مدرسة علوم المعلومات
ⵜⵉⵎⵔⵉⵜ ⵏ ⵜⵓⵙⵙⵓⵔⵉⵏ ⵉⵏⵖⵎⴰⵙⵏ
ECOLE DES SCIENCES DE L'INFORMATION

Cours /élément de module : 3.5.2 Moteurs de recherche d'information (S3)
Module: 3.5– Technologies de recherche d'information
Niveau et option : 2ème année cycle Ingénieur-Filière Ingénierie de l'Information Numérique (IIN)
Charge horaire globale : 24 heures
Année universitaire : 2023-2024
Professeur : Dr. Amine SENNOUNI

Conception et réalisation d'un moteur de recherche

IntelliSearch

Travail de : MATA Ayoub

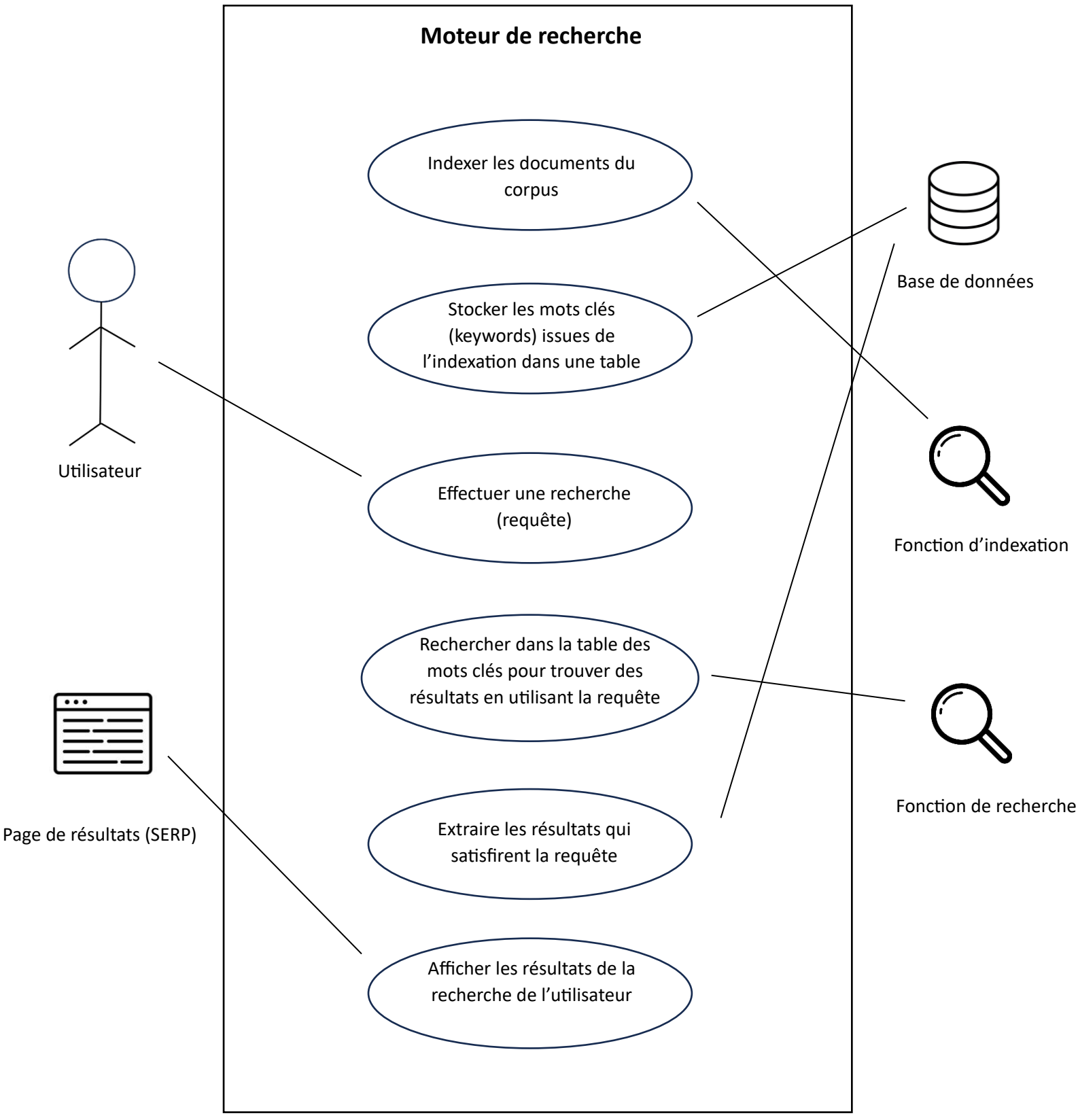
Filière : IIN

Numéro d'ordre : 122

Tableau de matière

Phase de conception :

