Bloc de Compétences E1:

Data Plateform IMO-Ops

Réalisation de la collecte, le stockage et la mise à disposition des données pour un projet en intelligence artificielle dédié au marché Immobilier.

Microsoft en France
Ecole IA

DEVELOPPEUR·SE EN INTELLIGENCE ARTIFICIELLE

SIMPLON

Microsoft

Rédigé par : Olivier LAVAUD, le 14 Novembre 2025

Pour l'acquisition du Titre RNCP 37827:
Développeur en Intelligence Artificielle

https://github.com/OlivierLAVAUD/imo-ops.git

Data Plateform IMO-Ops pour un projet en intelligence artificielle dédié au marché Immobilier.

Dev IA RNCP 37827 /

Contexte & objectifs:

- L'objectif est de proposer une **Data Plateform** du marché immobilier alimentée par différentes sources (fichiers, web databases, api). Une application IA par exemple un chatbot LLM RAG pourra avoir accès à des données en temps réel structurées et enrichies.

L'architecture (écosystème/stack technologique)

- Orchestration de pipelines avec messaging, parallélisation & mise à l'échelle (Apache Airflow+Redis - Dags - Logs)
- Scraping: Playwright avec fichiers de configuration
- Manipulation et transformation des données: Pandas, Regex, SQLAlchemy, Requests
- Bases de données: Sql (PostgreSQL-pgadmin) & NoSQL (MongoDB)
- sources de fichiers batch: (JSON, ...)
- sources à partir d'APIs: FastAPI
- Rapports & consultation des données (Prometheus- Grafana, Plotly-Dashboards, Gradio)
- Conteneurisation: Docker

Les processus:

- batchs/pipelines de collecte à partir de différentes sources: scraping web, fast API, fichiers csv, bases de données postgres, fichiers csv
- batchs/pipelines de normalisation après l'extraction des données
- batch/pipeline d'agrégation des données en provenance des différentes sources
- batch/pipelines de stockage en base de données SQL & NoSQL
- batch/pipeline d'interrogation des données
- batch/pipeline de mise à disposition des données

Les sources:

site web: IAD-Immobilier (https://www.iadfrance.fr/): annonces achats, ventes, location sur critères multiples, fichiers, web databases, API

Data Plateform IMO-Ops pour un projet en intelligence artificielle dédié au marché Immobilier.

