Plan de codage – Agrégateur « Sites Abandonnés » (France)

Spécification technique avant développement (sans code exécutable)

Date : 2025-11-04 16:08

Ce document décrit l'architecture, les scripts, les appels de pages, les API, les schémas de données et les workflows DevOps pour l'application RJS (React/Next.js) d'agrégation légale de « sites abandonnés » sur toute la France, avec filtres régionaux et scoring explicable.

1. Objectifs & périmètre (Definition of Done)

• Robustesse : un résultat dégradé est toujours affiché (snapshots/caches) avec explication claire. • Observabilité : console intégrée (logs, santé des sources, métriques, audit). • Recoupement : fusion multi-sources (Mérimée, Cartofriches, BASIAS/BASOL, OSM, Admin-Express, ANFR, IGN/Ortho) avec scoring 0–5★ explicable. • Filtrage : France entière, filtre par Région/Département/Commune et bbox. • Éthique & légal : exclusion par défaut des privés/militaires/dangereux. • DevOps : CI/CD, cron ETL, snapshots régionaux versionnés.

2. Structure de dépôt (monorepo conseillé)

```
/agregateur-fr/
■■ apps/
■  ■■ web/                    # Front Next.js (RJS) - UI carte, filtres, console
■  ■■ api/                    # Backend serverless (Node/Fastify) - endpoints REST
■■ data/
■  ■■ etl/                    # Scripts ETL (Python/Node), connecteurs par source
■  ■■ snapshots/              # GeoJSON/MBTiles versionnés par région
■  ■■ schemas/                # OpenAPI, JSONSchema, SQL DDL
■■ packages/
■  ■■ shared/                 # Types TypeScript, constantes, règles de scoring
■  ■■ ui/                     # Composants réutilisables (carte, rating, console)
■■ .github/workflows/         # CI/CD & ETL (cron)
■■ docker/                    # Dockerfiles, compose (dev ETL)
■■ Makefile
■■ pnpm-workspace.yaml
■■ package.json
■■ README.md
■■ LICENSE
```

3. Environnements & secrets

```
# apps/web/.env.local
NEXT_PUBLIC_API_BASE=https://api.example.com
NEXT_PUBLIC_MAP_STYLE=https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png
NEXT_PUBLIC_DEFAULT_REGION=IDF

# apps/api/.env
POSTGRES_URL=postgresql://user:pass@host:5432/db
JWT_SECRET=change-me
ALLOWED_ORIGINS=https://example.github.io,https://example.vercel.app

# data/etl/.env
GEFABRIK_FR_URL=https://download.geofabrik.de/europe/france-latest.osm.pbf
OVERPASS_URL=https://overpass-api.de/api/interpreter
CULTURE_API=https://data.culture.gouv.fr/api/records/1.0/search/
CEREMA_API=...      # Cartofriches (si exposé)
GEORISQUES_API=...  # BASIAS/BASOL
ANFR_API=...        # Open data
```

4. Scripts npm (racine & apps)

```
// package.json (racine, PNPM conseillé)
{
  "private": true,
  "workspaces": ["apps/*", "packages/*", "data/etl"],
  "scripts": {
    "dev": "pnpm -r --parallel dev",
    "build": "pnpm -r build",
    "lint": "pnpm -r lint",
    "test": "pnpm -r test",
    "etl:run": "pnpm --filter @data/etl start",
    "etl:snapshot": "pnpm --filter @data/etl snapshot",
    "db:migrate": "pnpm --filter @apps/api db:migrate",
    "start": "pnpm --filter @apps/web start"
  }
}
```

5. Appels aux pages & routing (Next.js)

Pages principales (App Router) : carte, liste, fiche lieu, régions, santé (console), mentions légales.

```
apps/web/app/
■■ layout.tsx
■■ page.tsx                    # / → carte + filtres + liste
■■ site/
■  ■■ [id]/page.tsx           # /site/{id} → fiche détaillée + justif du score
■■ regions/
■  ■■ [code]/page.tsx         # /regions/{code} → vue filtrée
■■ health/page.tsx            # /health → console (santé des sources, logs)
■■ about/page.tsx
■■ legal/page.tsx
```

Flux d'appels front → API (exemples) :

```
// Initialisation carte (France ou région par défaut)
GET /api/regions
GET /api/sites?region=IDF&score_min=3&limit=500

// Sur zoom/bbox
GET /api/sites?bbox=48.73,2.21,48.92,2.53&type=ruins&score_min=2

// Fiche lieu
GET /api/sites/{id}

// Console Santé
GET /api/health
GET /api/snapshots/meta
```

