Réalisation d’une application de type RAG

**      **

Certification : Développeur d’application en Intelligence Artificiel

Marseille - Nice

Bloc de compétences 1

Réaliser la collecte, le stockage et la mise à disposition des données d’un projet en intelligence artificielle

LOMBARDI Joachim

Table des matières

[Introduction 3](#_Toc192153772)

[A1. Compétence collecte de données 4](#_Toc192153773)

[C1. Extraction des données 4](#_Toc192153774)

[1) Collecte des données 5](#_Toc192153775)

[2) Spécification techniques relatifs à l’extraction des données 6](#_Toc192153776)

[C2. Requêtes SQL 14](#_Toc192153777)

[C3. Nettoyage et agrégation des données 15](#_Toc192153778)

[1) Insertion des données en base 15](#_Toc192153779)

[2) CRUD 17](#_Toc192153780)

[Conclusion 19](#_Toc192153781)

# Introduction

Afin de réaliser mon application de manière sécurisée, efficace et structurée, j’utilise le Framework Django que l’on utilise dans mon entreprise.

Django est un Framework populaire, datant de 2005, qui permet de créer des applications robustes et sécurisées.

Django comprend :

* Un moteur ORM utilisant des objets python plutôt que des requêtes SQL, ce qui simplifie l’interaction avec la base de données.
* Une interface d’administration pour gérer les données. (Notamment le CRUD).
* Un système de routage d'URL séparé de la vue.
* Un système de templates HTML qui permet de séparer la logique métier (dans les vues et modèles) de l’affichage (dans les templates). Il intègre également des variables, des filtres pour transformer les données et des balises pour la logique.
* La sécurité intégrée avec la protection contre les vulnérabilités courantes comme :
* Les attaques XSS (Cross-Site Scripting) : Django échappe automatiquement les variables rendues dans les templates pour empêcher l'exécution de code malveillant.
* Les injections SQL.
* Les attaques CSRF (Cross-Site Request Forgery) ciblent des actions qui modifient les données : Django utilise un système de jetons CSRF pour valider l'origine des requêtes POST, PUT, PATCH et DELETE.
* Le clickjacking : Django inclut un middleware qui ajoute l'en-tête X-Frame-Options pour empêcher l'intégration des pages web dans des cadres (frames).
* Le stockage sécurisé des mots de passe.
* La modularité qui permet de diviser un projet en modules réutilisables.
* Un système de migration de base de données. Les migrations permettent de synchroniser facilement les changements apportés aux modèles Python avec la structure de la base de données.

Le langage utilisé est le Python car c’est celui utilisé pour appli en IA, de plus c’est celui qu’on utilise dans mon entreprise.

L’outil de versioning Git est utilisé pour stocker le script de l’application en ligne.

# A1. Compétence collecte de données

## C1. Extraction des données

Tout média numérique est susceptible d’être utilisé comme source de données, les documents bien évidemment, mais aussi les sites internet, les vidéos, les bandes sonores, les images…

Afin de pouvoir en extraire des données, il faut d’abord connaître les règles de confidentialité.

Ces règles servent à protéger les données personnelles comme :

* Le nom, le prénom, l’adresse, le numéro de téléphone.
* Les informations d'identification (numéro de carte d'identité, numéro de sécurité sociale, etc.).
* Les données numériques : adresses IP, cookies, géolocalisation.
* Les données sensibles : informations de santé, opinions politiques, orientation sexuelle, croyances religieuses, etc.

Selon la CNIL, les 7 grands principes des règles de protection des données personnelles sont les suivants :

**Principe 1** : Licéité, limitation des finalités, minimisation des données

Licéité : Les données personnelles ne doivent être collectées que pour des finalités légitimes, explicites et déterminées.

Limitation des finalités : Les données collectées ne peuvent être utilisées que pour les finalités pour lesquelles elles ont été collectées.

Minimisation des données : La quantité de données collectées doit être réduite au minimum nécessaire pour atteindre les finalités déterminées.

En d'autres termes, les entreprises doivent être transparentes sur la raison pour laquelle elles collectent des données et ne collecter que les informations dont elles ont réellement besoin. Les mentions légales sur votre site internet sont là pour les informer.

**Principe 2** : Exactitude

Les données personnelles doivent être exactes et, si nécessaire, mises à jour. Des mesures doivent être prises pour que les données inexactes soient rectifiées ou effacées.

Cela signifie que les entreprises doivent prendre des mesures pour s'assurer que les données personnelles qu'elles détiennent sont exactes et à jour.

**Principe 3** : Limitation du traitement

Les données personnelles ne doivent être conservées que pendant une durée proportionnelle aux finalités pour lesquelles elles sont traitées.

En d'autres termes, les entreprises doivent avoir une politique claire de conservation des données et supprimer les données lorsqu'elles ne sont plus nécessaires.

**Principe 4** : Intégrité et confidentialité

Les données personnelles doivent être traitées de manière à garantir leur sécurité, y compris la protection contre le traitement non autorisé ou illicite et contre la perte, la destruction ou les dommages accidentels, par des mesures techniques ou organisationnelles appropriées.

