pertes, stock début/fin).
2. Génération CSV (délimiteur `;`) + PDF simple.
3. Historiser artefacts (table `DRMExport` + fichier).
**Acceptation** : fichier ouvert par tableur; PDF lisible; totaux cohérents avec journaux mouveme

#### [J6] Landing + Waitlist + Logo (deadline 3 oct 18:00)
**Objectif** : capter des leads.
**Étapes** :
1. Logo SVG simple, palette bordeaux/or.
2. Landing (promesse, capture email, privacy).
3. Stockage leads (Mailinglist table ou service tiers).

**Acceptation** : formulaire enregistre une adresse; page responsive.

### Semaine du **6 oct**
#### [J7] Démo vidéo & Outreach (deadline 10 oct 18:00)
**Objectif** : produire une **démo 2–3 min** + contacter ≥5 domaines.
**Étapes** : script, capture, montage, upload, emails ciblés.
**Acceptation** : 5 envois prouvés; lien vidéo OK.

### Semaine du **13 oct**
#### [J8] Pack juridique v1 (deadline 15 oct 18:00)
**Objectif** : ML, CGU/CGV, RGPD (bannière cookies).
**Acceptation** : pages visibles, textes cohérents, cookies non-invasifs par défaut.

### Semaine du **20 oct**
#### [J9] Retours intégrés (v1.1) (deadline 24 oct 18:00)
**Objectif** : intégrer 5 retours utilisateurs structurés.
**Acceptation** : changelog publié, issues fermées.

#### [J10] Go/No■Go pilote payant (31 oct 16:00)
**Critères** : 1 domaine utilise la chaîne complète en staging; ≥20 leads; DRM export OK.

---

## 11. Spécifications détaillées (par module)

### 11.1 Accounts (utilisateurs & rôles)
**Objectif** : gérer les utilisateurs d'un **domaine** (tenant simple).
**Modèles** : `User` (email unique), `Profile` (rôle), `Domaine`.
**API** : signup/login, inviter utilisateur au domaine, changer rôle.
**UI** : gestion équipe (admin).
**Tests** : accès limité par domaine; un opérateur ne voit pas les ventes si pas autorisé.

### 11.2 Core (chai)
**Objectif** : modéliser le chai et les flux de volumes.
**Modèles** :
- `Parcelle(id, domaine_id, nom, cepage, surface_ha, lat, lng, annee_plantation, created_at)`
- `Cuve(id, domaine_id, nom, capacite_hl, materiau, created_at)`
- `Lot(id, domaine_id, cuve_id, volume_disponible_hl, ref_interne, created_at)`
- `Mouvement(id, domaine_id, type, source_lot_id?, destination_cuve_id?, volume_hl, pertes_hl, da
verrouille, meta_json, created_at)`
- `BouteilleLot(id, domaine_id, source_lot_id, nb_bouteilles, contenance_ml, date, ref_interne)`

**Services** : `vendanges.create()`, `mouvements.create_inter_cuves()`, `mouvements.valider()`,
`mise_en_bouteille.exec()`
**Règles** : validations volumes, conservation, verrou.
**Tests** : voir §7.3.

### 11.3 Sales (ventes & factures)
**Objectif** : lier ventes au stock et générer les factures.
**Modèles** : `Client`, `Commande(status)`, `LigneCommande`, `Facture(numero, pdf_path, total_ttc
date_emission)`
**Services** : `commandes.valider()` (déclenche mouvement `vente_sortie_stock`)
**PDF** : gabarit simple, logo, mentions légales, numérotation séquentielle par domaine/année.

### 11.4 DRM
**Objectif** : produire les exports mensuels.
**Entrées** : agrégations sur **Mouvement** (validés) + **BouteilleLot**.
**Sorties** : CSV (séparateur `;`) + PDF récap.
**Logs** : `DRMExport(id, domaine_id, periode, chemin_csv, chemin_pdf, checksums, created_at)`.

---

## 12. Qualité & sécurité
- **Linters** (pre-commit): black, isort, flake8.

- **Tests** : seuil de couverture 70% (MVP), montez ensuite.
- **Sécurité** : headers HTTPS, `SECURE_*` en prod, CSRF pour session.
- **Permissions** : DRF `IsAuthenticated & DomainePermission & RolePermission`.

---

## 13. Données de démo & fixtures
- Script `manage.py seed_demo` : crée un domaine, 2 parcelles, 1 cuve, 1 vendange, 1 mise en bout
produit, 1 client, 1 commande.
- Permet de tourner la **démo vidéo** rapidement.

---

## 14. Onboarding guidé (style jeu vidéo)
- **Intro.js** (ou équivalent) : surbrillance d'un seul élément à la fois + pop■ups.
- Étapes : `Créer vendange` → `Créer mouvement` → `Mise en bouteille` → `Créer produit` → `Crée
`Facture PDF` → `Export DRM`.
- **Skippable**, sauvegarde de progression (localStorage).

---

## 15. Marketing & métriques (MVP)
- Landing : promesse claire, capture email, preuve sociale (à venir).
- KPI octobre : 1 domaine pilote actif; ≥20 leads; 5 retours qualifiés; temps "vendanges→cuve" <
"vente→facture" < 60s.

---

## 16. Annexes
### 16.1 Exemple de **Conventional Commits**
- `feat(core): add Mouvement model with validation rules`
- `feat(api): POST /mouvements + e2e inter-cuves`
- `fix(stock): prevent negative volume on bottle lot creation`
- `chore(ci): add pytest coverage to pipeline`

### 16.2 Makefile (extrait)
```
setup:
	poetry install || pip install -r requirements/dev.txt
	npm install

run:
	python backend/manage.py runserver

test:
	pytest -q

lint:
	black . && isort . && flake8
```

### 16.3 Variables d'environnement (exemple)
```
DJANGO_SECRET_KEY=change_me
DATABASE_URL=postgres://user:pass@localhost:5432/monchai
ALLOWED_HOSTS=localhost,127.0.0.1
SENTRY_DSN=
STRIPE_API_KEY=
```

---

**Fin — v1.0 (8 septembre 2025)**
*Mon Chai — du chai à la vente, sans friction.*