Cahier des charges – Modèle générique (Module <Nom_du_module>)

Glossaire

- **MVP** : première version utilisable, minimale.
- **RBAC** : gestion de droits par rôle (Admin/Éditeur/Lecteur).
- **i18n** : internationalisation (activer plusieurs langues).
- **Staging** : environnement de test en ligne (pré-prod).
- **SLA** : objectif de disponibilité.
- **p95** : 95% des requêtes sont plus rapides que cette valeur.
- **RAG** : recherche + extraction de passages pour répondre avec sources.
- **Tenant** : entreprise cliente isolée (séparation des données).
- **Proxy** : passerelle qui relaie les requêtes (ici Next → FastAPI).

0) Semaine 0 / Definition of Ready (prérequis)

- **Environnements installés** : Node (LTS), pnpm, Python 3.10+, Docker, Git, Postgres (ou Supabase).
- **Accès & secrets** : dépôt Git, URL DB, bucket de stockage, clés `.env` (voir §17).
- **Hello world technique** :
    - Lancer **Next.js** (socle MakerKit) → afficher `/home`.
    - Lancer **FastAPI** → `GET /health` retourne `{ ok: true }`.
    - **Proxy** Next → FastAPI : `POST /api/<module>/run` renvoie un JSON mock.

- **Jeux de démo** : 2 petits fichiers CSV/Excel réalistes (10–50 lignes) + un export d'exemple.

- **Definition of Done (S0)** : front, back et proxy **tournent en local** ; un **bouton appelle l'API** et affiche la réponse.

1) Contexte & objectifs

Contexte.

- Entreprise : `<PME/ETI/Division>` • Domaine : `<industrie/services>` • Environnement : `<SaaS / on-prem>`.

Objectifs v1.0.

- En **3 écrans maximum**, fournir :

    1. **Vue synthèse** (KPI/alertes).

    2. **Vue détail** (analyse/interaction).

    3. **Import/Export** (données & rapports).

- Méthodes simples, **explicables** et **actionnables**.

2) Périmètre (MoSCoW)

- **Must (v1)** : import CSV/Excel, 3 écrans, calculs/alertes de base, exports Excel/PDF, multi-tenant simple.

- **Should (v1.1)** : plan d'actions (responsable, échéance), favoris, filtres avancés.

- **Could (v2)** : connecteurs externes (ERP/Drive/API), notifications temps réel, modèles avancés.

- **Won't (v1)** : fonctionnalités non indispensables / lourdes.

Out-of-scope (v1)

- Paiements/Stripe, connecteurs ERP lourds, temps réel WebSocket généralisé, édition riche WYSIWYG, IA avancée non nécessaire au MVP.

Hypothèses & dépendances

- Données fournies au format CSV/Excel ; un **référent métier** disponible 1h/semaine pour valider les règles.
- Les seuils/poids de calcul sont **configurables** (table `settings` ou `.env`).

3) Personae & rôles

- **Personae** : `<Opérateur / Analyste / Manager / Direction>`.
- **Rôles & droits (RBAC)** :
  - **Admin (tenant)** : tout.
  - **Éditeur** : import/édition, exports.
  - **Lecteur** : lecture, filtres, exports.
  - **Invité (optionnel)** : lecture restreinte.

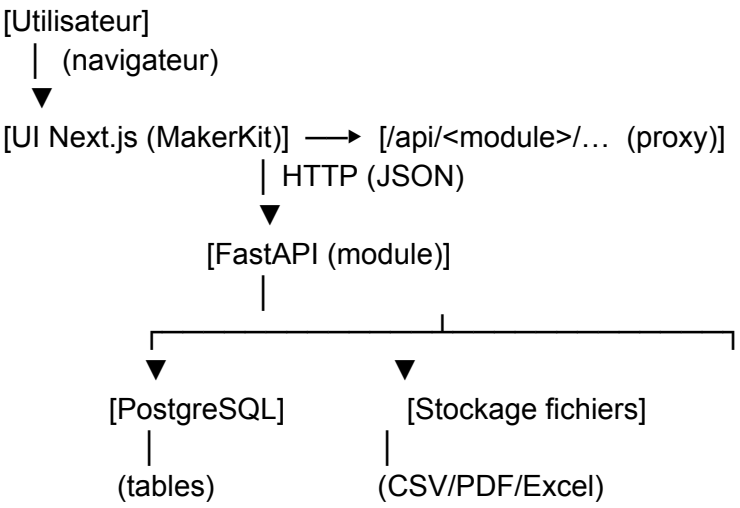
4) Parcours clés (User flows)