Cela signifie que les entreprises doivent mettre en place des mesures de sécurité adéquates pour protéger les données personnelles contre les accès non autorisés, les utilisations abusives et les pertes accidentelles.

**Principe 5** : Responsabilisation

Le responsable du traitement est responsable du respect des principes susmentionnés et doit être en mesure de démontrer sa conformité.

Cela signifie que les entreprises doivent désigner un responsable de la protection des données (DPO) qui sera responsable de la mise en œuvre et du respect du RGPD.

**Principe 6** : Respect des droits des personnes

Les personnes ont le droit d'accéder à leurs données personnelles, de les rectifier, de les effacer, de limiter leur traitement, de s'opposer à leur traitement et de les transférer.

En d'autres termes, les entreprises doivent informer leurs clients de leurs droits en matière de protection des données et leur permettre d'exercer ces droits facilement.

**Principe 7** : Protection des données dès la conception et par défaut :

Le responsable du traitement doit mettre en œuvre des mesures techniques et organisationnelles appropriées pour protéger les données personnelles dès la conception du traitement et tout au long des produits et services.

Cela signifie que les entreprises doivent intégrer le traitement des données dans leur stratégie globale d'entreprise.

Compétence validée : Rédaction ou mise à jour du registre des traitements de données personnelles en vue de la mise en conformité de la base de données avec le RGPD.

Ensuite, il existe également des contraintes lors de l’extraction des données :

### Collecte des données

Comme indiqué dans l’introduction, il existe différents types de données outre les différents médias, un même type de source peut avoir plusieurs supports, par exemple ils peuvent être en format texte, JSON, CSV, base de données relationnelles… Ainsi, les méthodes d’extraction sont différentes et peuvent nécessiter du prétraitement comme pour les PDF afin d’avoir des données propres.

Ces méthodes d’extraction peuvent avoir des limites, comme lors du Scraping (extraction de données depuis une page web) où le nombre de requêtes peut être limité, le téléchargement de fichier là aussi peut être limité ou également l’accès à des bases de données protégées par mot de passe. Il peut également y avoir des problèmes de collecte dû au réseau ou bien à la limitation imposée selon la région du monde où se trouve l’utilisateur ou bien les données.

Certaines données sont très volumineuses et imposent des contraintes au niveau de l’espace de stockage et la mémoire, nécessitant des ressources informatiques importantes et des optimisations de traitement.

Compétence validée : Identification des contraintes techniques propres aux sources de données.

### Spécification techniques relatifs à l’extraction des données

L’objectif du projet est de réaliser une application utilisant les données médicales du site : pubmed.ncbi.nlm.nih.gov afin de poser des questions techniques sur ces données et d’obtenir des réponses en langage naturel ainsi que les sources ayant servis à produire ces réponses.

#### *Pubmed*

PubMed (pubmed.ncbi.nlm.nih.gov) est une ressource utilisée pour la recherche biomédicale et scientifique offrant un accès gratuit à une vaste base de données de références et de résumés d'articles scientifiques. Gérée par la National Library of Medicine (NLM) des États-Unis, cette plateforme regroupe des millions de publications provenant de revues spécialisées, couvrant des domaines tels que la médecine, la biologie, la pharmacologie et bien plus encore.

Cette source de donnée est une des sources de données utilisées par l’entreprise dans laquelle j’effectue mon alternance : Data Observer, car elle contient les publications des professionnels et des groupes de professionnels de santé. Un des principaux objectifs de l’entreprise étant justement d’identifier ces professionnels et ces groupes de professionnels par rapport à leurs publications. Si deux professionnels par exemple on les mêmes noms et prénoms on doit pouvoir identifier qui à produit quoi et pour ça, on utilise leurs affiliations (les établissements dans lesquels ils sont travaillés), les postes qu’ils ont occupés, les villes où ils ont travaillé, les personnes avec qui ils ont travaillés…

J’ai également choisi cette source de donnée car dans l’ensemble elle est bien structurée : les titres, abstracts, auteurs et affiliations… sont très souvent identifié par les mêmes balises html ce qui en facilitent grandement l’extraction. De plus, comme précisé dans le premier paragraphe, il s’agit de résumés ce qui simplifie le traitement des données lorsque l’on va vouloir les interroger.

La veille est effectuée tout au long du rapport.

Compétence validée : Organiser et réaliser une veille technique et réglementaire en animant le travail collectif de sélection des sources, de collecte, de traitement et de partage des informations afin de formuler des recommandations pour le projet toujours en phase avec l’état de l’art.

