

Rapport étape 1

Découverte des sources de données disponibles

- Définir le contexte et le périmètre du projet (ne sous-estimez pas cette étape)
- Prise en main des différentes sources de données (explorer les API fournies mais qui vous sont disponibles, les pages webs dont vous allez appliquer le web scraping)
- Livrable attendu : le présent rapport expliquant les différentes sources de données accompagné des exemples de données collectées

Contexte et périmètre du projet.....	2
Prise en main des différentes sources de données.....	5
Web scraping de Welcome to the Jungle par Nam.....	6
Connexion à l'API de Linkedin et Glassdoor par Jean.....	11
Connexion à l'API d'Adzuna et observation sur l'API The Muse par Elsa.....	12

Contexte et périmètre du projet

Plusieurs réunions en visioconférence nous ont permis des phases de travail commun et d'aligner les objectifs des temps de travail individuel :

- lundi 05 février à 10h pour faire connaissance entre Elsa, Jean et Nam
- jeudi 08 février à 12h la première réunion avec Antoine ou l'étape 0 :
 - introduction de chaque membre
 - présentation du cadre du projet et ses 6 grandes étapes :
 - Étape 0 / Cadrage (Notre première réunion)
Introduction de chaque membre de l'équipe
 - Explication du cadre du projet (les différentes étapes)
 - Étape 1 / Découverte des sources de données disponibles
 - Étape 2 / Organisation des données : Deadline 24 Février 2024
 - Étape 3 / Consommation des données : Deadline 15 Mars 2024
 - Étape 4/ Déploiement: Deadline 29 Mars 2024
 - Étape 5 / Automatisation des flux (ÉTAPE FACULTATIVE)
 - Étape 6 / Démonstration de l'application + Soutenances (30 minutes): prévu mardi 9 Avril 2024
 - création par Antoine du repository partagé pour le code du projet
<https://github.com/DataScientest-Studio/JANV24-BDE-JOBMARKET>.
 - mardi 13 février à 11h entre Elsa, Jean et Nam avec comme objectif la définition du contexte et du périmètre du projet.
À l'issue de notre échange, nous avons déterminé de :
 - concentrer notre recherche sur les offres d'emploi disponibles en **France** (limite géographique)
 - concentrer notre recherche sur les offres d'emploi dont le **titre** contient précisément les **deux termes “Data” et “Engineer”** (limite lexicale).
À noter que selon le temps restant à la fin de notre projet, nous envisageons de potentiellement élargir cette limite pour incorporer des termes similaires (par exemple Ingénieur de données - Développeur Data - Ingénieur Big Data)
 - créer une **base de données relationnelles** avec les champs suivants : [intitulé, entreprise, salaire, présentation entreprise, présentation poste, technico, contact entreprise, type de contrat (CDI, freelance, alternance, stage) et lieu]. Ces champs sont amenés à être complétés selon les informations finalement obtenues.
 - nous concentrer sur le conseil d'Antoine pour réaliser un **dashboard fonctionnel** (= la partie Machine Learning n'est pas une priorité) et voir selon le temps restant quelles fonctionnalités supplémentaires mais non essentielles nous pourrions ajouter