- **Revue périodique** : ouvrir → se connecter → choisir période/projet → lire **KPI & alertes** → exporter 1 page.
- **Analyse détail** : filtrer → consulter séries/tables → lancer calcul → enregistrer décision/note.
- **Cycle données** : **Importer** fichier → contrôles automatiques → corriger erreurs → confirmer → recalculer.
- **Rapport rapide** : ouvrir **Vue fournisseur/projet** → exporter **briefing PDF (3 pages)**.

5) Architecture technique

- **Socle UI** : MakerKit (**Next.js + TypeScript**) • **Tailwind + shadcn/ui**.
- **API module** : **FastAPI** (+ `pydantic`) • endpoints `/api/<module>/…` proxifiés par Next.
- **Base** : **PostgreSQL** (Supabase ou managé) • **Stockage** : Supabase Storage / MinIO.
- **Exports** : `pandas`, `openpyxl/xlsxwriter` (Excel), `WeasyPrint/ReportLab` (PDF).
- **Observabilité** : logs JSON, Sentry (erreurs).
- **Déploiement** : Docker Compose (dev/on-prem) / CI/CD (GitHub Actions) / VM ou Cloud Run.

Schéma simple d'architecture



Jargon (explications rapides)

- **RBAC** : droits par rôle (Admin/Éditeur/Lecteur).
- **i18n** : internationalisation (activer plusieurs langues via fichiers clé/valeur).
- **Sentry** : outil qui collecte les erreurs de l'appli.
- **Staging** : environnement **de test en ligne** (pas la prod).

6) Modèle de données (schéma minimal – à spécialiser)

Règle : toutes les tables portent `tenant_id`.

Entités type

- `projects` (ou `contexts`) : `id, tenant_id, name, status`.
- `records` (fait métier) : `id, tenant_id, project_id, date, metrics…`.
- `imports` : `id, tenant_id, type, file_url, status[queued|ok|error], report(json)`.
- `actions` (optionnel) : `id, tenant_id, target_ref, type, assignee, due_date, status, note`.
- `attachments` (optionnel) : `id, tenant_id, record_id, file_url, meta`.

Index & intégrité

- Index `(tenant_id, project_id, date)` sur `records`.
- FK explicites (on delete restrict).
- Champs dates en ISO (UTC).

7) Formats d’import / export

Import CSV/Excel (obligatoire)

- Fichiers modèles fournis (colonnes, formats, exemples).
- Contrôles : colonnes requises, types, dates valides, doublons.
- Rapport d’import : lignes en erreur, corrections proposées.

Export

- **Excel** (tableau principal) + **PDF** (briefing 1–3 pages).
- Nommage : `<module>_<client>_<YYYYMMDD>_<hmm>.xlsx|pdf`.

Exemples de modèles CSV

Commandes (`orders.csv`)

po_number,supplier_code,sku,qty,date_promised,date_actual
PO-001,SUP-A,SKU-10,120,2025-05-01,2025-05-04
PO-002,SUP-B,SKU-22,80,2025-05-02,

Défauts (`defects.csv`)

date,supplier_code,sku,qty_received,qty_defect
2025-05-01,SUP-A,SKU-10,120,2
2025-05-02,SUP-B,SKU-22,80,0

8) Règles de gestion (modèle à compléter)

- **Calculs** : définir clairement inputs → formules → arrondis → unités.
- **Scores/Seuils** : `score ∈ [0..100]`, seuils **configurables** (`.env` ou table `settings`).
- **Couleurs** : Rouge / Orange / Vert (valeurs exactes).
- **Horodatage** : TZ = UTC ; affichage local côté UI.

9) API (contrats – patron générique)

Toutes les routes sont proxifiées par Next : `/api/<module>/…` (auth Bearer / session).