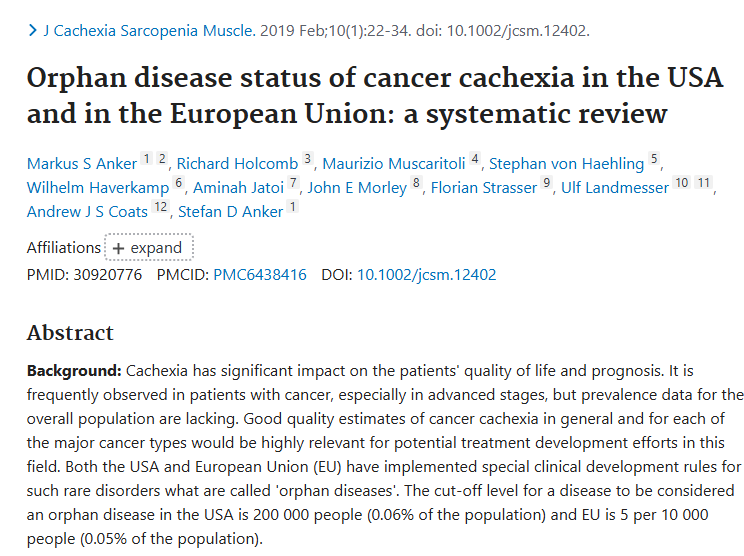


Figure . Extrait d'un article Pubmed : https://pubmed.ncbi.nlm.nih.gov/30920776/

#### Extraction des données pubmed

Pour extraire les données depuis le site PubMed, j’ai identifié trois outils :

* **Scrapy** qui est un Framework complet pour le scraping web.
* **Beautiful Soup** qui est une bibliothèque axée sur la manipulation de l’HTML
* **L’api PubMed** qui permet de récupérer les différentes données, d’effectuer des recherches basées sur des mots-clés, des termes MeSH (Medical Subject Headings), ou des identifiants uniques (comme les PMID). Les résultats sont donnés sous format JSON ou XML.

L’extraction des données d’intérêts étant simple et l’utilisant déjà en entreprise j’ai choisi d’utiliser Beautiful Soup

J’ai choisi d’extraire les données de deux maladies : multiple sclérosis (une maladie qui revient souvent dans les données de l’entreprise) et herpès zoïster (une autre maladie pour alimenter la base de données mais dont le contenu était plus limité) sur la période 2024. Pour ce faire, j’ai scraper les liens de toutes les pages web traitant de ces deux maladies et j’ai scraper le contenu des articles associés à savoir :

* Le titre de la revue dans laquelle un article est publié
* La date de publication de l’article
* Le titre de l’article
* Le résumé de l’article
* L’id pubmed de l’article
* Le doi (Digital Object Identifier) permettant d’identifier et de localiser du contenu numérique.
* Le disclosure indiquant les éventuels conflits d’intérêts.
* Les termes MeSH (Medical Subject Headings) sont un système de vocabulaire contrôlé et hiérarchique développé par la **National Library of Medicine (NLM)** des États-Unis. Ils sont utilisés pour indexer, classer et rechercher des articles scientifiques dans des bases de données biomédicales, comme PubMed et MEDLINE.
* L’URL de l’article.
* Les auteurs de l’article et leurs affiliations.

Pour chaque extraction, on n’extrait les données que si le résumé est présent. Si une donnée est manquante elle prend la valeur None.

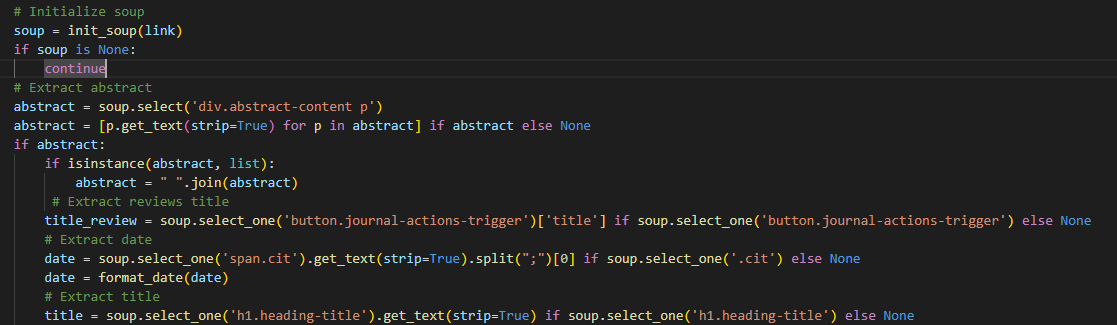


Figure . Extrait du code de scraping

Compétence validée : Exécution programmatique des requêtes d’extraction depuis un système big data.

##### Initialisation de BeautifulSoup

* Envoi d'une requête GET via *requests.get(url).*
* Vérification du succès de la requête :
  + Si *response.status\_code == 200*, le contenu HTML est parsé avec BeautifulSoup ('html.parser').
  + Sinon, la fonction retourne None.
* Retourne un objet soup prêt pour l'extraction des données.

1. Extraction des URLs des articles PubMed

* Construction de l'URL de recherche :
  + *url = base\_url + "/?term=" + term + "&filter=years." + filter + "-2025"*
* Initialisation du parsing avec init\_soup(url).
* Récupération du nombre total de pages
  + Sélection de l'élément *label.of-total-pages* pour obtenir le nombre total de pages de résultats.
  + Si l'élément n'existe pas, la valeur par défaut est 1.
* Boucle de pagination
  + De 1 à *page\_max*, récupération des blocs d'articles *div.search-results-chunk*.
  + Extraction des 10 premiers liens par article avec *article.find\_all('a', href=True)[:10*].
  + Ajout des liens trouvés à links.
  + Pause de 1 seconde (*time.sleep(1)*) pour éviter d'être bloqué par PubMed.
  + Rechargement de la page suivante en modifiant l'URL *(url+"&page="+str(i))* et répétition du processus.
* Retourne la liste des URLs d'articles trouvés.