- **travailler en parallèle tous les 3** sur chaque étape pour que nous puissions tous monter en compétence (contrairement à avoir chacun un rôle différent) et être en mesure de connaître tout notre code et d'être confiant et à l'aise le jour de la soutenance
 - **répartir les sources de données entre nous 3** : Welcome To The Jungle pour Nam, Glassdoor & Linkedin pour Jean et les API Azuna / The Muse pour Elsa
 - utiliser git au mieux des **bonnes pratiques** (= avec utilisation de branches ...)
 - de **coder en anglais** (nom de variables et documentation)
 - suivre l'avis de Nam quant à l'utilisation de **Streamlit** (vs Dash). Il a déjà utilisé les deux et juge le premier plus facile à prendre en main
 - viser un dashboard interactif avec la possibilité selon notre récolte de données bien sûr, de pouvoir sélectionner des technologies ou outils spécifiques pour faire apparaître les offres directement liées
 - partager dans un google slide le travail préparatoire sur chaque étape en amont de la rédaction de chaque rapport comme livrable.
-
- vendredi 16 février à 12h avec Antoine à qui nous avons présenté les grandes lignes issues de notre réunion à 3, le notebook déjà finalisé du travail d'extraction de Nam et les explorations préliminaires des sources de Jean et Elsa.
Antoine a validé les points suivants
 - le **choix d'une base de données relationnelles**, concaténant toute les données des différentes sources dans un format unique
[l'alternative aurait été par exemple une base de donnée NoSQL avec un certain type pour chacune des sources mais cela ne nous aurait pas permis la même qualité d'analyses comparatives entre les sources]
 - la **taille de celle-ci** (estimée alors à ~5000 lignes) qui s'explique par le choix du périmètre avec la limite géographique et lexicale. Nous cherchons les offres d'emploi avec le titre data engineer strictement en France.
 - la **forte variabilité sur la disponibilité des attributs pour les offres issues de l'API Adzuna** (un aggrégateur qui redirige vers l'url primaire de l'offre qui présente des formats fortement différents).
Cet aspect de nos données aura un fort impact sur les indicateurs d'analyse que nous choisirons et ça sera une étape du post-processing que nous devrons argumenter.
 - que la **vitesse de la partie récupération** ne pose pas de problème, que pour la partie webscraping cela est compréhensible en raison des pauses imposé au module selenium pour tenter d'imiter le comportement humain
 - la **question du traitement des potentiels doublons**, car nous récupérerons les données de 2 agrégateurs, il existe 2 possibilités :
 - soit on fusionne les doublons que nous repérons
 - soit on labellise selon la source pour analyser les différences entre les deux sources (par exemple WTTJ est plus précis sur les salaires ou au contraire Adzuna fournit cette information 80% fois plus que WTTJ)

Nous choisissons de retenir la deuxième technique et ajoutons une colonne source à notre future base de données

- le besoin de **mentionner dans le rapport** le parallélisme mis en place par Nam et plus généralement tout autre **choix technique**
- de même de ne pas faire l'impasse sur toutes les **situations bloquantes** (ex Linkedin et Glassdoor dont le scraping est difficile) et quelles sont les décisions qui en découlent (ex techniques de contournement ...)
- les possibilités de configuration du livrable attendu, 2 possibilités
 - soit dès maintenant rédiger davantage pour détailler au maximum et avoir la matière qui composera le document final à rendre en pdf
 - soit compléter les slides en l'état (Antoine est ok pour accepter ce format de livrable), et reporter à la fin la rédaction plus complète du document final

Nous choisissons la première configuration, à savoir la rédaction en parallèle du travail de fond. Avancer sur le projet et le document en simultané nous aide à être le plus précis possible

- que le rendu sera à assurer en **script python** et que nos notebooks sont donc à transformer, ce sur quoi nous nous étions déjà entendu.
- mercredi 21 février à 14h entre Elsa, Jean et Nam, l'objectif était de relire tous ensemble le code de Jean et Elsa avant de concaténer les fichier CSV pour créer la base de données. Nam avait finalisé son web scraping de Welcome to the jungle et partagé son travail sur le repository github le jeudi 15 (pour plus de détails voir [la partie explicative de l'exploration de cette source](#)).
Elsa n'avait pas réussi à avancer sur son code et n'avait donc pas finaliser le fichier CSV résultant des requêtes API Adzuna et Nam et Jean ont aidé à déboguer son code.
Jean avait des difficultés variées par rapport à Linkedin (pour plus de détails voir [la partie explicative de l'exploration de cette source](#))
- vendredi 23 février à 14h entre Elsa, Jean et Nam pour travailler ensemble sur la demande d'aide d'Elsa qui n'arrivait pas proprement catégorisé la réponse obtenue suite à la requête API (pour plus de détails voir [la partie explicative de l'exploration de cette source](#))
- lundi 26 février à ? h entre Elsa, Jean et Nam pour relire ce rapport puis l'envoyer à Antoine et créer la bdd

Prise en main des différentes sources de données

Les sources de données mentionnées dans la fiche projet sont :

- API The Muse ([lien](#)) : vous devrez vous créerz un compte ; vous trouverez des offres d'emplois ainsi que des informations sur les compagnies
- API Adzuna ([lien](#)) : clé API requise ; vous trouverez des offres d'emplois, des données historiques sur les salaires
- Scraper Welcome To The Jungle (démo [ici](#)), Indeed, etc ...

Nous avons décidé de travailler en parallèle afin que chacun s'exerce et se familiarise avec une source précise.