Dev IA RNCP 37827 / © 2025 Olivier LAVAUD

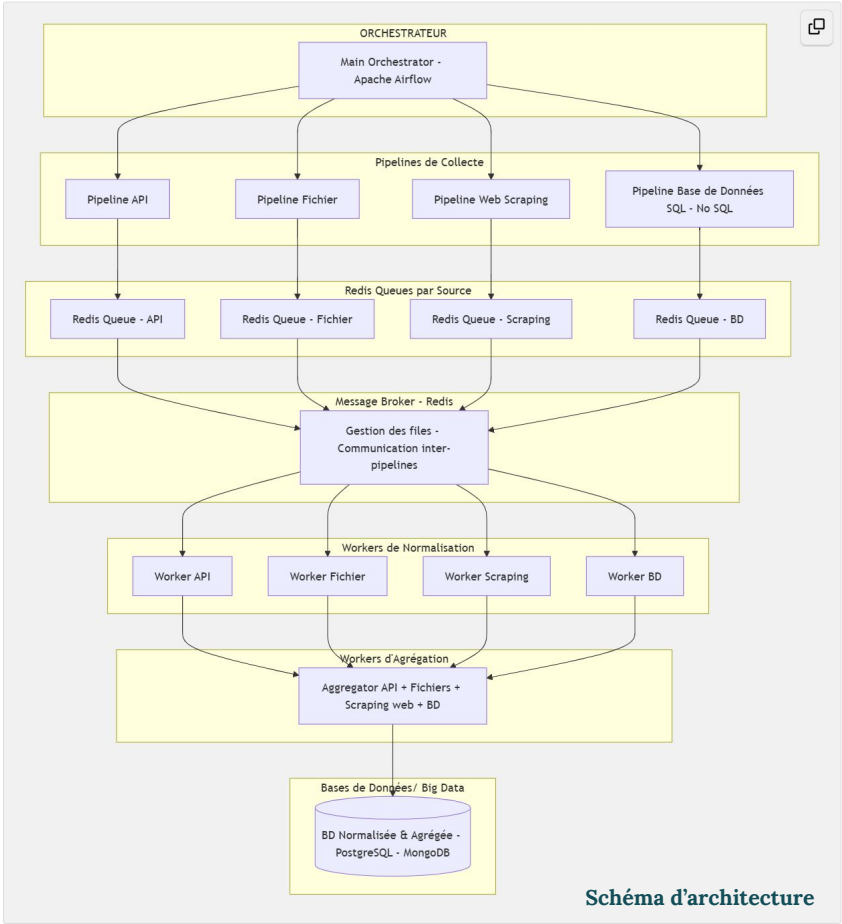


Schéma d'architecture

Compétence C1: Automatisation de l'extraction de données

`Data from Web Scraping, batch files, sources Databases and APIs`

services/c1-scrap (iad-scraper, iad-scraper-custom, extract-from-files, extract-from-api, extract-from-db)

Le scraper web high-tech implémente une architecture asynchrone basée sur Playwright pour le rendu JavaScript complet, avec une gestion avancée du cycle de vie des pages via async/await.

Le système utilise un pattern de stratégies multiples pour le bypass des bannières cookies, combinant sélecteurs CSS dynamiques, injection JavaScript et simulation d'événements claviers.

L'extraction des données suit une approche modulaire avec des classes spécialisées : DataExtractor pour le parsing HTML via XPath/CSS combinés, MediaExtractor pour l'optimisation des URLs multimédias avec regex de nettoyage qualité.

La configuration externalisée en JSON schema (config.json) permet la définition déclarative des sélecteurs, tandis que le pipeline de pagination intelligent gère automatiquement la navigation multi-pages. L'implémentation inclut un system de retry robuste, des timeouts adaptatifs et la génération de datasets structurés avec métadonnées complètes.

Les autres services de collecte suivent tous la même logique de la normalisation avant agrégation en base de données.


```
{
  "site": {
    "name": "IAD France",
    "base_url": "https://www.iadfrance.fr",
    "search_url": "https://www.iadfrance.fr/annonces/vente",
    "wait_timeout": 10000,
    "navigation_delay": 2000
  },

  "navigation_menu": {
    "container": {
      "css_selector": "nav.flex-1",
      "xpath": "//nav[contains(@class, 'flex-1')]",
      "description": "Menu de navigation principal"
    },
```


Scénarios courants

```
# Lancement standard (Paris, 200 biens)
docker-compose up iad-scraper

# Recherche à Toulouse, 25 biens (PowerShell)
$env:LOCALISATION="Toulouse"; $env:MAX_BIENS=25; docker-compose up iad-scraper-custom

# Recherche à Strasbourg, 40 biens (Linux/Ubuntu)
LOCALISATION="Strasbourg" MAX_BIENS=40 docker-compose up iad-scraper-custom

# Mode démon (exécution en arrière-plan)
docker-compose up -d iad-scraper-custom

# Grandes métropoles
LOCALISATION="Lyon" MAX_BIENS=100 docker-compose up iad-scraper-custom
LOCALISATION="Marseille" MAX_BIENS=80 docker-compose up iad-scraper-custom

# Villes moyennes
LOCALISATION="Montpellier" MAX_BIENS=50 docker-compose up iad-scraper-custom
LOCALISATION="Nantes" MAX_BIENS=60 docker-compose up iad-scraper-custom

# Test de développement (limité)
LOCALISATION="Paris" MAX_BIENS=5 MAX_PAGES=1 docker-compose up iad-scraper-custom
```

Compétences C2: Développement de requêtes SQL

`SQL Scripts`

services/c2-sql (stats_views.sql)

Vues SQL d'Analyse Immobilière Avancée

Les 15 vues SQL constituent un système d'analyse prédéfinie exploitant les fonctionnalités analytiques de PostgreSQL. Chaque vue implémente des agrégations complexes : v_stats_type_bien utilise les fonctions d'agrégation standard (AVG, MIN, MAX) avec regroupement multi-critères, tandis que v_stats_distribution_surfaces emploie PERCENTILE_CONT(0.5) pour calculer les médianes robustes aux valeurs extrêmes.

Les vues temporelles comme v_stats_prix_m2_par_mois exploitent DATE_TRUNC('month') pour l'analyse de tendances saisonnières, avec partitionnement par type de bien. Les analyses DPE (v_stats_performance_energetique) utilisent des jointures optimisées entre les tables annonces et dpe, avec tri sémantique via CASE pour ordonner naturellement les classes énergétiques.

