Décomposition Statique du Code — Jira Issue Extractor

Structure des répertoires

```
jira-extractor-app/
│
├── Documentation/              ← Documentation projet (ce répertoire)
│
├── alembic/                    ← Migrations de base de données
│   ├── env.py                  ← Environnement Alembic (async, SQLite/PG)
│   ├── script.py.mako          ← Template de migration
│   └── versions/
│       ├── 0001_initial_schema.py  ← Migration initiale (toutes les tables)
│       └── 0002_add_confluence_schedule_fields.py  ← Champs Confluence dans schedules
│
├── backend/                    ← Application FastAPI (Python 3.11)
│   ├── main.py                 ← Point d'entrée : app FastAPI, lifespan, routers
│   │
│   ├── jira/                   ← Tout ce qui est spécifique à Jira
│   │   ├── auth.py             ← Construction du client Jira + OAuth2 token refresh
│   │   ├── client.py           ← Wrapper async autour de la lib jira
│   │   ├── attachments.py      ← Téléchargement pièces jointes
│   │   ├── copy_processor.py   ← Suivi de progression (compteurs, erreurs)
│   │   ├── exporters/
│   │   │   ├── xml_exporter.py  ← Export XML
│   │   │   ├── html_exporter.py ← Export HTML (rendu complet)
│   │   │   ├── json_exporter.py ← Export JSON (payload brut Jira)
│   │   │   ├── csv_exporter.py  ← Export CSV (tabulaire)
│   │   │   └── index_builder.py ← Génération _index.html
│   │   └── api/
│   │       ├── export.py       ← /export (lancement export Jira)
│   │       ├── projects.py     ← /projects, /server-info
│   │       └── search.py       ← /search/validate, /search/preview
│   │
│   ├── confluence/             ← Tout ce qui est spécifique à Confluence
│   │   ├── client.py           ← Client HTTP async Confluence REST API
│   │   ├── exporters/
│   │   │   ├── html_exporter.py ← Export page Confluence en HTML
│   │   │   ├── json_exporter.py ← Export page en JSON (payload brut API)
│   │   │   ├── markdown_exporter.py ← Export page en Markdown
│   │   │   └── index_builder.py ← Génération index HTML (arborescence)
│   │   └── api/
│   │       └── export.py       ← /confluence/* (export, espaces, pages, CQL)
│   │
│   ├── common/                 ← Utilitaires partagés Jira + Confluence
│   │   └── archive_builder.py  ← Création archive ZIP
│   │
│   ├── api/                    ← Endpoints transverses (auth, jobs, utilisateurs…)
│   │   ├── user_auth.py        ← JWT, hachage bcrypt, tokens email
│   │   ├── dependencies.py     ← Injecteurs FastAPI (get_current_user, require_editor…)
│   │   └── endpoints/
│   │       ├── auth.py         ← /auth/*
│   │       ├── admin.py        ← /admin/users/*
│   │       └── config.py       ← /config/*
```

```
│   │       ├── connections.py   ← /connections/*
│   │       ├── jobs.py          ← /jobs/* (statut, SSE, browse, download, files)
│   │       ├── history.py       ← /history/*
│   │       ├── schedules.py     ← /schedules/*
│   │       ├── directories.py   ← /directories/*
│   │       └── oauth2_callback.py ← /oauth2/callback
│   │
│   ├── config/
│   │   └── settings.py          ← Paramètres globaux (Pydantic Settings)
│   │
│   ├── db/                      ← Couche persistance
│   │   ├── database.py          ← Engine SQLAlchemy + init DB + runner Alembic
│   │   ├── models.py            ← Modèles ORM (SQLAlchemy 2.0 déclaratif)
│   │   └── repository.py        ← Data Access Objects (CRUD par domaine)
│   │
│   ├── jobs/                    ← Gestion des jobs d'export
│   │   ├── job_manager.py       ← État en mémoire + SSE + Redis
│   │   └── redis_bus.py         ← Client Redis pub/sub (optionnel)
│   │
│   ├── storage/
│   │   └── minio_store.py       ← Client MinIO objet (optionnel)
│   │
│   ├── scheduler/
│   │   ├── main.py              ← Point d'entrée conteneur scheduler
│   │   └── runner.py            ← Boucle de planification
│   │
│   └── utils/
│       └── email.py             ← Envoi SMTP asynchrone
│
├── frontend/                    ← Application React 18 / Vite / Tailwind
│   ├── package.json
│   ├── vite.config.js           ← Proxy /api → backend:8080
│   ├── tailwind.config.js
│   └── src/
│       ├── main.jsx             ← Point d'entrée React (ReactDOM.createRoot)
│       ├── App.jsx              ← Layout principal + routage par state
│       ├── index.css
│       ├── api/
│       │   └── client.js        ← Instance Axios + toutes les fonctions API
│       ├── pages/               ← Composants de page (routage géré dans App.jsx)
│       │   ├── LoginPage.jsx
│       │   ├── SetPasswordPage.jsx
│       │   ├── HistoryPage.jsx
│       │   ├── ConfluencePage.jsx
│       │   ├── ConnectionsPage.jsx
│       │   ├── SchedulesPage.jsx
│       │   ├── AdminUsersPage.jsx
│       │   ├── DirectoriesPage.jsx
│       │   └── FileBrowserPage.jsx
│       ├── components/          ← Composants réutilisables
│       │   ├── ConnectionForm.jsx
│       │   ├── ConnectionPicker.jsx
│       │   ├── ProjectSelector.jsx
│       │   ├── JQLBuilder.jsx
│       │   ├── FormatSelector.jsx
│       │   ├── OutputConfig.jsx
│       │   ├── DirectoryPicker.jsx
│       │   ├── ProgressPanel.jsx
│       │   └── JobStatusBadge.jsx
│       └── context/
│           └── AuthContext.jsx   ← Contexte React d'authentification
│
├── nginx/                       ← Config nginx optionnelle
└── alembic.ini                  ← Configuration Alembic
```

```
■■■ docker-compose.yml          ← Orchestration production
■■■ Dockerfile                  ← Build multi-stage (frontend + backend)
■■■ requirements.txt            ← Dépendances Python
■■■ .env.example                ← Template de configuration
```