Cette partie sera donc divisée selon la source étudiée de la manière suivante :

- Welcome to the Jungle par Nam
- Linkedin et GlassDoor par Jean
- Adzuna et The Muse par Elsa

Web scraping de Welcome to the Jungle par Nam

1. Récupération les liens des offres d'emploi:

Récupération des offres disponibles sur ce [lien](#)

The screenshot shows the homepage of Welcome to the Jungle. At the top, there is a navigation bar with links to Accueil, Trouver un job (which is underlined), Trouver une entreprise, Média, a search bar, and buttons for Employeurs and Se connecter.

In the center, there is a search bar with the placeholder "Cherchez un job, une entrepr...", a location filter set to France, and a dropdown for Type de job. Below the search bar are filters for Télétravail, Professions, Secteur, and a prominent "Tous les filtres" button.

Below the search area, there is a section titled "Jobs" with a count of 329. It includes a "Rechercher uniquement dans le titre de l'offre" checkbox, a "Créer une alerte" button, and a "Trier par pertinence" dropdown. To the right, there are icons for printing and sharing.

Three job cards are displayed:

- LESAFFRE** **Data Engineer** (highlighted in yellow) located in Marcq En Baroeul, CDI, posted yesterday.
- MP DATA** **Data Engineer - Transport** located in Boulogne-Billancourt, CDI, Télétravail fréquent, posted yesterday.
- SKILLS** **Data Engineer Spark Scala H/F** located in Suresnes - Suresnes, CDI, Télétravail fréquent, posted yesterday.

a. Utilisation de la librairie BeautifulSoup

Les offres d'emploi se trouvent dans les balises en HTML et ne s'affichent pas dans la réponse de BeautifulSoup. Cela est probablement dû au fait que le contenu est chargé dynamiquement avec JavaScript. BeautifulSoup ne peut pas exécuter le JavaScript ; il se limite à analyser le HTML statique retourné par la requête HTTP GET initiale. Pour extraire le contenu chargé dynamiquement, il est nécessaire d'utiliser Selenium ou un autre outil capable d'automatiser un navigateur web. Cela permettra d'interagir avec la page web comme le ferait un utilisateur, y compris attendre que le JavaScript soit chargé.

b. Utilisation de la librairie Selenium

Comment mentionné ci-dessus Beautiful Soup ne fonctionne pas car la liste des offres et leur contenu sont chargés via du JavaScript.

Suite à l'inspection du site, chaque offres se trouvent dans la balise `` et la class `.ais-Hits-list-item`

The screenshot shows the Chrome DevTools Elements tab with the DOM tree. A list item `` is selected, which contains an anchor tag ``.

Avec Selenium on peut attendre que l'ensemble des offres soit chargé avec **WebDriverWait**, **EC.presence_of_all_elements_located**

On récupère les contenus avec la méthode **.find_elements**.

Pour cibler un objet HTML ici on utilise **By.CSS_SELECTOR** et **By.TAG_NAME**.

```
WebDriverWait(driver, 15).until(
    EC.presence_of_all_elements_located((By.CSS_SELECTOR, ".ais-Hits-list-item"))
)
job_offers = driver.find_elements(By.CSS_SELECTOR, ".ais-Hits-list-item")

for job in job_offers:
    try:
        link = job.find_element(By.TAG_NAME, "a").get_attribute('href')
        links.append(link)
    except NoSuchElementException:
        print("Failed to find link for one of the job offers.")
```

Le but est de récupérer le lien de l'offre dans le `href` de la balise `<a>`

The screenshot shows the Chrome DevTools Elements tab with the DOM tree. An anchor tag `` is highlighted.

2. Récupération des données de chaque offres

- Repérage de la structure de la page web pour la récupération des données

Identification de l'intitulé du poste, la location, le salaire, le télétravail, expérience, etc.



LESAFFRE

Data Engineer

CDI | Marcq En Baroeul | Salaire : Non spécifié | Télétravail occasionnel | Expérience : > 5 ans

[Postuler](#) [Sauvegarder](#)

avant-hier [Partager](#)

La description du profil recherché via l'id job-section-experience

flex == \$0

Endroit pour récupérer les informations de l'entreprise

L'entreprise

 LESAFFRE

Pharmaceutique / Biotechnologique, Agroalimentaire...

11000 collaborateurs • Crée en 1853

Âge moyen : 41 ans

Chiffre d'affaires : 2.7 milliards

Voir le site • Voir toutes les offres 95

L'informations à récupérer: le secteur, le nombre de collaborateurs, l'année de création, l'âge moyen, le chiffre d'affaires.