Diagramme de cas d'utilisation



Phase de réalisation :

Corpus :

Des articles existant dans le web dans le domaine de l'intelligence artificielle, chatbots, Machine Learning etc…

Langages utilisés :

- Python (pour la logique derrière le moteur de recherche)

- HTML et CSS (pour l'interfaçage de la barre de recherche et la page des résultats de la recherche)

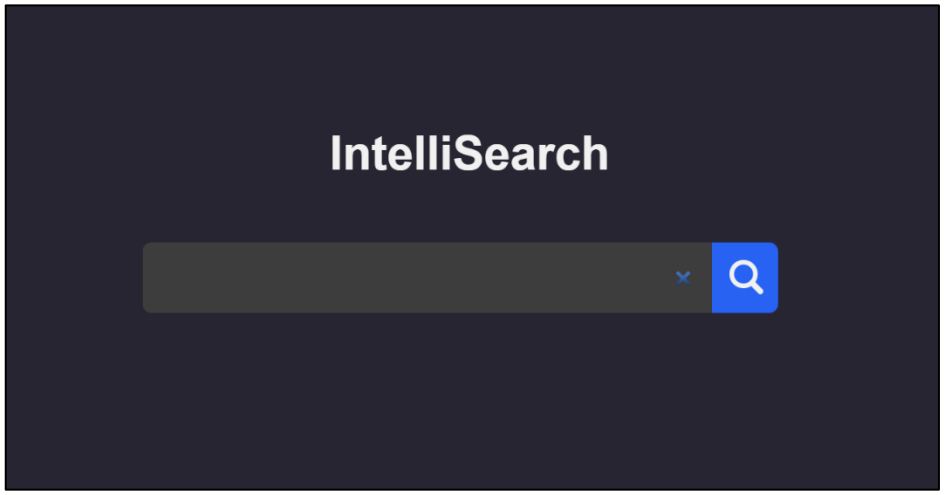
SGBDR utilisé :

MySQL : pour stocker les liens vers les documents de corpus dans une table documents, et pour stocker les mots clés issues de la phase d'indexation dans une table keywords.

Framework web utilisé :

Flask : Un Framework web flexible qui permet de faire la liaison entre l'interface de l'application web et la logique en python de l'application.

Les interfaces des pages web :



Interface de la barre de recherche

Interface de la page des résultats SERP

Fonctionnement du moteur de recherche :

Le fonctionnement de ce moteur de recherche s'appui sur deux fonctions principales :

- **La fonction d'indexation (index_documents(documents))**

```
- def index_documents(documents):
      connection = MySQLdb.connect(host=DB_HOST, user=DB_USER, password=DB_PASSWD,
  database=DB_DATABASE)
      cursor = connection.cursor()
      cursor.execute("CREATE TABLE IF NOT EXISTS keywords (id INT AUTO_INCREMENT PRIMARY
  KEY, name VARCHAR(255), documents VARCHAR(255),title VARCHAR(255),description
  TEXT,pathtopdf VARCHAR(255))")
      for document in documents:
          cursor.execute("INSERT INTO documents (name, path,title,description,pathtopdf)
  VALUES (%s, %s, %s, %s, %s)", (document["name"], document["path"], document["title"],
  document["description"], document["pathtopdf"]))
      # Création de la table `keywords` si elle n'exist pas encore
      cursor.execute("CREATE TABLE IF NOT EXISTS documents (id INT AUTO_INCREMENT PRIMARY
  KEY, name VARCHAR(255), path VARCHAR(255),title VARCHAR(255),description TEXT,pathtopdf
  VARCHAR(255))
      # Indexation des documents
      for document in documents:
          # Lecture du contenu du document
          with open(document["path"], "r") as f:
              content = f.read()
          # lemmatisation de contenu
          lemmatized_content = WordNetLemmatizer().lemmatize(content)
          # Suppression des mots vides
          stop_words = stopwords.words("french")

  mots_vides=["tous","tout","toute","toutefois","toutes","treize","trente","tres","trois","
  troisième","troisièmement"]
          filtered_content = " ".join([word for word in lemmatized_content.split() if word
  not in stop_words])
          # Extraction des mots clé
          keywords = re.findall(r"\b[a-z0-9_-]+\b", filtered_content)
          # Insertion des mots clés dans la table `keywords`
          for keyword in keywords:
              if cursor.execute("SELECT COUNT(*) FROM keywords WHERE name = %s AND
  documents = %s", (keyword, document["name"])):
                  count = cursor.fetchone()[0]
              if count > 0:
                  continue
              if(keyword in mots_vides):
                  continue
              if(len(keyword)<=2):
                  continue
              if(keyword not in mots_vides):
                  cursor.execute("INSERT INTO keywords (name,
  documents,title,description,pathtopdf) VALUES (%s, %s, %s, %s, %s)",(keyword,
  document["name"], document["title"], document["description"], document["pathtopdf"]))
      connection.commit()
      cursor.close()
      connection.close()
      return
```