Les vues de distribution comme v_stats_caracteristiques_populaires calculent des pourcentages relatifs avec sous-requêtes corrélées, tandis que v_stats_rapport_surface_prix implémente des ratios prix/surface avec gestion des divisions par zéro via NULLIF. Les catégorisations avancées dans v_stats_annee_construction et v_stats_coproprietes utilisent des plages dynamiques avec CASE pour segmenter les données en cohorts pertinentes.

La vue synthèse v_synthese_generale agrège des métriques cross-tables via sous-requêtes scalaires, offrant un dashboard performance en une seule requête. L'ensemble utilise systématiquement le typage explicite (::numeric) et le tri sémantique pour une exploitation analytique immédiate.

Extrait de scrips:

```sql
-- Segmentation dynamique des prix avec ranking et analyse comparative
WITH ranked_annonces AS (
    SELECT
        localisation,
        type_bien,
        prix,
        prix_au_m2,
        ROUND(AVG(prix_au_m2) OVER (
            PARTITION BY localisation, type_bien
        )::numeric, 2) as prix_m2_moyen_zone,
        ROUND((prix_au_m2 / NULLIF(AVG(prix_au_m2) OVER (
            PARTITION BY localisation, type_bien
        ), 0) - 1) * 100, 2) as ecart_percent_moyen,
        NTILE(4) OVER (
            PARTITION BY localisation, type_bien
            ORDER BY prix_au_m2
        ) as quartile_prix,
        RANK() OVER (
            PARTITION BY localisation
            ORDER BY prix_au_m2 DESC
        ) as rank_prix_dans_commune
    FROM annonces
    WHERE prix_au_m2 IS NOT NULL
      AND localisation IS NOT NULL
      AND type_bien IS NOT NULL
)
SELECT *
FROM ranked_annonces
WHERE rank_prix_dans_commune <= 10  -- Top 10 par commune
ORDER BY localisation, type_bien, prix_au_m2 DESC;
```

SQL — Showing rows: 1 to 45

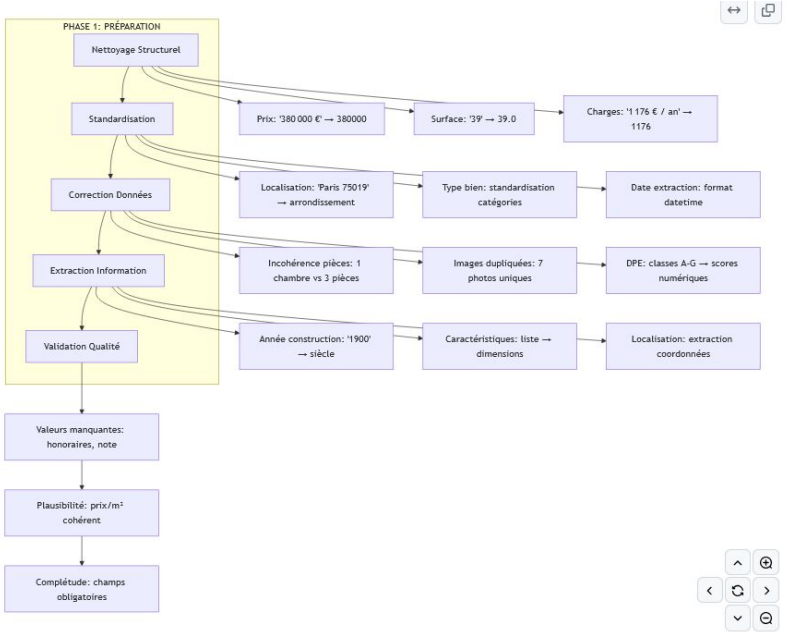
	localisation character varying (255)	type_bien character varying (100)	prix numeric (12,2)	prix_au_m2 numeric (10,2)	prix_m2_moyen_zone numeric	ecart_percent_moyen numeric	quartile_prix integer	rank_prix_dans_commune bigint
15	Montpellier (34070)	Appartement	340000.00	4000.00	3301.02	21.17	4	5
16	Montpellier (34070)	Appartement	379500.00	3872.45	3301.02	17.31	4	7
17	Montpellier (34070)	Appartement	169000.00	3840.91	3301.02	16.36	4	8
18	Montpellier (34070)	Appartement	199000.00	3754.72	3301.02	13.74	4	10
19	Montpellier (34070)	Duplex	299000.00	3883.12	3883.12	0.00	1	6
20	Montpellier (34070)	Maison	258000.00	4300.00	4076.97	5.47	4	2
21	Montpellier (34070)	Maison	519000.00	4086.61	4076.97	0.24	2	3
22	Montpellier (34070)	Maison	519000.00	4086.61	4076.97	0.24	3	3