6. API REST – contrat (extrait OpenAPI)

```
openapi: 3.1.0
info:
  title: Agregateur FR API
  version: 1.0.0
servers:
  - url: https://api.example.com
paths:
  /api/regions:
    get:
      summary: Liste des régions (code INSEE + bbox)
      responses:
        "200":
          description: OK
  /api/sites:
    get:
      summary: Recherche de sites
      parameters:
        - in: query; name: region; schema: {type: string}
        - in: query; name: dept; schema: {type: string}
```

```
        - in: query; name: bbox; schema: {type: string} # south,west,north,east
        - in: query; name: type; schema: {enum: [patrimonial, friche, indus, ruins-osm]}
        - in: query; name: score_min; schema: {type: number}
        - in: query; name: limit; schema: {type: integer, default: 500}
      responses:
        "200": {description: OK}
  /api/sites/{id}:
    get:
      summary: Détail d'un site (sources + score + explications)
      responses: {"200": {description: OK}, "404": {description: Not found}}
  /api/snapshots/{region}.geojson:
    get:
      summary: Snapshot régional (GeoJSON)
      responses: {"200": {description: OK}}
  /api/health:
    get:
      summary: Santé (fraîcheur des données + disponibilité des sources)
      responses: {"200": {description: OK}}
```

7. Modèle de données (PostGIS) – DDL simplifié

```
-- Table des régions (ADMIN-EXPRESS simplifié)
CREATE TABLE regions (
  code        VARCHAR(5) PRIMARY KEY,
  name        TEXT NOT NULL,
  geom        geometry(MULTIPOLYGON, 4326)
);

-- Table des sites (fusion multi-sources)
CREATE TABLE sites (
  id          UUID PRIMARY KEY,
  name        TEXT,
  kind        VARCHAR(24) CHECK (kind IN ('patrimonial','friche','indus','ruins-osm')),
  region_code VARCHAR(5) REFERENCES regions(code),
  dept_code   VARCHAR(3),
  access_flag VARCHAR(12) CHECK (access_flag IN ('public','unknown','excluded')) DEFAULT 'unknown',
  risk_flags  TEXT[],
  score       NUMERIC(2,1),
  score_reasons JSONB,
  links       JSONB,
  last_seen   TIMESTAMP WITH TIME ZONE,
  geom        geometry(GEOMETRY, 4326)
);

-- Table sources (piste d'audit)
CREATE TABLE site_sources (
  site_id     UUID REFERENCES sites(id),
  source      VARCHAR(24) NOT NULL, -- 'merimee','cartofriches','basias','basol','osm'
  source_id   TEXT,
  fetched_at  TIMESTAMPTZ,
  payload     JSONB,
  PRIMARY KEY (site_id, source)
);

-- Index spatiaux
CREATE INDEX idx_sites_geom ON sites USING GIST (geom);
CREATE INDEX idx_regions_geom ON regions USING GIST (geom);
```

8. ETL – pipeline d’ingestion & fusion (par source)

Étapes communes : fetch → normalise → import (staging) → enrich → merge/dedupe → score → snapshot.

```
# data/etl/Makefile
.PHONY: all geofabrik merimee cartofriches georisques merge score snapshot

all: geofabrik merimee cartofriches georisques merge score snapshot

geofabrik:
■python etl_osm_geofabrik.py --pbf "$(GEFABRIK_FR_URL)" --out staging/osm.geojson

merimee:
```

```
■python etl_merimee.py --api "$(CULTURE_API)" --out staging/merimee.geojson

cartofriches:
■python etl_cartofriches.py --api "$(CEREMA_API)" --out staging/cartofriches.geojson

georisques:
■python etl_georisques.py --api "$(GEORISQUES_API)" --out staging/georisques.geojson

merge:
■python merge_dedupe.py --in staging --out staging/merged.geojson

score:
■python score_sites.py --rules rules/score.yml --in staging/merged.geojson --out staging/scored.geojson

snapshot:
■python snapshot_export.py --in staging/scored.geojson --out ../snapshots/fr-IDF.geojson
```

9. Règles de scoring (déclaratif)

```
# packages/shared/rules/score.yml
version: 1
weights:
  osm_abandoned: 2.0
  osm_disused: 1.0
  osm_historic_ruins: 1.5
  merimee_ruines: 2.0
  cartofriches_present: 1.0
  basias_cessation: 0.5
penalties:
  access_private: -2.0
  military_zone: -99.0   # exclusion en amont
exclusions:
  - tag: military
    operator: exists
explanations:
  merimee_ruines: "Notice patrimoniale mentionne 'ruines' ou 'vestiges'"
  osm_abandoned: "Préfixe lifecycle abandoned:*"
  osm_historic_ruins: "Tag historic=ruins"
```