Modèles de données (ORM SQLAlchemy)

`Connection`

Stocke une instance Jira avec ses paramètres d'authentification.

Champ	Type	Description
`id`	String PK	UUID v4
`name`	String	Nom d'affichage
`instance_url`	String	URL Jira (ex: https://acme.atlassian.net)
`instance_type`	String	`cloud` ou `onprem`
`auth_type`	String	`basic`, `pat`, `oauth2`
`username`	String?	Identifiant (Basic Auth)
`password`	String?	Mot de passe ou API token (chiffré)
`token`	String?	PAT (Personal Access Token)
`verify_ssl`	Boolean	Validation du certificat SSL
`oauth2_client_id`	String?	Client ID OAuth2
`oauth2_client_secret`	String?	Secret OAuth2
`oauth2_access_token`	String?	Token d'accès courant
`oauth2_refresh_token`	String?	Token de rafraîchissement
`oauth2_token_expiry`	DateTime?	Expiration du token
`created_at`, `updated_at`	DateTime	Horodatages

`Job`

Enregistrement d'un export (en cours ou terminé).

Champ	Type	Description
`id`	String PK	UUID v4
`status`	String	`pending`, `running`, `done`, `failed`, `cancelled`
`jql`	String	Requête JQL exécutée

Champ	Type	Description
`formats`	JSON	Liste des formats : `["xml","html","json","csv","zip"]`
`include_attachments`	Boolean	Pièces jointes incluses
`output_dir`	String	Répertoire de sortie demandé
`output_path`	String?	Chemin effectif (local ou `minio:...`)
`instance_url`	String?	URL Jira utilisée
`error_message`	String?	Message d'erreur si échec
`issue_count`	Integer?	Nombre d'issues exportées
`attachment_count`	Integer?	Nombre de pièces jointes téléchargées
`events_json`	JSON	Log des événements SSE (replay)
`connection_id`	String?	FK → Connection
`created_at`, `started_at`, `finished_at`	DateTime	Horodatages

`User`

Compte utilisateur de l'application.