Les informations de l'entreprise ont un point commun : elles se trouvent toutes dans la même classe **.sc-dQEtJz.kiMwl**

```

▼ <div class="sc-bXCLTC igTTNe" (flex)
  ▼ <div data-testid="job-company-tag" variant="default" px="sm" py="xs" class="sc-dQEtJz kiMwl" style="flex: 1;">
    > <i class="sc-fUBkdm gSNuBS wui-icon-font" name="tag" style="flex: 1;">

```

- Création de 2 classes JobOffer et Company pour structurer les données

JobOffer: title, company, contract_type, location, salary, remote_type, starting_date, require_experience, education, description, profil_experience, url_direct_offer, publication_date
Company: name, sector, employees, creation_year, turnover, mean_age

```
class Company:
    def __init__(self):
        self.name = None
        self.sector = None
        self.employees = None
        self.creation_year = None
        self.turnover = None
        self.mean_age = None

class JobOffer:
    def __init__(self):
        self.title = None
        self.company = None
        self.contract_type = None
        self.location = None
        self.salary = None
        self.remote_type = None
        self.starting_date = None
        self.require_experience = None
        self.education = None
        self.description = None
        self.profil_experience = None
        self.publication_date = None
        self.url_direct_offer = None
```

- Classe **JobScrapper**

- Fonction **scrap_company_info**: récupération des informations de l'entreprise.
- Fonction **get_description**: récupération des informations des descriptions sur le poste et sur le profil demandé.
- Fonction **get_publication_date**: récupération de la date de publication de l'offre.
- Fonction **scrap_job_offer_info**: récupération des informations de l'offre.
- Fonction **driver_get**: appel au moteur selenium pour ouvrir la page web via le lien de l'offre.
- Fonction **scrape_job_details**: Fait appel aux deux fonctions **scrap_company_info** et **scrap_job_offer_info**. Retourne l'objet **job_offer** qui contient les informations de l'offre.

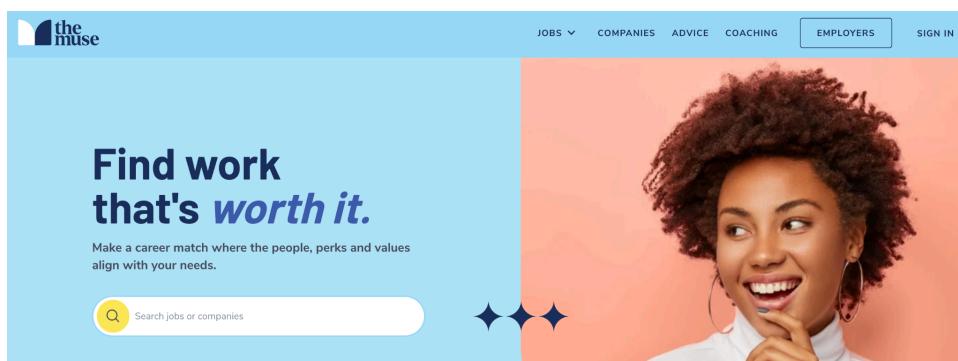
Connexion à l'API de Linkedin et Glassdoor par Jean

Connexion à l'API d'Adzuna et observation sur l'API The Muse par Elsa

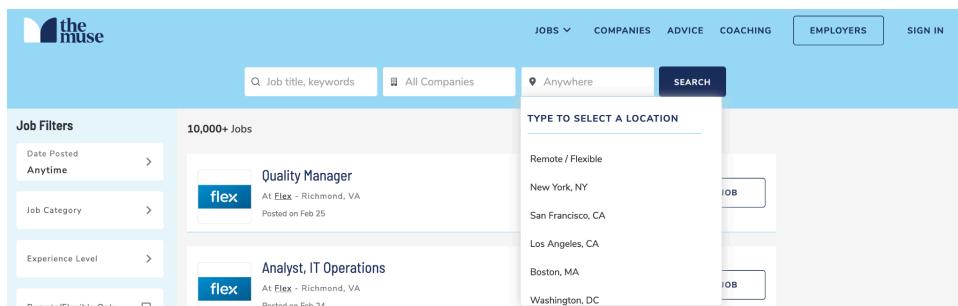
Le premier lien proposé était [l'API de The Muse](#), une plateforme recensant des offres d'emplois dans tous les domaines.