- `GET /health` → `{ ok: true, version }`
- `POST /imports/<type>` → démarre un import : `{ import_id, status }`
- `GET /summary?project_id&from&to` → KPI + cartes + séries agrégées
- `POST /run` → lance un calcul/scoring/inférence : **entrée** `{ data, options }` → **sortie** `{ result, logs?, exportUrl? }`

- `GET /export/<kind>?params…` → retourne un fichier (Excel/PDF)

Erreurs standardisées
400 (validation), 401 (auth), 403 (tenant), 404 (ressource), 422 (schéma), 500 (général).

**Ex## 10) Écrans (gabarits UI)

1. **Dashboard (synthèse)**
   Filtres (période, projet, entité…) • KPI • alertes • **Exporter 1-page**.

2. **Analyse (détail)**
   Graphs/tables • panneau “Raisons / explications” • **Lancer calcul** • **Exporter**.

3. **Données (import/export)**
   Upload → validation → rapport d’erreurs → confirmation • historique imports/exports.

Wireframes (très simples)

- **Dashboard** : Bandeau de filtres en haut → 3 KPI (cartes) → Table “alertes récentes” → Bouton **Exporter 1-page**.

- **Analyse** : Graph à gauche + Table à droite → Panneau latéral “Raisons/explications” → Bouton **Lancer calcul**.

- **Données** : Zone **Upload** → Liste des erreurs (table) → Bouton **Confirmer** → Historique des imports.

11) Non-fonctionnels},

"exportUrl": null
}

- **Format d’erreur**
```json
{ "error": { "code": 400, "message": "Invalid dates" } }
```

10) Écrans (gabarits UI)

1. **Dashboard (synthèse)**
   Filtres (période, projet, entité…) • KPI • alertes • **Exporter 1-page**.

2. **Analyse (détail)**
   Graphs/tables • panneau “Raisons / explications” • **Lancer calcul** • **Exporter**.

3. **Données (import/export)**
   Upload → validation → rapport d’erreurs → confirmation • historique imports/exports.

11) Non-fonctionnels

- **Perf** : page ≤ 2 s (10k lignes cumulées), import ≤ 2 min (50k lignes).
- **Disponibilité** : 99,5% (SaaS) • Backups quotidiens.
- **Sécurité** : RBAC, filtrage strict par `tenant_id`, TLS, logs d’accès.
- **Accessibilité** : contrastes, clavier, tailles adaptatives.
- **i18n** : FR d’abord, EN/AR activables (clé/valeur).

NFR chiffrés

- **Performance** : p95 latence API < **400 ms** (lecture), < **1.5 s** (calcul léger).
- **Import** : fichier ≤ **25 MB**, ≤ **50k lignes** en < **120 s**.
- **Téléchargement** : export Excel/PDF < **5 s** (10k lignes).
- **SLA SaaS** : **99.5%** ; sauvegardes quotidiennes, rétention 14 jours.
- **Sécurité** : contrôle strict `tenant_id` **à chaque requête** ; logs d’accès conservés 90 jours.

12) Tests & qualité

- **Unitaires (Python)** : parsing, règles de calcul, agrégations.
- **Contrats API** : schémas `pydantic` (200/4xx).
- **E2E UI (Playwright)** : importer → voir → exporter.
- **Données de démo** : dossier `assets/demo/` (jeu minimal reproductible).

Definition of Done (checklist)

- ☐ 1 écran opérationnel
- ☐ 1 endpoint principal (`/run`) → JSON propre
- ☐ 1 export (Excel ou PDF) téléchargeable
- ☐ 0 secret en dur (tout dans `.env`)
- ☐ 1 README (installer, lancer, tester)
- ☐ Tests unitaires clés + script de démo

13) Livrables attendus

- **Code** :
    - Front (MakerKit) : `apps/web/app/(marketing|home)/<module>/page.tsx` + proxy `app/api/<module>/…/route.ts`.
    - Package UI réutilisable : `packages/features/<module>/…`
    - API Python : `modules/<module>/api` (FastAPI) + `modules/<module>/assets` (démos, templates).
- **Docs** : README (10 étapes), **Schéma données**, **Contrats API**, **Changelog**.
- **Exemples** : fichiers d'import modèle, exports d'exemple.
- **Infra** : `docker-compose.yml` (local), scripts CI/CD (build images).