Champ	Type	Description
`id`	String PK	UUID v4
`email`	String UNIQUE	Identifiant de connexion
`role`	String	`admin`, `editor`, `visitor`
`password_hash`	String?	Haché bcrypt
`is_active`	Boolean	Compte actif
`must_change_password`	Boolean	Force le changement au prochain login
`preferences`	JSON?	Derniers paramètres d'export (JQL, formats…)
`created_at`, `updated_at`	DateTime	Horodatages

`AuthToken`

Tokens à usage unique (invite, reset, browse).

Champ	Type	Description
`id`	String PK	UUID v4
`user_id`	String FK	→ User

Champ	Type	Description
`token_hash`	String	SHA-256 du token brut (jamais stocké en clair)
`purpose`	String	`invite`, `reset`, `browse`, `browse_session`
`expires_at`	DateTime	Date d'expiration
`used_at`	DateTime?	Date d'utilisation (révocation logique)
`created_at`	DateTime	Horodatage

`AppConfig`

Configuration globale clé-valeur.

Champ	Type	Description
`key`	String PK	Clé de configuration
`value`	JSON	Valeur structurée
`updated_at`	DateTime	Dernière modification

`Directory`

Favoris de répertoires de sortie.

Champ	Type	Description
`id`	String PK	UUID v4
`name`	String	Nom d'affichage
`path`	String	Chemin absolu sur le serveur
`created_at`	DateTime	Horodatage

`Schedule`

Planification d'export récurrent.

Champ	Type	Description
`id`	String PK	UUID v4
`name`	String	Nom
`is_active`	Boolean	Activée/désactivée
`jql`	String?	Requête JQL (Jira) ou CQL (Confluence scope=cql)

Champ	Type	Description
`formats`	JSON	Formats d'export
`include_attachments`	Boolean	
`output_dir`	String	Répertoire de sortie
`max_issues`	Integer?	Limite optionnelle
`connection_id`	String?	FK → Connection
`target_type`	String	`jira` ou `confluence`
`confluence_scope`	String?	`space`, `page` ou `cql` (Confluence uniquement)
`space_key`	String?	Clé d'espace Confluence
`root_page_id`	String?	ID de page racine (scope `page`)
`schedule_type`	String	`once`, `daily`, `weekly`
`days_of_week`	JSON?	`[0..6]` (0=lundi) pour `weekly`
`time_of_day`	String?	Format `HH:MM`
`run_once_at`	DateTime?	Pour `once`
`next_run_at`	DateTime?	Prochaine exécution calculée
`last_run_at`	DateTime?	Dernière exécution
`last_job_id`	String?	FK → Job
`created_by`	String?	FK → User
`created_at`, `updated_at`	DateTime	Horodatages

Classes et modules principaux (Backend)

`backend.main` — Point d'entrée

```
app = FastAPI(lifespan=lifespan)
■
■■■ lifespan()
■     ■■■ init_db()          ← Migrations Alembic + bootstrap admin
■     ■■■ minio_store.is_available()  ← Test connexion MinIO
■     ■■■ get_redis()        ← Test connexion Redis
■     ■■■ create_task(scheduler_loop)  ← Si ENABLE_SCHEDULER=true
■
■■■ include_router() × 11   ← Tous les routeurs avec préfixe /api
```

`backend.config.settings.Settings` — Configuration

Classe Pydantic Settings lue depuis les variables d'environnement.