Malgré sa création originale, fondée par deux femmes qui souhaitaient prioriser les valeurs au sein de la culture d'entreprise pour que les employés et employeurs soient le mieux alignés possible, The Muse propose exclusivement des postes basés aux États-Unis ou en télétravail.

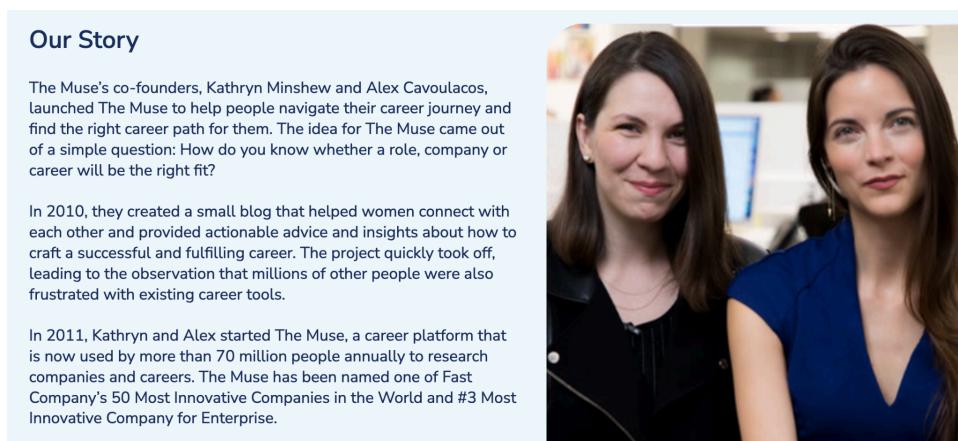
Comme nous choisissons de nous concentrer sur le marché de l'emploi en France, cette source n'a pas été retenue.



The Muse homepage features a large search bar with placeholder text "Search jobs or companies". Below it is a photo of a smiling woman with curly hair. The main tagline reads "Find work that's worth it." with a subtitle "Make a career match where the people, perks and values align with your needs."



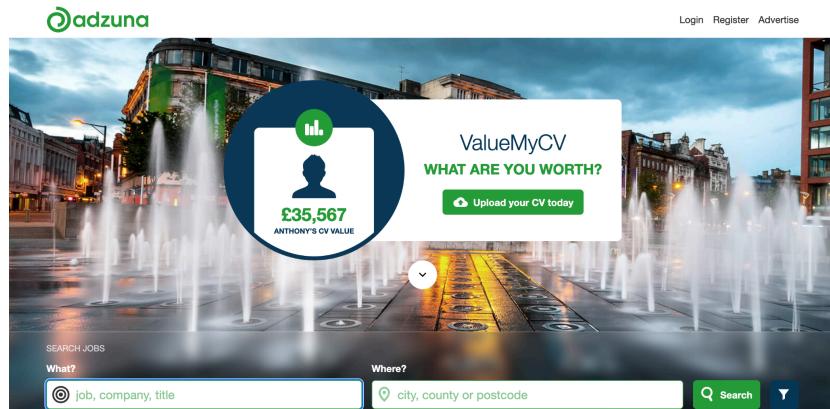
The search results page shows a sidebar with "Job Filters" including Date Posted (Anytime), Job Category, Experience Level, and Remote/Flexible Only. Two job listings are shown: "Quality Manager" at Flex - Richmond, VA and "Analyst, IT Operations" at Flex - Richmond, VA. A dropdown menu for "TYPE TO SELECT A LOCATION" lists cities like New York, NY; San Francisco, CA; Los Angeles, CA; Boston, MA; and Washington, DC.



The "Our Story" page includes a photo of Kathryn Minshew and Alex Cavoulacos. The text on the page discusses how they created a blog in 2010 to help women connect and find career paths, and how The Muse platform was born from that idea.

La deuxième API proposée concernait Adzuna, autre agrégateur d'annonces fondé en 2011 et basé dans un pays anglophone plus proche cette fois, le Royaume-Uni et surtout listant depuis 2014 des offres d'emplois situées en France.

Le nom même semble prometteur, "zuna" signifie abondance dans plusieurs langues africaines et l'objectif d'Adzuna d'être le site Internet qui propose en avant première le plus grand nombre des meilleures offres d'emploi, "*most abundant job ads site on the web, and to bring you the best ads sooner than anyone else*" sont deux bons points de départ pour créer une base de données riches.