##### Extraction des informations d’un article

* Initialisation des variables
  + *articles\_data* : Liste pour stocker les articles scrappés.
  + *suffix = term + "\_" + filter* : Permet d’ajouter un suffixe personnalisé au fichier de sortie.
* Récupération des liens :
  + Si url est fourni, utilise la liste existante.
  + Sinon, appelle *extract\_pubmed\_url()* pour récupérer les liens à partir du base\_url.
* Parcours des articles :
  + Pour chaque link, initialise *soup = init\_soup(link).*
  + Si soup est None, passe à l'article suivant.
* Extraction des informations :
  + title\_review : Sélection de *button.journal-actions-trigger['title'].*
  + date :
    - Extraction via *span.cit*.
    - Nettoyage et formatage avec *format\_date(date).*
    - title : Sélection *h1.heading-title*.
  + abstract : Extraction et nettoyage des paragraphes *div.abstract-content p*.
  + pmid : Extraction du PMID via *span.identifier.pubmed strong.current-id.*
  + doi :
  + Extraction de l'identifiant via *span.identifier.doi a.id-link*.
  + Formatage en URL *https://doi.org/ + doi*.
  + disclosure : Déclaration des conflits d'intérêt dans *div.conflict-of-interest div.statement p*.
  + mesh\_terms :
  + Extraction via *button.keyword-actions-trigger.*
  + Conversion en texte *(", ".join(mesh\_terms)).*
  + url : Création du lien absolu via *get\_absolute\_url(pmid).*
* Extraction des auteurs et affiliations :
  + Sélection des auteurs : *soup.select('.authors-list-item').*
  + Filtrage des doublons avec *seen\_authors* (set pour éviter de scraper plusieurs fois le même auteur).
* Récupération des affiliations :
  + Sélection *author\_tag.select('.affiliation-link').*
  + Récupération des noms via *.get('title', None).*
  + Ajout des noms et affiliations dans *authors\_affiliations*.

Compétence validée : Rédaction des spécifications techniques pour l’extraction des données.

Compétence validée : Téléchargement de l’HTML d’une ou plusieurs pages web visées par une action de scraping.

Compétence validée : Documentation des requêtes d’extraction.

Compétence validée : Rédaction des spécifications techniques pour l'agrégation des données.

Compétence validée : Programmation des règles d’agrégation des données collectées depuis chaque source en un jeu de données brutes unique.

Compétence validée : Documentation des scripts.

#### Merise

##### Modèle Conceptuel de Données (MCD)

Article Authors Affiliations

title\_review name name

date

title

abstract

pmid

doi

disclosure

mesh\_terms

url

term

#### **Entités et Associations**

* **Affiliations**
  + name : Nom de l'affiliation
* **Authors**
  + name : Nom de l'auteur
* **Article**
  + title\_review : Titre de la revue
  + date : Date de publication
  + title : Titre de l'article
  + abstract : Résumé de l'article
  + pmid : Identifiant PubMed
  + doi : DOI de l'article
  + disclosure : Conflits d'intérêts
  + mesh\_terms : Termes MeSH associés
  + url : URL de l'article
  + term : Terme associé

#### **Relations (Cardinalités)**

* **Relation entre Article et Authors (via Authorship) :**
* Un article peut être écrit par plusieurs auteurs.
* Un auteur peut contribuer à plusieurs articles.
* Relation : N:M (Many-to-Many), gérée par le modèle intermédiaire Authorship.
* **Relation entre Authors et Affiliations (via Authorship) :**
* Un auteur peut être associé à plusieurs affiliations.
* Une affiliation peut être liée à plusieurs auteurs.
* Relation: N:M (Many-to-Many), gérée implicitement via Authorship.
* **Relation entre Article et Affiliations (via Authorship) :**
* Un article peut être lié à plusieurs affiliations (via ses auteurs).
* Une affiliation peut apparaître dans plusieurs articles.
* Relation: N:M (Many-to-Many), gérée indirectement via Authorship.