Groupe	Variables clés
Application	`secret_key`, `jwt_secret_key`, `jwt_algorithm`, `jwt_expire_minutes`
Base de données	`database_url`, `postgres_*`
Scheduler	`enable_scheduler`
Redis	`redis_url`
MinIO	`minio_endpoint`, `minio_access_key`, `minio_secret_key`, `minio_bucket`, `minio_secure`
SMTP	`smtp_host`, `smtp_port`, `smtp_user`, `smtp_password`, `smtp_from`
Admin	`admin_email`, `admin_password`

`backend.db.database` — Moteur de base de données

Fonction	Description
`init_db()`	Init async : détecte si Alembic géré, applique migrations via subprocess
`get_db()`	Générateur de session async (dépendance FastAPI)
`_run_alembic(command, revision)`	Lance Alembic en subprocess isolé
`_is_alembic_managed()`	Vérifie la présence des tables `alembic_version` et `users`

`backend.db.repository` — Accès aux données

Objet	Méthodes principales
`job_repository`	`create()`, `get()`, `update_status()`, `update_output()`, `list_all()`, `delete()`
`user_repository`	`get_by_email()`, `get_by_id()`, `create()`, `update()`, `delete()`, `create_auth_token()`, `get_auth_token_by_hash()`, `mark_auth_token_used()`
`connection_repository`	`list()`, `get()`, `create()`, `update()`, `delete()`
`config_repository`	`get(key)`, `set(key, value)`, `get_connection_config()`
`schedule_repository`	`list()`, `get()`, `create()`, `update()`, `delete()`, `get_due_schedules()`
`directory_repository`	`list()`, `create()`, `delete()`

`backend.jobs.job_manager.JobManager` — Gestionnaire de jobs

```
JobManager (singleton: job_manager)
```

```
■
■■■ _jobs: Dict[str, Job]          ← État en mémoire
■■■ _queues: Dict[str, Queue]      ← Files SSE par job
■
■■■ create_job(jql, formats, …)    → job_id
■■■ start_job(job_id, instance_url)
■■■ emit(job_id, event)            → append mémoire + publish Redis
■■■ finish_job(job_id, output_path, …)
■■■ fail_job(job_id, error)
■■■ cancel_job(job_id)
■■■ get_job(job_id)                → Job | None
■■■ get_job_summary(job_id)        → Dict | None
■■■ event_stream(job_id)          → AsyncGenerator[str]
      ■■■ _stream_via_redis()      ← Si Redis disponible
      ■■■ _stream_via_memory()     ← Fallback
```

`backend.storage.minio_store.MinioStore` — Stockage objet

```
MinioStore (singleton: minio_store)
■
■■■ _available: Optional[bool]     ← Cache de disponibilité
■■■ _build_client()                → minio.Minio
■
■■■ is_available() → bool          ← Test ping (lazy)
■■■ upload_path(local_path, job_id) → "minio:{prefix}"
■     ■■■ is_dir → _upload_dir_sync()
■     ■■■ is_file → _upload_file_sync()
■■■ get_object_data(object_name) → (bytes, content_type)
■■■ delete_path(minio_uri)
■■■ path_exists(minio_uri) → bool
```

Toutes les opérations synchrones Minio sont wrappées dans `anyio.to_thread.run_sync()`.

`backend.jira` — Moteur d'extraction Jira

```
jira/api/export.py::_run_export(job_id, request)
■
■■■ jira/auth.py::get_jira_client_from_connection()  ← Auth + token refresh OAuth2
■■■ jira/client.py::JiraClientWrapper.get_all_issues(jql)
■■■ jira/copy_processor.py::CopyProcessor()          ← Suivi de progression
■
■■■ Pour chaque issue :
■     ■■■ jira/exporters/xml_exporter.export_issue_xml()
■     ■■■ jira/exporters/html_exporter.export_issue_html()
■     ■■■ jira/exporters/json_exporter.export_issue_json()
■     ■■■ jira/exporters/csv_exporter.add_issue()
■     ■■■ jira/attachments.extract_attachments()
■
■■■ jira/exporters/index_builder.build_index()       ← Génération _index.html
■
■■■ Si option "zip" :
■     ■■■ common/archive_builder.build_zip()
■
■■■ Si MinIO disponible :
      ■■■ storage/minio_store.upload_path()
```