Les points principaux dans la logique de la fonction d'indexation sont :

-Lecture du contenu du document.

-Lemmatisation du mot avec WordNetLemmatizer.

-Suppression des mots vides (stop words).

-Extraction des mots-clés à l'aide d'une expression régulière.

-Insertion des mots-clés dans la table keywords en associant chaque mot-clé au document correspondant.

- **La fonction de recherche (search(query))**

```
- def search(query):
      connection = MySQLdb.connect(host=DB_HOST, user=DB_USER, password=DB_PASSWD,
  database=DB_DATABASE)
      cursor = connection.cursor()
      # Séparer le requete en des mots individuelle
      query_words = query.lower().split()
      # Cherchef chaque mot dans la table 'keywords'
      results = []
      for word in query_words:
          # obtenir les documents associés avec le mot
          cursor.execute("SELECT * FROM keywords WHERE name = %s", (word,))
          documents = cursor.fetchall()
          # Si le mot est trouvé, ajouter les informations liées au mot dans la
  liste 'results'
          if documents:
              for document in documents:
                  if document[2] not in results:
                      results.append(document[2])
                  if document[3] not in results:
                      results.append(document[3])
                  if document[4] not in results:
                      results.append(document[4])
                  if document[5] not in results:
                      results.append(document[5])
      return list(results)
```

Les points principaux dans la logique de la fonction d’e recherche sont :

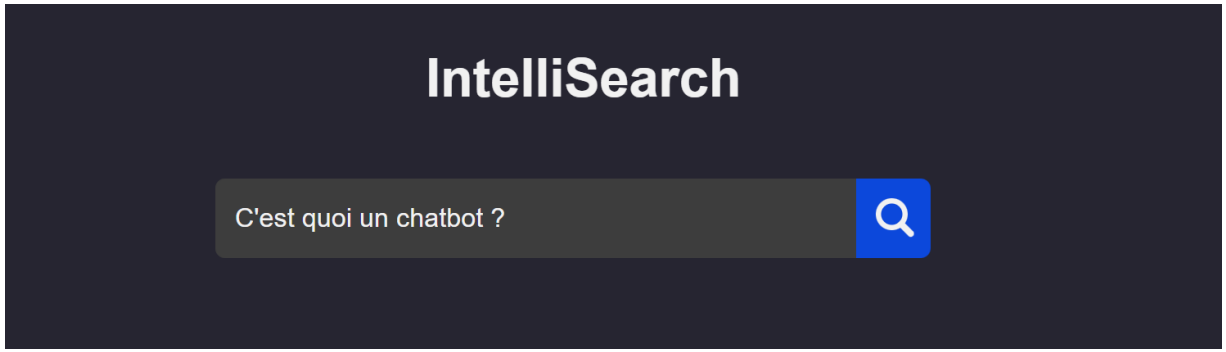
-Séparer la requête en des mots individuelles pour pouvoir faire la comparaison ‘’binaire’’ entre chaque mot de la requête et chaque mot dans la table des mots clés.