##### Modèle Logique de Données (MLD)

**Affiliations**

* **id\_affiliation** (clé primaire)
* name (nom de l'affiliation)

**Authors**

* **id\_author** (clé primaire)
* name (nom de l'auteur)

**Article**

* **id\_article** (clé primaire)
* title\_review (titre de la revue)
* date (date de publication)
* title (titre de l'article)
* abstract (résumé de l'article)
* pmid (identifiant PubMed)
* doi (identifiant DOI)
* disclosure (déclaration de conflit d'intérêt)
* mesh\_terms (termes MeSH)
* url (URL)
* term (terme associé à l'article)

**Authorship (Table de jonction entre Authors, Article, et Affiliations)**

* **id\_authorship** (clé primaire)
* id\_article (clé étrangère vers Article)
* id\_author (clé étrangère vers Authors)
* id\_affiliation (clé étrangère vers Affiliations)

##### Modèle Physique des Données (MPD)

#### 

|  |  |  |
| --- | --- | --- |
| Attribut | Type | Contrainte |
| id | INTEGER | PRIMARY KEY |
| name | TEXT | NOT NULL |

Table. Affiliations

|  |  |  |
| --- | --- | --- |
| Attribut | Type | Contrainte |
| id | INTEGER | PRIMARY KEY |
| name | VARCHAR | NOT NULL |

Table. Authors

|  |  |  |
| --- | --- | --- |
| Attribut | Type | Contrainte |
| id | INTEGER | PRIMARY KEY |
| title\_review | VARCHAR | NULL |
| date | DATE | NULL |
| title | VARCHAR | NULL |
| abstract | TEXT | NULL |
| pmid | INTEGER | NULL |
| doi | VARCHAR | NULL |
| disclosure | TEXT | NULL |
| mesh\_terms | TEXT | NULL |
| url | VARCHAR(200) | NULL |
| term | VARCHAR | NULL |

Table. Article

|  |  |  |
| --- | --- | --- |
| Attribut | Type | Contrainte |
| id | INTEGER | PRIMARY KEY |
| author\_id | INTEGER | FOREIGN KEY -> Authors |
| article\_id | INTEGER | FOREIGN KEY -> Article |
| affiliation\_id | INTEGER | FOREIGN KEY -> Affiliations |
| UNIQUE (author\_id, article\_id, affiliation\_id) |  |  |

Table. Authorship

Compétence validée : Rédaction des spécifications techniques pour le stockage des données.

Compétence validée : Modélisation de la structure des données de la base de données selon la méthode Merise\*.

#### Extraction des données depuis une base externe

Le principe est assez similaire, on précise simplement dans le modèle le nom de la table et que l’on ne la gère pas. Dans la vue, on indique le type de la base de données (ici external) que l’on utilise. Le reste est similaire. Ici, on veut entrer le nom d’un gène et extraire les informations correspondantes à ce gène stockées dans la base de données externe.

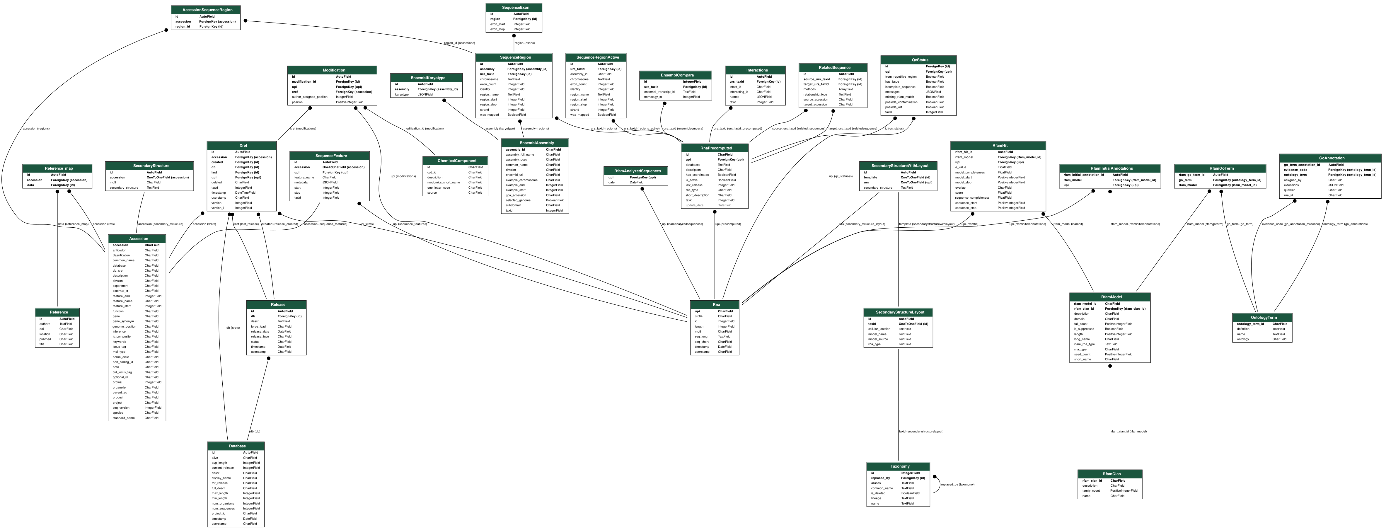


Figure . Base de données externe

Compétence validée : Connexion à une base de données externe.

## C2. Requêtes SQL

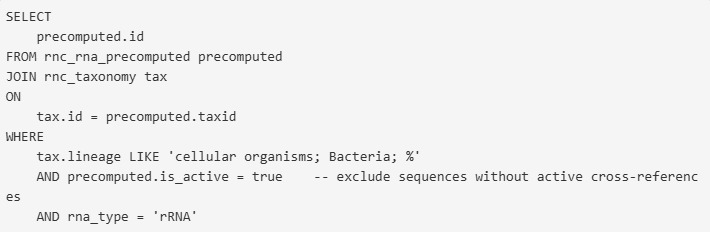


Figure 5. Requête SQL pour extraire des identifiants de la table rna\_precomputed

Cette requête retourne une liste des identifiants (id) de la table rna\_precomputed.