Welcome Overview Register Login

Build with Adzuna

Get the very latest ads and data with Adzuna's API. Get job ads to display on your own website. Use Adzuna's up-to-the-minute employment data to power your own website, reporting and data visualisations.

[Read the overview](#)

Get ad listings

Search Adzuna's full listings of job adverts using keywords and locations.

```

    {
        "category": "Software Development",
        "category_id": 1,
        "category_name": "Software Development"
    },
    {
        "category": "Marketing & PR",
        "category_id": 2,
        "category_name": "Marketing & PR"
    },
    {
        "category": "Finance & Banking",
        "category_id": 3,
        "category_name": "Finance & Banking"
    },
    {
        "category": "Retail & E-commerce",
        "category_id": 4,
        "category_name": "Retail & E-commerce"
    },
    {
        "category": "Manufacturing & Engineering",
        "category_id": 5,
        "category_name": "Manufacturing & Engineering"
    },
    {
        "category": "Healthcare & Life Sciences",
        "category_id": 6,
        "category_name": "Healthcare & Life Sciences"
    },
    {
        "category": "Transport & Logistics",
        "category_id": 7,
        "category_name": "Transport & Logistics"
    },
    {
        "category": "Telecommunications",
        "category_id": 8,
        "category_name": "Telecommunications"
    },
    {
        "category": "Automotive & Manufacturing",
        "category_id": 9,
        "category_name": "Automotive & Manufacturing"
    },
    {
        "category": "Food & Beverage",
        "category_id": 10,
        "category_name": "Food & Beverage"
    },
    {
        "category": "Real Estate",
        "category_id": 11,
        "category_name": "Real Estate"
    },
    {
        "category": "Construction & Engineering",
        "category_id": 12,
        "category_name": "Construction & Engineering"
    },
    {
        "category": "Energy & Utilities",
        "category_id": 13,
        "category_name": "Energy & Utilities"
    },
    {
        "category": "Manufacturing & Production",
        "category_id": 14,
        "category_name": "Manufacturing & Production"
    },
    {
        "category": "Retail & Consumer Goods",
        "category_id": 15,
        "category_name": "Retail & Consumer Goods"
    },
    {
        "category": "Manufacturing & Assembly",
        "category_id": 16,
        "category_name": "Manufacturing & Assembly"
    },
    {
        "category": "Manufacturing & Processing",
        "category_id": 17,
        "category_name": "Manufacturing & Processing"
    },
    {
        "category": "Manufacturing & Distribution",
        "category_id": 18,
        "category_name": "Manufacturing & Distribution"
    },
    {
        "category": "Manufacturing & Assembly Services",
        "category_id": 19,
        "category_name": "Manufacturing & Assembly Services"
    },
    {
        "category": "Manufacturing & Processing Services",
        "category_id": 20,
        "category_name": "Manufacturing & Processing Services"
    },
    {
        "category": "Manufacturing & Distribution Services",
        "category_id": 21,
        "category_name": "Manufacturing & Distribution Services"
    }
  
```

Get job data

Get direct access to the latest employment figures.

[Use this data to show off the latest](#)

Jobs in London

Year	Jobs in London (%)
2018	~20%
2019	~25%
2020	~30%
2021	~22%
2022	~20%

Une fois le compte créé et les identifiants, `app_id` et `app_key`, créés, [la documentation interactive](#) a permis de définir les paramètres souhaités pour notre recherche.

```
COUNTRY = "fr"
API_ID = "37fe08af"
API_KEY = "17c1369ea8fb68bc48f834aebc0bec53"
RESULTS_PER_PAGE = str(25)
PAGE_SCRAPED = 100
QUERY_PARAMETERS = "&title_only="
KEYWORDS = "data%20engineer"
MAX_OLD_DAYS = str(30)
```

Le script se décompose alors en 4 fonctions :

- `create_url` pour assembler les paramètres choisis et créer l'URL sur laquelle lancer la requête `HTTP`.
- `scrape_url` pour récupérer via la librairie `requests` les informations contenant l'offre et les renvoyer au format json*
- `combing_each_offer` pour retenir les informations qui constituent l'annonce à proprement parler, non seulement page par page mais également au sein de chaque page si le paramètre `RESULTS_PER_PAGE` autorisent plusieurs offres par page. Ces informations clés alimentent alors les classes spécifiquement créées pour structurer les données.
- `main` pour appeler les 3 fonctions précédentes dans le bon ordre et enregistrer le résultat au format CSV dans un dossier output présent au même niveau que le script.