`backend.confluence` — Moteur d'extraction Confluence

```
confluence/api/export.py::_run_confluence_export(job_id, request)
■
■■■ confluence/client.py::ConfluenceClient
■     ■■■ list_spaces()                    ← Liste des espaces
■     ■■■ get_page_tree(space, root_id)    ← Arborescence récursive
■     ■■■ search_pages(cql)                ← Recherche CQL
■     ■■■ get_page_attachments(page_id)    ← Pièces jointes
■
■■■ Pour chaque page :
■     ■■■ confluence/exporters/html_exporter.export_page_html()
■     ■     (liens internes /pages/PAGE_ID réécrits vers fichiers locaux)
■     ■■■ confluence/exporters/json_exporter.export_page_json()
■     ■■■ confluence/exporters/markdown_exporter.export_page_markdown()
■
■■■ confluence/exporters/index_builder.build_confluence_index()
■
■■■ Si option "zip" :
■     ■■■ common/archive_builder.build_zip()
■
■■■ Si MinIO disponible :
      ■■■ storage/minio_store.upload_path()
```

`backend.scheduler.runner` — Planificateur

```
scheduler_loop()
■
■■■ Boucle infinie (30 s) :
      ■■■ schedule_repository.get_due_schedules()
      ■■■ Pour chaque planification :
      ■     ■■■ Si target_type == "confluence" → _run_confluence_schedule()
      ■     ■■■ Si target_type == "jira"       → _run_jira_schedule()
      ■     ■■■ schedule_repository.update(next_run_at, last_run_at, last_job_id)
      ■■■ asyncio.sleep(30)
```

`compute_next_run(schedule, after)` calcule la prochaine exécution :

- `once` : `run_once_at` (non renouvelé)
- `daily` : lendemain à `time_of_day` (UTC)
- `weekly` : prochain jour de la semaine correspondant à `days_of_week`

Composants Frontend

`App.jsx` — Routeur et état global

```
AppContent
■
■■■ State
■     ■■■ page (export|confluence|history|connections|schedules|directories|users|filebrowser)
```

```
■     ■■■ connection, connected         ← Connexion Jira manuelle
■     ■■■ selectedConnectionId, namedConnected  ← Connexion nommée
■     ■■■ jql, formats, outputDir, maxIssues    ← Paramètres d'export Jira
■     ■■■ jobId, exporting, exportError         ← État export courant
■     ■■■ fileBrowserJob                        ← Navigation fichiers
■
■■■ Pages rendues conditionnellement
      ■■■ page=export      → ConnectionPicker + ConnectionForm + ProjectSelector
      ■                      + JQLBuilder + FormatSelector + OutputConfig + ProgressPanel
      ■■■ page=confluence  → ConfluencePage (espace, périmètre, formats, destination)
      ■■■ page=history     → HistoryPage
      ■■■ page=connections → ConnectionsPage
      ■■■ page=schedules   → SchedulesPage
      ■■■ page=directories → DirectoriesPage
      ■■■ page=users       → AdminUsersPage
      ■■■ page=filebrowser → FileBrowserPage
```

`ProgressPanel.jsx` — Panneau de progression temps réel

- Ouvre un `EventSource` vers `/api/jobs/{id}/stream?token={jwt}`
- Parse les événements SSE : `progress`, `log`, `done`, `error`
- Affiche : barre de progression, log terminal (fond noir), statut
- Ferme le flux à réception de `done` ou `error`

`HistoryPage.jsx` — Historique des exports

Actions par ligne (conditionnelles selon le statut et le type d'export) :

- **Info** : Détail erreur (jobs `failed`)
- ■ **Télécharger** : ZIP uniquement (`output_path` se termine par `.zip`)
- ■ **Ouvrir** : Répertoire HTML uniquement (nécessite browse token)
- ■ **Naviguer** : Répertoire HTML uniquement (FileBrowserPage)
- ■ **Relancer** : Tous statuts (pré-remplit le formulaire d'export)
- ■ **Purger** : Supprime DB + fichiers (MinIO ou filesystem)