Ces identifiants correspondent à des ARN ribosomiques actifs appartenant aux bactéries.

Voici un exemple d’une requête SQL pour récupérer toutes les données d’un article avec ses auteurs et leurs affiliations:

|  |
| --- |
| SELECT a.\*, au.\*, af.\* FROM article art JOIN authorship a ON a.article\_id = art.id JOIN author au ON a.author\_id = au.id JOIN affiliation af ON a.affiliation\_id = af.id WHERE art.id = {pk}; |

Figure 6. Requête SQL

On sélectionne les entrées des trois tables puis on indique la première source ici article que l’on va lier à la table intermédiaire via l’id de l’article et article\_id puis lier cette table aux deux tables restantes auteur par l’id de l’auteur et auteur\_id et affiliation par l’id de l’affiliation et affiliation\_id, enfin on précise de quel article il s’agit avec son id et pk (primary key) qui a été récupéré dans l’url via une requête http sur l’article). J’utilise pk car Django, identifie les primaries key par pk.  
  
Compétence validée : Exécution programmatique des requêtes d’extraction de type SQL\*.

Compétence validée : Ecriture des requêtes d’extraction de type SQL de récupération de données stockées en base de données et depuis un système big data (Hive).

## C3. Nettoyage et agrégation des données

### Insertion des données en base

Les scripts de scraping et d’insertion en base de données s’écrivent dans le fichier business\_logic.py (logique métier) car il s’agit d’un script propre à ce projet, les fonctions courantes sont stockées dans utils.py.

Afin de réaliser le filtrage des données pour gérer les données vides ou au mauvais format, j’indique à la fonction qui extrait les données qui si l’information n’existe pas, il retourne une chaîne vide ou None et pour le format (ici la date), j’utilise une fonction pour la remettre au bon format.

Compétence validée : Programmation des filtrages/parsing des données utiles dans les résultats obtenus depuis : une API, depuis la lecture d’un ou de plusieurs fichiers de données, et depuis l’HTML collecté d’un site web (scraping).

Compétence validée : Programmation de l’identification des entrées corrompues dans le jeu de données (par exemple, données partielles et/ou manquantes).

Compétence validée : Programmation de la suppression des entrées corrompues.

Compétence validée : Programmation de l’identification des entrées au format non normalisé.

Compétence validée : Programmation de l’homogénéisation des formats de données (par exemple, format des dates, des unités, etc.).

Afin de réaliser l’extraction et l’insertion en base de données, j’ai créé deux commandes dans le fichier commands.py : commands scrap\_article et commands article\_to\_database.

Une fois scrappés, les articles sont stockés dans un JSON ou un CSV, puis de là insérés dans une base de données.

Le stockage dans un JSON permet de séparer les étapes de scraping et d’insertion en base de données. Ce qui est pratique car s’il y a un problème lors de l’insertion des données en base, il faut reprendre le scraping depuis le début, de plus stocker les données à deux endroits différents permet d’éviter la perte des données si la base de données est effacée, cela permet également de pouvoir modifier la structure de la base de données tout en conservant les données ou encore de faire une insertion dans plusieurs bases de données…

Compétence validée : Lecture d’un fichier de données dans un script.

La base de données choisie est Postgres, car c’est celle utilisée en entreprise et elle est native avec Django.

Compétence validée : Choix du système de gestion de base de données.

Postgres est open-source, elle existe depuis 1986 donc il y a une vaste communauté, beaucoup de modules et de documentation. Elle supporte les bases relationnelles et les objets.

Pour installer la base de données, j’ai téléchargé le logiciel Pg4 admin.

Compétence validée : Documentation de la procédure d’installation du système de gestion de base de données.

J’ai défini quatre classes :

* *Affiliations* pour stocker les noms des affiliations.
* *Authors* pour stocker les noms des auteurs.
* *Article* pour stocker toutes les informations basiques liées aux articles et la relation Many-to-Many permettant de relier un article avec ces auteurs et pour chaque auteur ses affiliations.
* *Authorship est* la table intermédiaire avec trois clés étrangères pour les articles, les auteurs et les affiliations.

La connexion avec la base de données se fait dans settings.py.

Pour créer la base de données on va créer dans models.py des classes dont chacune correspond à une table de la base de données. On définit ici, le type de variable, si elle peut être nulle, le nom des colonnes... Ensuite on valide les migrations avec la commande makemigrations et on crée la base de données avec migrate. La base de données est créée.

Pour ajouter les données en base si les champs sont ne sont pas des clés étrangères, il se remplissent directement en ajoutant les champs dans l’objet model. Pour chaque article, on a une liste d’auteurs et pour chaque auteur une liste d’affiliations. On itère sur chaque article, puis sur chacun de ses auteurs et chacune de leur affiliation, on les ajoute en base et on récupère les objets de chaque ligne correspondante (les objets auteurs et affiliations) que l’on insère à chaque itération dans l’objet Authorship.