14) Planning indicatif (Sprints)

1. **S1** : Schéma DB + endpoints base (`/health`, `/summary`) + écran Dashboard (lecture JSON).
2. **S2** : Import/validation + écran Données + export Excel.

3. **S3** : Endpoint `/run` (calcul/inférence) + écran Analyse + export PDF.
4. **S4** : Durcissement (RBAC, perfs), tests E2E, packaging Docker, doc finale.

15) Risques & mitigations

- **Données hétérogènes** → contrôles stricts + rapport d'erreurs + fichiers modèles.
- **UX complexe** → 3 écrans max + libellés simples + tooltips.
- **Montée en charge** → pagination, index DB, jobs asynchrones si besoin.

16) KPI de succès (produit)

- **Temps de revue** cible `<X min>` ;
- **Taux d'actions traitées** `<Y %>` ;
- **Baisse du risque / défaut** sur `<N>` jours ;
- **Adoption** : utilisateurs actifs / exports générés.

17) Convention & configuration

- **Nommage fichiers** : `<module>_<client>_<date>_<heure>.<ext>`.
- **.env (préfixe module)** :
    - `<MODULE>_DB_URL=…` • `<MODULE>_STORAGE_BUCKET=…` • `<MODULE>_…`
- **Branches** : `main` (stable), `dev` (int), `feat/<module>`.

**Ta## 18) Planning de réalisation (6 semaines)

Objectif : livrer un MVP utilisable en 6 semaines, avec 3 revues intermédiaires et des critères d'acceptation clairs.

Semaine 1 — Cadrage & socle

- **Livrables** :
    - Repo créé (`web`, `modules/<module>`), environnements `.env` initiaux, Docker Compose de base.
    - Maquette UI (3 écrans) + arborescence des routes.
    - Schéma de données initial + migrations.
    - Endpoints squelette : `GET /health`, `GET /summary` (mock JSON).
- **Critères d'acceptation** : app Next.js lance, API FastAPI répond, navigation UI basique ok.
- **Risques & parades** : périmètre flou → valider user stories & DoD lors du kickoff.

Semaine 2 — Ingestion & validation des données

- **Livrables** :
    - Écran **Données** (import CSV/Excel), contrôles (colonnes/types/dates/doublons).
    - Rapport d'erreurs exportable (CSV/PDF) ; persistance en DB.
    - Données de démo réalistes (`assets/demo/`).
- **Critères d'acceptation** : import ≤ 2 min (50k lignes), rapport d'erreurs lisible, données prêtes pour calculs.
- **Jalon** : **Sprint Review #1** (S1–S2) → validation par le PO.

Semaine 3 — Logique métier v1 & exports Excel

- **Livrables** :
    - Endpoint principal : `POST /run` (calcul/scoring/inférence v1).
    - Écran **Analyse** (table/graph) + bouton **Exporter Excel**.
    - Tests unitaires sur parsing & règles de calcul.

- **Critères d’acceptation** : `/run` stable (200/4xx), Excel téléchargé conforme au modèle.
- **Risques & parades** : qualité de données → valeurs par défaut & règles d’imputation documentées.

Semaine 4 — UX/perf, RBAC & E2E

- **Livrables** :
    - Pagination, index DB, caches simples ; RBAC minimal (Admin/Éditeur/Lecteur).
    - Tests E2E (import → analyse → export) ; instrumentation basique (logs/Sentry).
    - Améliorations UI (tooltips, couleurs seuils, responsive desktop).
- **Critères d’acceptation** : page ≤ 2 s (10k lignes cumulées), flux E2E vert.
- **Jalon** : **Sprint Review #2** (S3–S4) → go/no-go v1 feature freeze.