Compétences C3: Agrégation et préparation des données

`Data Preparation (Data Cleaning, Data Integration, Data Transformation, Data Reduction) & Data Aggrégation`

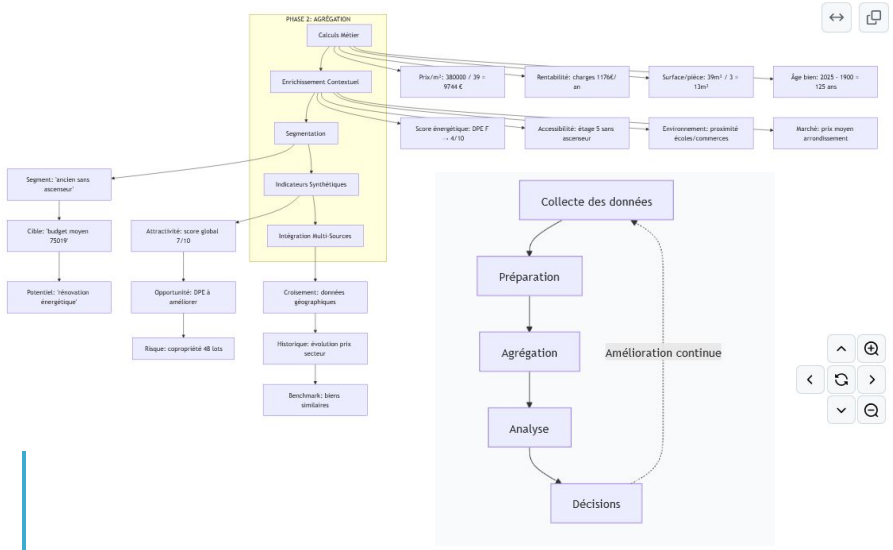
services/c3_aggr (README.md, README.preparation, README.agregation)

Préparation des données (Data Préparation)



La data preparation est le processus crucial de nettoyage, de transformation et d'enrichissement des données brutes pour les rendre exploitables pour l'analyse.

Agrégation des données (Data Aggrégation)



L'agrégation de données est le processus qui consiste à collecter et rassembler des informations dispersées pour les synthétiser en ensembles cohérents, permettant une analyse globale et une prise de décision éclairée

Compétence C4 : Création de Base de Données

`Create and Ingestion in Database`

services/c4-create_db

J'ai conçu et implémenté une base de données PostgreSQL complète pour la gestion d'annonces immobilières, en partant d'un modèle UML jusqu'au déploiement opérationnel. Le schéma relationnel intègre six tables principales (annonces, caractéristiques, images, conseiller, DPE, copropriété) avec leurs relations, contraintes d'intégrité et index optimisés. Des vues matérialisées permettent des analyses statistiques sur les prix, surfaces et performances énergétiques.

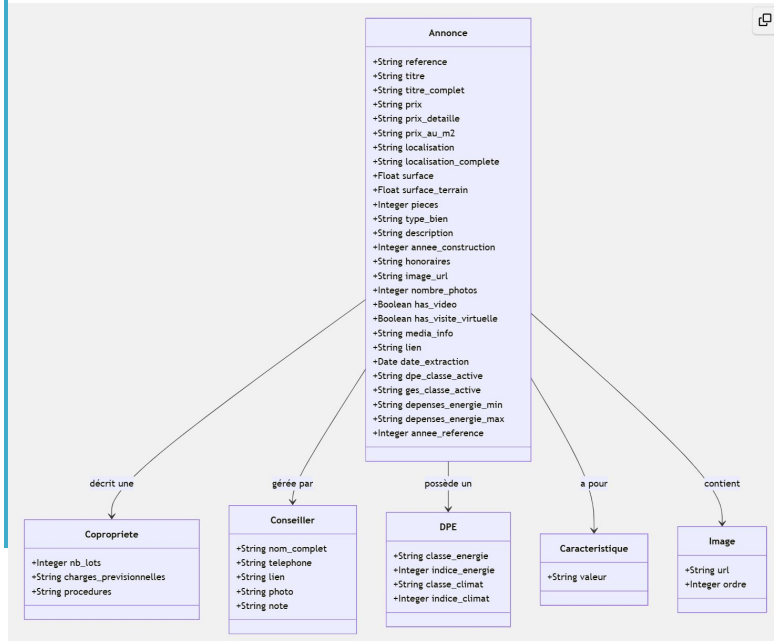
Un script Python robuste a été développé pour l'import et le nettoyage des données JSON, incluant la validation des prix, l'extraction des surfaces, la gestion des diagnostics DPE et le traitement des relations un-à-plusieurs. Le système gère automatiquement les doublons, les erreurs d'import et fournit des statistiques en temps réel, formant une solution ETL complète pour l'analyse du marché immobilier.

```
# Creation de la base de données imo_db
psql -U postgres -f sql\create_imo_db_init.sql

# Acces a imo_db et reation de la structure et des données partielles.
psql -U postgres -d imo_db -f sql\create_imo_db.sql

# Import des données dans la base imo_db a partir du fichier json
uv run import_annonces.py
```