10. Fallbacks & circuit breaker (fetch wrapper)

```
// packages/shared/net/fetcher.ts (pseudo-code)
export async function resilientFetch(url, opts) {
  const t0 = Date.now();
  try {
    const ctrl = new AbortController();
    const id = setTimeout(()=>ctrl.abort(), opts.timeout ?? 8000);
    const res = await fetch(url, {...opts, signal: ctrl.signal, headers:{...opts.headers, "User-Agent":"agregateu
    clearTimeout(id);
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    return {ok:true, data: await res.json(), ms: Date.now()-t0};
  } catch (e) {
    // retry once with jitter
    await new Promise(r=>setTimeout(r, 200 + Math.random()*400));
    try {
      const res2 = await fetch(url, opts);
      if (!res2.ok) throw new Error(`HTTP ${res2.status}`);
      return {ok:true, data: await res2.json(), ms: Date.now()-t0};
    } catch (e2) {
      return {ok:false, error:String(e2), ms: Date.now()-t0};
    }
  }
}
```

11. Console intégrée (onglets & métriques)

```
Tabs:
- Pipeline: étapes et durées (fetch, normalize, merge, score, serve)
- Sources: état par source (uptime 24h, latence p50/p95, fallbacks actifs)
- Données: nb sites par type, taux de fusion/dédoublonnage, % champs manquants
```

```
- Scoring: histogramme des étoiles, règles déclenchées, dérive vs snapshot N-1
- Dépendances: npm outdated/audit (résumé), versions libs
- Journal: logs JSON (ts, corrId, route, source, bbox, duration_ms, outcome)
- Licences: ODbL/Etalab/IGN/ANFR
Fonctions:
- Export diagnostic (.zip) : logs + health + env masqué
- Simulateur de panne : désactiver une source (feature flag) pour QA
```

12. GitHub Actions – CI & ETL

```
# .github/workflows/ci.yml
name: CI
on: [push, pull_request]
jobs:
  build-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: pnpm/action-setup@v4
        with: { version: 9 }
      - run: pnpm install --frozen-lockfile
      - run: pnpm lint && pnpm test && pnpm build
```

```
# .github/workflows/etl.yml
name: ETL Nightly
on:
  schedule:
    - cron: "15 2 * * *"   # 02:15 UTC every day
  workflow_dispatch: {}
jobs:
  etl:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: '3.11' }
      - uses: pnpm/action-setup@v4
        with: { version: 9 }
      - run: pnpm install --frozen-lockfile
      - run: cp data/etl/.env.example data/etl/.env  # secrets via env vars
      - run: pnpm etl:run
      - run: pnpm etl:snapshot
      - name: Upload snapshots
        uses: actions/upload-artifact@v4
        with:
          name: snapshots
          path: data/snapshots/**
```

13. Journaux & erreurs – format JSON

```
{
  "ts": "2025-11-04T09:30:12.345Z",
  "level": "info",
  "corr_id": "ab12-...",
  "route": "GET /api/sites",
  "source": "merimee",
  "bbox": "48.73,2.21,48.92,2.53",
  "duration_ms": 412,
  "outcome": "fallback_snapshot",   // ok | error | fallback_live | fallback_snapshot
  "msg": "Degraded mode: data.culture timeout, served snapshot 2025-11-03"
}
```

14. Tests & QA – plan

• Unit : normaliseurs de sources, fusion, scoring (cas limites : access=private, military, tags lifecycle). • Contract : schémas d’API (OpenAPI), champs indispensables présents et typés. • E2E (Playwright) : parcours carte→filtre→fiche→export ; simulation de pannes (feature flag). • Charge : vue Île-de-France, cluster et recherche bbox; médiane < 1,5 s avec cache chaud. • Chaos : désactiver une source (Overpass down) →

bascule snapshots + badge.

15. Performance, cache & distribution

• Snapshots régionaux (GeoJSON ≤ quelques Mo) servis via CDN ; option tuiles vectorielles (pbf) si volumétrie élevée. • Cache navigateur (ETag) + revalidation ; pré-chargement région par défaut. • Aucune requête Overpass « France entière » en live ; extraction Geofabrik en ETL.

16. Sécurité, licences & bannière légale

• Filtre légal strict par défaut (masquage privé/militaire/dangereux). • Mentions ODbL (OSM), Etalab/MTE/Cerema, IGN/ANFR ; attribution en pied de page. • Pas de scraping intrusif ; respect rate-limits ; pas de données personnelles. • Bannière : “Information documentaire. Ne pas pénétrer sur des sites privés/dangereux. Vérifiez les statuts locaux.”

17. Annexe – Makefile & commandes utiles

```
# Makefile (racine) – cibles utiles
.PHONY: dev build test etl health snapshot

dev:          ## Lance web + api en parallèle
■pnpm dev

build:        ## Build monorepo
■pnpm build

test:         ## Lint & tests
■pnpm lint && pnpm test

etl:          ## Lance ETL (local)
■cd data/etl && make all

snapshot:     ## Exporte snapshots régionaux
■pnpm etl:snapshot

health:       ## Génère le Health Report
■node apps/api/scripts/health-report.js
```

Fin du document – prêt pour implémentation.