* ce choix, de format `json`, pourtant évident quand on travaille avec une API qui a justement pour objectif de faciliter le transfert d'information entre deux services ou logiciels, n'a pas été inné pour moi.

En effet, avant d'avoir la relecture de Nam et d'utiliser le décodeur `json` intégré à la librairie `requests`, je récupérais le contenu de la réponse grâce à l'attribut `text` et non la méthode `json`, ce qui me compliquait la catégorisation ultérieure.

Ce qui était d'autant plus frustrant vu qu'Adzuna étant un aggrégateur qui ne présente pas une fiche propre pour chaque annonce, mais redirige systématiquement vers le site primaire de l'annonce, une grande partie des attributs systématiques remplis pour le scraping de WTTJ restaient vides pour moi.

Par exemple, contrairement à la mention de données historiques récupérables sur les salaires, **l'intégralité des offres récupérées ont une valeur nulle pour la clé `salary_is_predicted`** (même lors de tests avec le paramètre `&salary_include_unknown` passé à 1 ou 0 dans l'url)

Par ailleurs, pour parer à de mauvais codes **HTTP** et malgré l'exception levée prévue via la méthode json (`requests.exceptions.JSONDecodeError`), la fonction `scrape_url` fait appel au `HTTPError`.

```
def scrape_url(url):
    """
    Get requests from Adzuna API for each url passed

    Parameters
    -----
    url : str
        the url needed to get the Adzuna API response

    Returns
    -----
    response.json() : json object
        a dictionary with 4 keys ['results', '__CLASS__', 'count', 'mean']
        the actual job offer details are in results

    Raises
    -----
    HTTPError
        If the status code is not 200 (Standard response for successful
        HTTP requests), the incorrect status code and url are displayed
    """
    response = requests.get(url, timeout=500)
    if response.status_code == 200:
        return response.json()
    raise HTTPError(f"The request for {url} was not successful : \
        status {response.raise_for_status()}")
```

Reprises des mêmes attributs que ceux présents dans les **classes** `JobOffer` et `Company` utilisées pour le scraping de Welcome To The Jungle par Nam.

<pre>class JobOffer: """ A class used to represent a job offer Attributes ----- matching the ones collected from the wwtj scraping title : str company : str contract_type : str location : str salary : int remote_type : str starting_date : str required_experience : str education : str description : str profil_experience : str publication_date : str url_direct_offer : str """ def __init__(self): self.title = None self.company = None self.contract_type = None</pre>	<pre>class Company: """ A class used to represent a company Attributes ----- matching the ones collected from the wwtj scraping name : str sector : str employees : int creation_year : int turnover : int mean_age : int """ def __init__(self): self.name = None self.sector = None self.employees = None self.creation_year = None self.turnover = None self.mean_age = None</pre>
--	---

J'ai documenté autant que possible mon code.

Après les difficultés rencontrées sur la partie technique, d'autres questions se sont posées.

Par exemple, vu la "pauvreté" des infos récoltés via Adzuna par rapport à la plateforme de WTTJ qui à l'instar de The Muse fournir une fiche pour chaque entreprise :

- est-ce que ce serait pertinent de quand un certain seuil d'infos manquantes est franchi (genre s'il manque plus de 50% des infos) > on scrape l'url de redirection (qui est forcément plus complète) ?
- et selon la variété des urls de redirection (autre agrégateur, ou site en nom propres) une méthode différente de scraping ?

Nous avons choisi de nous concentrer sur ce que nous avons déjà pu récupérer et d'aviser selon le temps restant une fois que notre application sera entièrement fonctionnelle pour pourquoi pas alimenter à nouveau la base de données.

Avec des critères plus larges, comme un paramètre `what_and` (qui cherche les mots clés dans l'ensemble de l'annonce) au lieu du `title_only` retenu, nous pouvions récupérer jusqu'à 2475 offres (avec 100 urls distinctes et 25 offres par page).

Or en jetant un coup d'oeil sur les dernières lignes du fichier CSV obtenu, il y a des offres avec des intitulés éloigné de notre objectif tel que Ingénieur Modélisation et Gouvernance de la donnée.