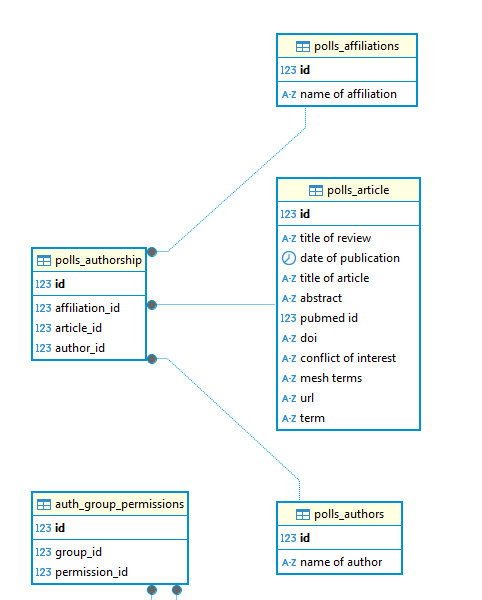


Figure 7. Schéma de la base de donnée interne

L’insertion des données peut aussi se faire via une requête http, comme ci-dessous pour le CRUD.

Compétence validée : Connexion programmatique à un système de gestion de base de données et à un système big data.

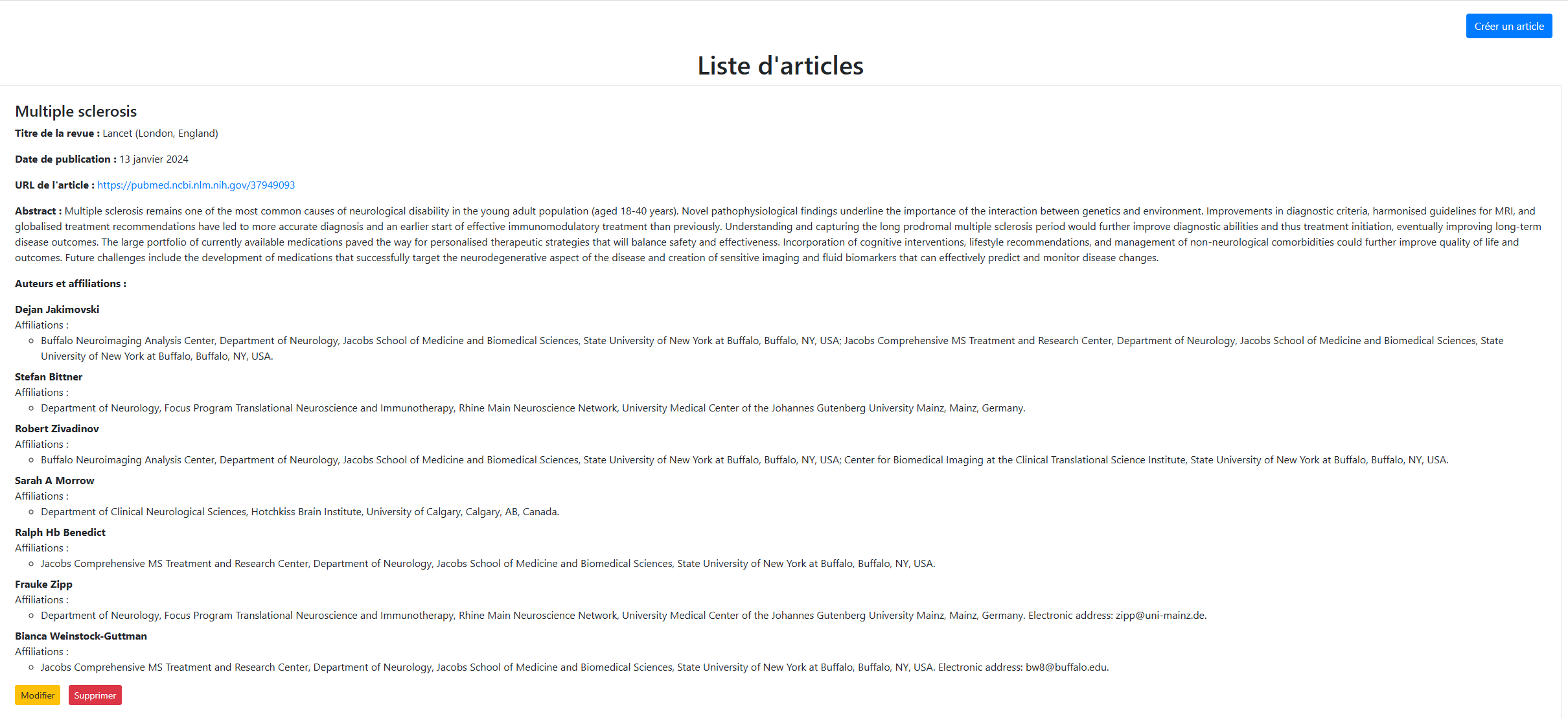
Compétence validée : Création de la base de données dans le système de gestion de base de données.

