Rendu 1 : Exploration et Collecte de Données

1. Introduction

Ce premier rendu présente la **phase d'exploration et de collecte** pour le projet de matching candidats-offres. Il illustre la maîtrise du cycle de vie Data Engineering : identification des sources, prototypage API, automatisation ETL, normalisation et contrôle qualité.

1.1 Objectifs

- **Business** : alimenter une base d'offres d'emploi et d'informations entreprises pour analyser – secteurs porteurs, compétences recherchées, géographies actives.
- **Technique** : démontrer la robustesse d'un pipeline de collecte : authentification, pagination, gestion d'erreurs, consolidation de schéma.

2. Sources de données

2.1 France Travail (API REST)

- **Endpoints testés** : recherche d'offres, marché du travail, codes ROME, compétences métiers, données territoriales.
- **Paramètres explorés** : date de publication, thème, code ROME, pageSize & page.
- **Authentification** : token OAuth2 géré automatiquement (rafraîchi toutes les 20 minutes).

2.2 Adzuna (API REST)

- Substitution d'Indeed (accès payant) par Adzuna, offrant un endpoint unique de recherche d'offres.

3. Prototypage & tests Postman

- **Workspaces partagés** : centralisation des requêtes GET, gestion de variables globales (URL, clés, tokens).
- **Validation des payloads** : inspection des structures JSON, test de tous les paramètres de requête avant codage.

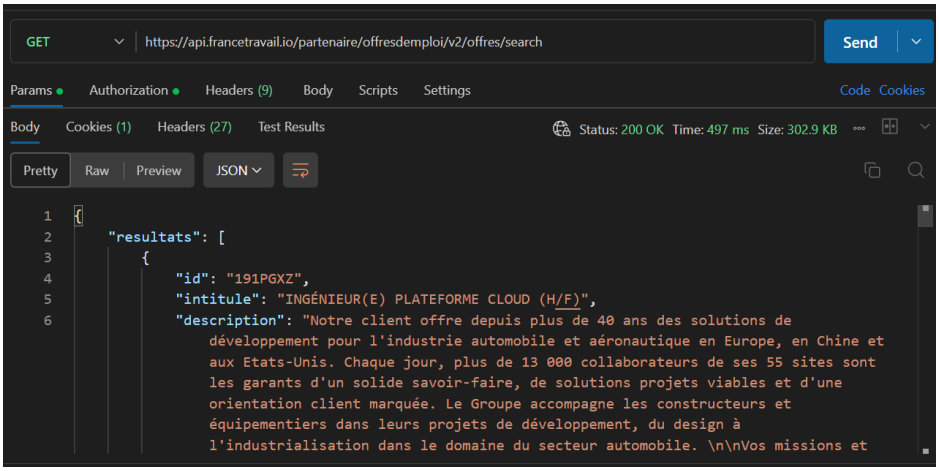


Image 1 : Requête Postman « offres d'emploi France Travail »