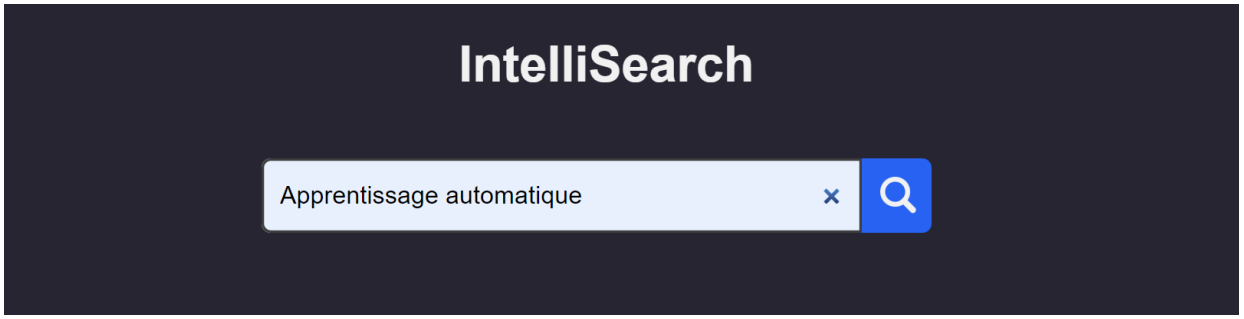
-Obtenir les informations sur les documents qui sont associés avec le mot

-Stocker chaque information (titre, description..) dans la liste ‘results’ pour pouvoir les récupérer et afficher dans la page de résultats

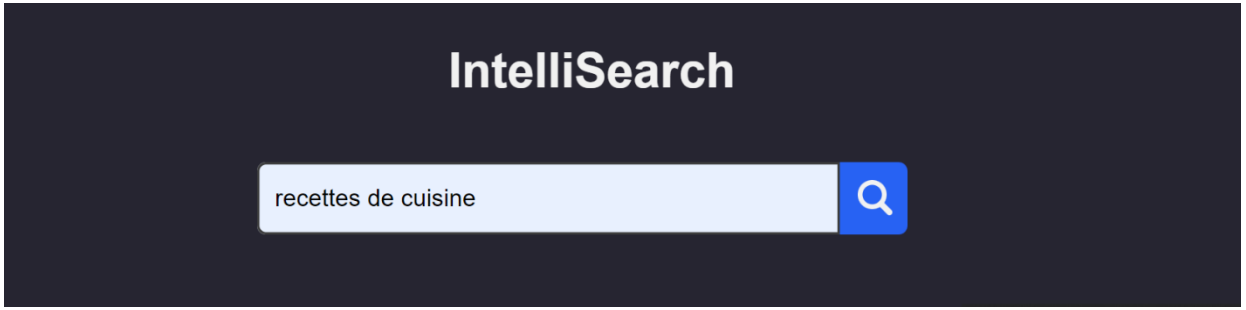
Test des requêtes fructueuses et non fructueuses

Requêtes fructueuses

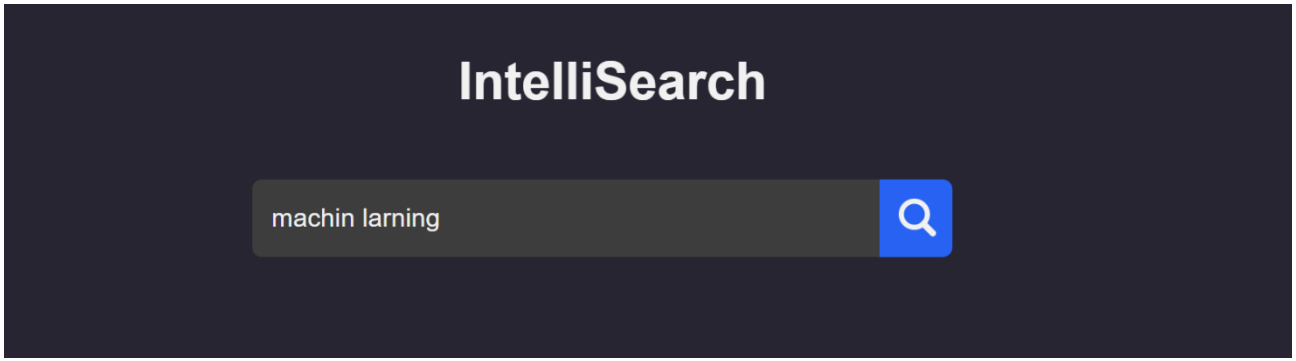




Requêtes non fructueuses



Résultats de recherche pour la requête "recettes de cuisine"



Résultats de recherche pour la requête "machin larning"

Améliorations et remédiations :

-Concernant le traitement de la requête :

-Ajouter d'autre opérations d'indexation comme la radicalisation et tokenisation…

-Concernant la recherche :

--Utiliser un fichier Json par exemple comportant le dictionnaire de la langue française, mettre en place une fonction de similarité qui va comparer le mot dans la requête et les mots dans la table des mots clés, cela va remédier au problème d'absence de résultats dans le cas des fautes d'orthographes au niveau de la requête.