### CRUD

Maintenant que toutes les données ont été ajoutées à la base, il faut pouvoir mettre à jour la base.

Le CRUD (Create, Read, Update, Delete) se fait dans le fichier views.py car il nécessite un formulaire pour modifier la base de données et un affichage pour visualiser les données.

On peut créer un article, lire l’ensemble des articles, mettre à jour un article ou le supprimer.

Figure 8. CRUD

Chacune de ces actions est réalisable via une requête HTTP sur l’API FastAPI ou sur Django.

Le principe est qu’en allant sur une adresse URL, la fonction correspondante dans la vue récupère le formulaire vide en GET, puis le renvoie sur la page HTML équivalente, là, l’utilisateur va remplir le formulaire et à la validation revenir sur la vue en POST qui va traiter les infomations du formulaire et selon le type d’action choisit par l’utilisateur faire de l’insertion en base de données, de la mise à jour…

Compétence validée : Construction les requêtes HTTP pour la récupération des données depuis un service web (REST\*).

La documentation de l’API est générée automatiquement par Swagger et est accessible à l’adresse : <http://127.0.0.1:8000/swagger/>

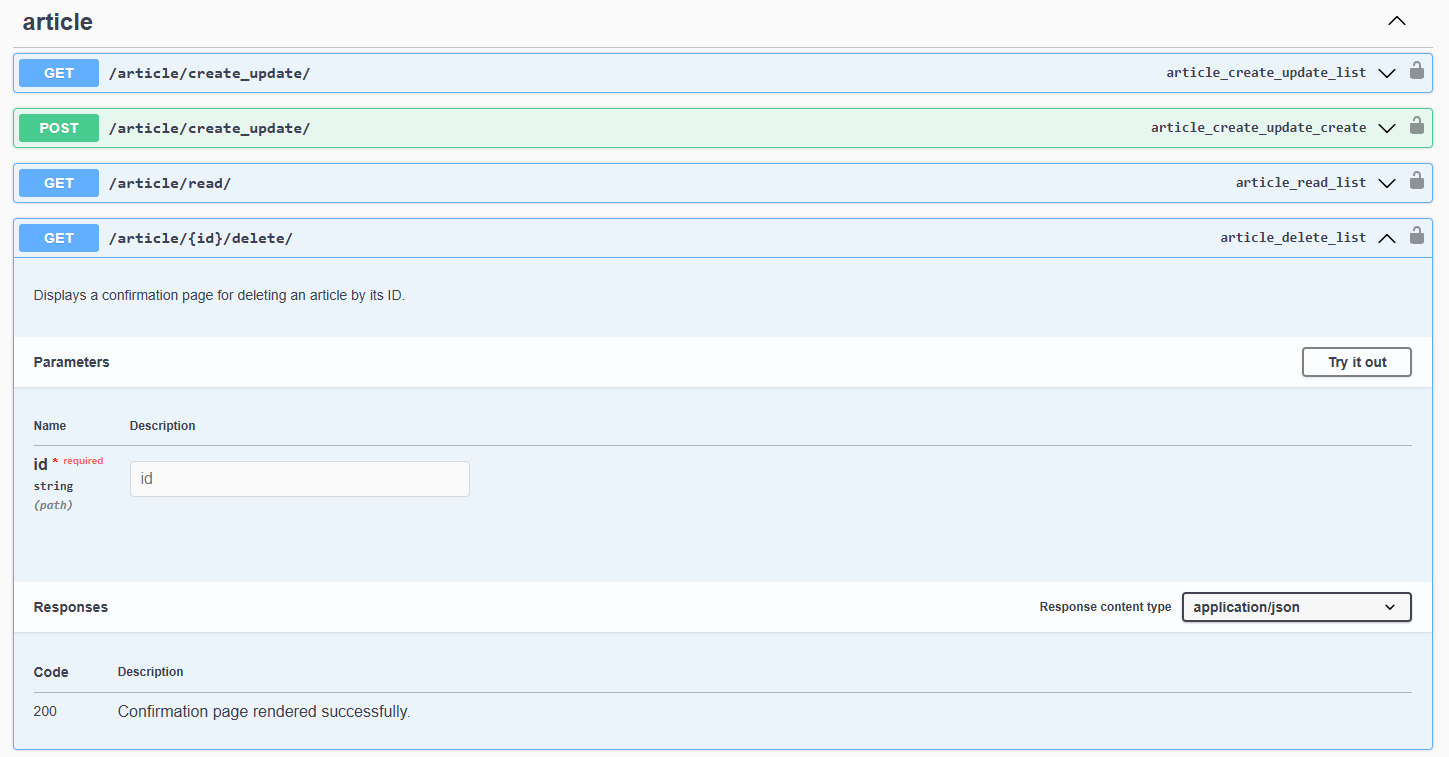


Figure 9. Documentation

# Conclusion

Nous avons récupéré toutes les données depuis PubMed et les avons rangées dans la base de données. Elles sont prêtes à être utilisées pour le RAG. Le CRUD a été mis en place permettant l’ajout et la correction des articles dans la base depuis une interface utilisateur.