Semaine 5 — PDF, i18n & staging

- **Livrables** :
    - Export **PDF** (briefing 1–3 pages) ; i18n FR→EN (clé/valeur).
    - Observabilité (tableau d’état imports, métriques simples) ; build CI/CD.
    - Déploiement **staging** (Docker/VM ou Cloud Run) + jeu de tests fumée.
- **Critères d’acceptation** : PDF conforme (logos, date, version), staging accessible aux testeurs.
- **Risques & parades** : CSS PDF → modèles simples (WeasyPrint) + marges testées.

Semaine 6 — Durcissement & mise en prod

- **Livrables** :
    - Sécurité (filtrage `tenant_id`, TLS), sauvegardes, plan de migration.

    - Documentation finale : README (10 étapes), Schéma données, Contrats API, Changelog.
    - Plan de **rollback** + **hypercare 1 semaine** (support correctifs mineurs).
- **Critères d'acceptation** : checklist DoD complète, démo finale validée.
- **Jalon** : **Release v1.0 (Sprint Review #3)**.

Revues & gouvernance

- **Reviews** : fin S2, S4, S6 (demo + décision).
- **Daily 15 min** ; **Retro** en fin S3 et S6.
- **Rôles (RACI light)** : PO (R), Dev front (A/R), Dev back (A/R), Data/ML (R), QA (C/A), Ops (C).

Indicateurs d'avancement (hebdo)

- Features complétées vs plan (burn-up), taux de tests passant (unit/E2E), temps de build, erreurs Sentry.

19) Matrice d'environnements (local / staging / prod)

Environnement	URL front	URL API	DB	Stockage	Auth	Backups
Local	http://localhost:3000	http://localhost:8000	Docker Postgres	Local/MinIO	Dev	Snapshots manuels
Staging	`app-staging.<domaine>`	`api-staging.<domaine>`	Postgres managé (small)	Bucket staging	Comptes test	Quotidiennes
Prod	`app.<domaine>`	`api.<domaine>`	Postgres managé (HA)	Bucket prod	Comptes réels	Quotidiennes + rétention 14 j

20) Registre de risques (exemple)

Risque	Prob.	Impact	Mitigation	Plan B
Import volumineux lent	M	M	Limite 25 MB, batch, feedback UI	Upload SFTP
PDF non conforme	M	M	Modèle sobre + test S3	Export Excel uniquement v1
Fuite de données (tenant)	B	H	Tests + middleware tenant	Audit log + blocage
Données manquantes	H	M	Règles d'imputation + rapport erreurs	Valeurs par défaut

Annexes – Squelettes (à garder pour tous les modules)

- **Critères d'acceptation** : checklist DoD complète, démo finale validée.
- **Jalon** : **Release v1.0 (Sprint Review #3)**.

Revues & gouvernance

- **Reviews** : fin S2, S4, S6 (demo + décision).
- **Daily 15 min** ; **Retro** en fin S3 et S6.
- **Rôles (RACI light)** : PO (R), Dev front (A/R), Dev back (A/R), Data/ML (R), QA (C/A), Ops (C).

Indicateurs d'avancement (hebdo)

- Features complétées vs plan (burn-up), taux de tests passant (unit/E2E), temps de build, erreurs Sentry.

1) Route proxy Next.js `apps/web/app/api/<module>/run/route.ts`

```
import { NextResponse } from "next/server";
export async function POST(req: Request) {
  const body = await req.json();
  const r = await fetch(`${process.env.<MODULE>_API_URL}/run`, {
    method: "POST", headers: { "Content-Type": "application/json" },
    body: JSON.stringify(body),
  });
  return NextResponse.json(await r.json(), { status: r.status });
}
```

2) Endpoint principal FastAPI `modules/<module>/api/main.py`

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Payload(BaseModel):
    data: dict
    options: dict | None = None

@app.post("/run")
def run(p: Payload):
    # TODO: logique métier
    result = {"ok": True}
    export_url = None  # une fois le fichier généré
    return {"result": result, "exportUrl": export_url}
```

3) Structure de module

```
modules/
  <module>/
    api/          # FastAPI
    assets/       # demo data, templates export
    README.md     # installer, lancer, tester
packages/
  features/
    <module>/
      ui/         # composants React réutilisables
apps/
  web/
    app/
      (marketing|home)/
```

```
    <module>/page.tsx    # wrapper de route
app/api/<module>/...     # proxy API
```