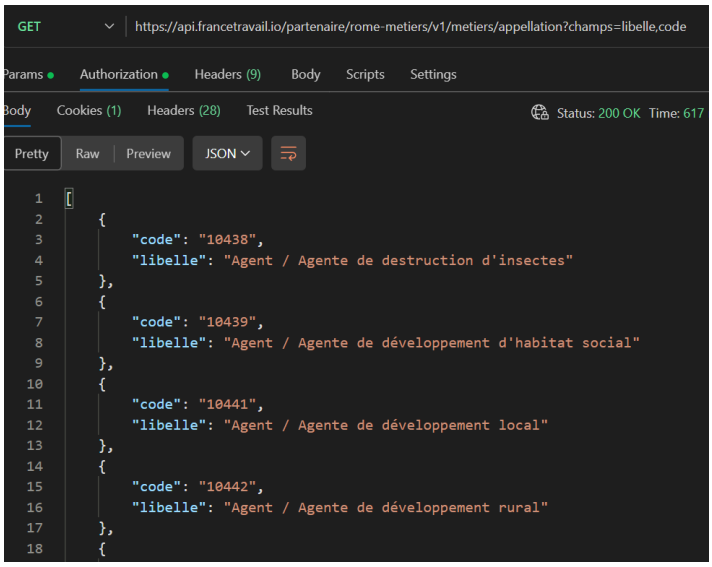


Image 2 : Requête Postman « liste CodeRome » pour l'API France Travail

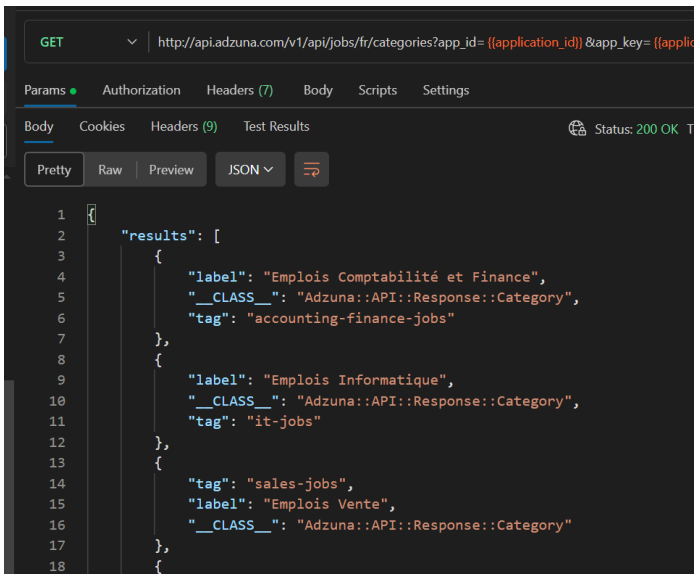


Image 3 : Requête Postman « liste Catégories » pour l'API Adzuna

4. Pipeline Collector (Python)

4.1 Architecture orientée classes

- BaseCollector : interface commune (authenticate(), fetch_offers(), paginate(), refresh_token()).
- FranceTravailCollector & AdzunaCollector héritent et implémentent les différences d'authentification et endpoints.

```python
class FranceTravailAPI:
    """
    Requêteur automatique pour l'API Offres d'emploi de France Travail
    """
    BASE_URL = "https://api.francetravail.io/partenaire/offresdemploi"
    TOKEN_URL = "https://entreprise.francetravail.fr/connexion/oauth2/access_token?realm=/partenaire"
    VERSION = "v2"

    def __init__(self, client_id: str, client_secret: str):
        """
        Initialise le requêteur avec un token d'authentification

        Args:
            client_id: Votre identifiant d'application France Travail
            client_secret: Votre clé d'API France Travail
        """
        token = self.get_token(client_id, client_secret)
        self.token = token
        self.headers = {
            "Authorization": f"Bearer {token}",
            "Accept": "application/json"
        }
```

Image 4 : Script python France travail pour refresh le token

Sur le collector, nous avions pour objectif de n'avoir que des offres liées à la data. Nous avons donc cherché les paramètres d'entrées sur l'API France travail et Adzuna qui pourraient faire office de filtrage. Finalement, pour France Travail, ce sera le paramètre « code Rome» et pour Adzuna la catégorie.

```python
# Liste des catégories Adzuna à collecter
ADZUNA_CATEGORIES = [
    "it-jobs",           # Informatique
    "engineering-jobs",  # Ingénierie
    "scientific-qa-jobs", # Scientifique et QA
    "creative-design-jobs", # Créatif et Design
    "consultancy-jobs",  # Conseil
    "graduate-jobs"      # Jeunes diplômés
]
```

Image 5 : Identification des catégories de l'API Adzuna liés à la data

```
# Liste des codes ROME liés à l'informatique, la data et l'IT
CODES_ROME_IT = [
    # 1. Développement et Programmation
    "M1805",  # Développeur / Développeuse web
    "M1806",  # Développeur / Développeuse multimédia
    "M1802",  # Analyste Concepteur / Conceptrice informatique
    "M1801",  # Analyste d'étude informatique

    # 2. Ingénierie et Architecture
    "M1807",  # Ingénieur / Ingénieure d'étude informatique
    "M1808",  # Ingénieur / Ingénieure systèmes et réseaux informatiques
    "M1809",  # Ingénieur / Ingénieure concepteur / conceptrice informatique
    "M1810",  # Architecte systèmes et réseaux des territoires connectés
    "M1811",  # Architecte cloud
    "M1812",  # Architecte IoT - Internet des Objets

    # 3. Administration et Sécurité
    "M1813",  # Administrateur / Administratrice réseau informatique
    "M1814",  # Administrateur / Administratrice sécurité informatique
```

Image 6 : Identification des codes Rome de l'API France Travail liés à la data

4.2 Pagination robuste

- Boucle sur page jusqu'à épuisement des résultats (présence de nextPage ou limite pageSize).
- Gestion des erreurs HTTP (429 Rate Limit, 500 Server Error) avec stratégie de retry et backoff exponentiel.

```
    # Boucler sur les catégories Adzuna
    for category in ADZUNA_CATEGORIES:
        logger.info(f"Récupération des offres pour la catégorie {category}")

        # Paramètres de pagination
        page = 1
        max_pages = 20  # Limite à 20 pages
        results_per_page = 50  # Nombre d'offres par page
```

Image 7 : intégrer une pagination pour éviter les erreurs

5. Normalisation & Consolidation

- **Pydantic** : définition de OfferSchema pour valider et typer chaque réponse JSON.

```python
class NormalizedJobOffer(BaseModel):
    """Modèle d'offre d'emploi normalisé pour les deux sources"""
    Ctrl+L to chat, Ctrl+K to generate
    id: str
    source: str  # 'adzuna' ou 'france_travail'
    title: str
    description: Optional[str] = None
    company_name: Optional[str] = None
    location_name: Optional[str] = None
    latitude: Optional[float] = None
    longitude: Optional[float] = None
    date_created: Optional[datetime] = None
    date_updated: Optional[datetime] = None
    contract_type: Optional[str] = None  # CDI, CDD, etc.
    contract_duration: Optional[str] = None
    working_hours: Optional[str] = None  # Temps plein, temps partiel
    salary_min: Optional[float] = None
    salary_max: Optional[float] = None
    salary_currency: Optional[str] = None
```

Image 8 : Utilisation class Pyandtic pour normaliser les colonnes

- **DataFrame unifié** : extraction des champs clés : offer_id, title, company, location, date_posted, description, category, code_rome.
- **Mapping métier** : transformation des statuts (permanent → CDI, contract → CDD), uniformisation des formats de date.