`FileBrowserPage.jsx` — Navigateur de fichiers

- Appelle `GET /api/jobs/{id}/files?path={dir}` à chaque navigation
- Affiche une arborescence paginée (répertoires en premier)
- Fil d'Ariane cliquable
- Clic fichier → browse token → ouverture dans un nouvel onglet

`AuthContext.jsx` — Contexte d'authentification

```
AuthProvider
■
■■■ user, loading, token (state)
■■■ login(email, password)     → POST /api/auth/login, stocke token localStorage
```

```
■■■ logout()                    → retire token, recharge la page
■■■ refreshUser()               → GET /api/auth/me

useAuth()                       ← Hook de consommation
```

Dépendances inter-modules

```
backend.main
    ■■■ backend.config.settings
    ■■■ backend.db.database
    ■■■ backend.jobs.job_manager
    ■■■ backend.api.endpoints.*         ← endpoints transverses
    ■■■ backend.jira.api.*              ← endpoints Jira
    ■■■ backend.confluence.api.*        ← endpoints Confluence

backend.api.endpoints.*
    ■■■ backend.api.dependencies
            ■■■ backend.api.user_auth
            ■■■ backend.db.repository
                    ■■■ backend.db.models

backend.jira.api.export
    ■■■ backend.jira.auth               ← client Jira + OAuth2
    ■■■ backend.jira.client             ← wrapper JIRA lib
    ■■■ backend.jira.attachments
    ■■■ backend.jira.copy_processor
    ■■■ backend.jira.exporters.*
    ■■■ backend.common.archive_builder
    ■■■ backend.jobs.job_manager
    ■■■ backend.storage.minio_store

backend.confluence.api.export
    ■■■ backend.confluence.client
    ■■■ backend.confluence.exporters.*
    ■■■ backend.common.archive_builder
    ■■■ backend.jobs.job_manager
    ■■■ backend.storage.minio_store

backend.scheduler.runner
    ■■■ backend.db.repository
    ■■■ backend.jobs.job_manager
    ■■■ backend.jira.api.export (_run_export)
```

Dépendances Python principales

Package	Version	Usage
`fastapi`	≥0.111.0	Framework HTTP async
`uvicorn[standard]`	≥0.30.0	Serveur ASGI

Package	Version	Usage
`sqlalchemy[asyncio]`	≥2.0	ORM + sessions async
`asyncpg`	≥0.29.0	Driver PostgreSQL async
`aiosqlite`	≥0.20	Driver SQLite async (dev)
`alembic`	≥1.13	Migrations de schéma
`pydantic`	≥2.7.0	Validation de données
`pydantic-settings`	≥2.3.0	Lecture des variables d'environnement
`jira`	≥3.8.0	Client API Jira
`httpx`	≥0.27.0	Client HTTP async (OAuth2, attachments)
`python-jose[cryptography]`	≥3.3.0	JWT HS256
`bcrypt`	≥4.0,<5.0	Hachage des mots de passe
`lxml`	≥5.2.0	Génération XML
`minio`	≥7.0	Client MinIO/S3
`redis[asyncio]`	≥5.0	Pub/sub Redis
`sse-starlette`	≥2.1.0	Server-Sent Events
`anyio`	≥4.4.0	Primitives async multi-backend
`aiosmtplib`	≥3.0.0	SMTP async
`aiofiles`	≥23.2.1	I/O fichiers async
`markdownify`	≥0.12	Conversion HTML → Markdown (export Confluence)

Dépendances Frontend principales

Package	Version	Usage
`react`	^18.3.1	Framework UI
`react-dom`	^18.3.1	Rendu DOM
`axios`	^1.7.2	Client HTTP (API calls)
`lucide-react`	^0.400.0	Icônes SVG
`vite`	^5.3.1	Build tool + dev server
`tailwindcss`	^3.4.4	CSS utilitaire