Modèle UML de la base de données imo_sql



Compétence C5: Développement d'une API Rest

`a APi Rest with Fast API from Postgres Datasource`

services/c5-api

API Rest avec FASTAPI

Cette API Immobilière est développée avec FastAPI, un framework web moderne et haute performance pour Python 3.8+. L'architecture repose sur une base de données PostgreSQL avec une connexion gérée via psycopg2 et le curseur RealDictCursor pour un mapping naturel des résultats en dictionnaires Python.

L'application implémente le pattern Dependency Injection de FastAPI pour la gestion des connexions à la base de données, assurant une ouverture/fermeture propre des ressources à chaque requête.

Le système inclut un middleware CORS configuré pour autoriser les requêtes cross-origin, rendant l'API consommable par des applications frontend. La validation des paramètres de requête est assurée par le système de types de Pydantic intégré à FastAPI, avec des contraintes sur les valeurs numériques (bornes min/max) et la gestion des paramètres optionnels. L'API expose des endpoints RESTful conventionnels avec pagination, filtrage avancé (par type de bien, prix, surface, localisation) et recherche plein texte via des clauses ILIKE PostgreSQL.

La documentation interactive est automatiquement générée via Swagger UI et ReDoc, tandis que la configuration est externalisée via python-dotenv pour une gestion sécurisée des credentials. Les erreurs sont uniformément gérées avec des HTTPException standardisées et des codes d'état appropriés, garantissant une expérience client prédictible.

Endpoints

GET	`/`	Root
GET	`/annonces`	Get Annonces
GET	`/annonces/{annonce_id}`	Get Annonce By Id
GET	`/annonces/reference/{reference}`	Get Annonce By Reference
GET	`/annonces/search`	Search Annonces
GET	`/statistiques`	Get Statistiques
GET	`/types-bien`	Get Types Bien
GET	`/localisations`	Get Localisations

```
uv run api.py
```

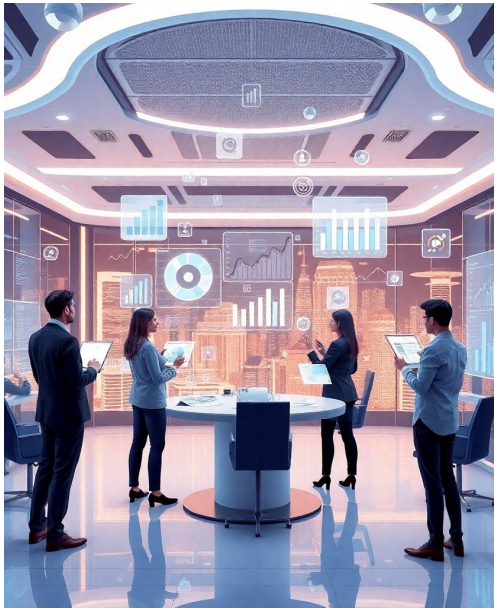
Conclusion

Ce projet de plateforme data immobilière IMO-OPS a permis la mise en œuvre d'une architecture complète et conteneurisée, orchestrant l'ensemble du cycle de vie des données.

L'infrastructure technique repose sur un écosystème Docker composé d'Airflow pour l'orchestration des pipelines, Redis est envidagée pour la gestion des files, PostgreSQL et MongoDB pour le stockage, ainsi que Grafana et des interfaces personnalisées pour la visualisation.

Cette stack technique robuste supporte une valeur métier significative, avec des calculs avancés de rentabilité, une segmentation fine du marché et des recommandations actionnables.

L'industrialisation sera garantie par l'automatisation des processus via les DAGs, la reproductibilité des environnements et le monitoring unifié, faisant de cette plateforme une solution opérationnelle et maintenable pour l'analyse du marché immobilier



`imo_ops`

Annexes

`Documentation`