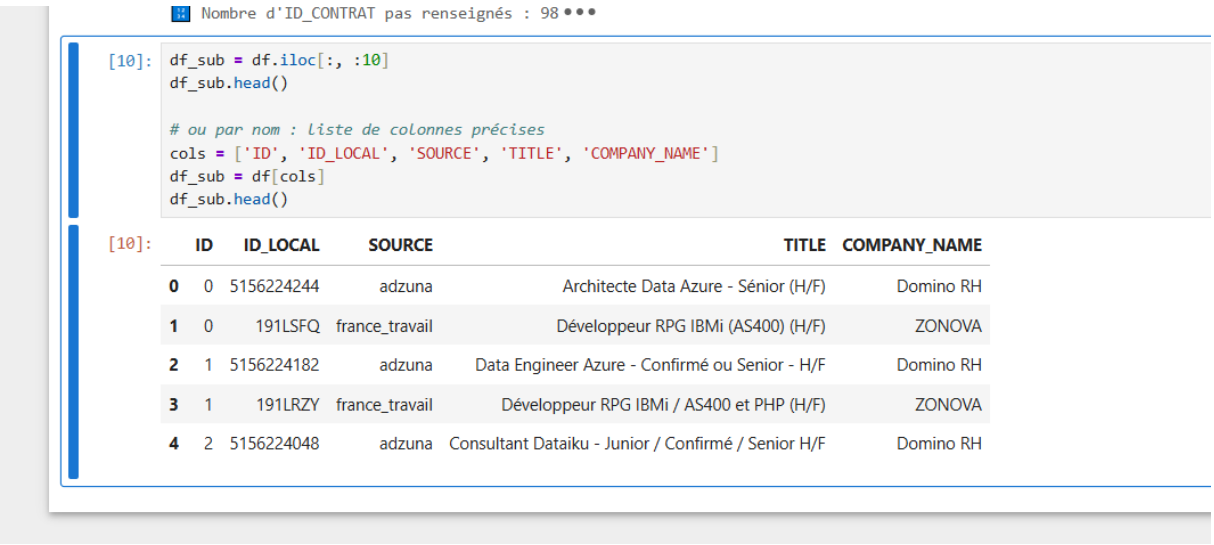
```python
    # Ces mappings vont aider à normaliser les données entre les sources
    self.contract_type_mapping = {
        # Adzuna
        "permanent": "CDI",
        "contract": "CDD",
        # Ces valeurs sont déjà conformes pour France Travail
        "CDI": "CDI",
        "CDD": "CDD",
        "MIS": "Mission intérimaire",
        "SAI": "Travail saisonnier",
        "LIB": "Profession libérale",
    }

    self.working_hours_mapping = {
        # Adzuna
        "full_time": "Temps plein",
        "part_time": "Temps partiel",
        # France Travail
        "Temps plein": "Temps plein",
        "Temps partiel": "Temps partiel",
```

Output Debug Console Terminal Ports 25 ... zsh - mar25_bootcamp_de_job_market

Image 9 : Exemple de mapping, pour normalisation des valeures entre France Travail et Adzuna

6. Analyse exploratoire préliminaire

- **Qualité** : df.isnull().mean() pour chaque colonne, détection de colonnes très incomplètes.
- **Synthèses** : histogrammes de répartition par code ROME, barplots de fréquences par ville.
- **Anomalies** : offres sans localisation, descriptions manquantes, doublons identifiés et filtrés.



	ID	ID_LOCAL	SOURCE	TITLE	COMPANY_NAME
0	0	5156224244	adzuna	Architecte Data Azure - Sénior (H/F)	Domino RH
1	0	191LSFQ	france_travail	Développeur RPG IBMi (AS400) (H/F)	ZONOVA
2	1	5156224182	adzuna	Data Engineer Azure - Confirmé ou Senior - H/F	Domino RH
3	1	191LRZY	france_travail	Développeur RPG IBMi / AS400 et PHP (H/F)	ZONOVA
4	2	5156224048	adzuna	Consultant Dataiku - Junior / Confirmé / Senior H/F	Domino RH

Image 10 : Aperçu du DataFrame de concaténation France Travail et Adzuna

7. Architecture cible & prochaines étapes

- **Modélisation UML** : tables offers, companies, locations, categories, liaisons en schéma en étoile.
- **Data Warehouse** : implémentation sur Snowflake, vues matérialisées pour reporting BI.
- **Orchestration** : automatiser l'ingestion et la transformation via Airflow & dbt, supervision continue.

8. Conclusion

Cette première étape capitalise sur un pipeline programmatique et validé, garantissant la fiabilité et la scalabilité de la collecte. Elle pose les bases d'analyses avancées et d'intégrations ML pour prédiction et matching.
