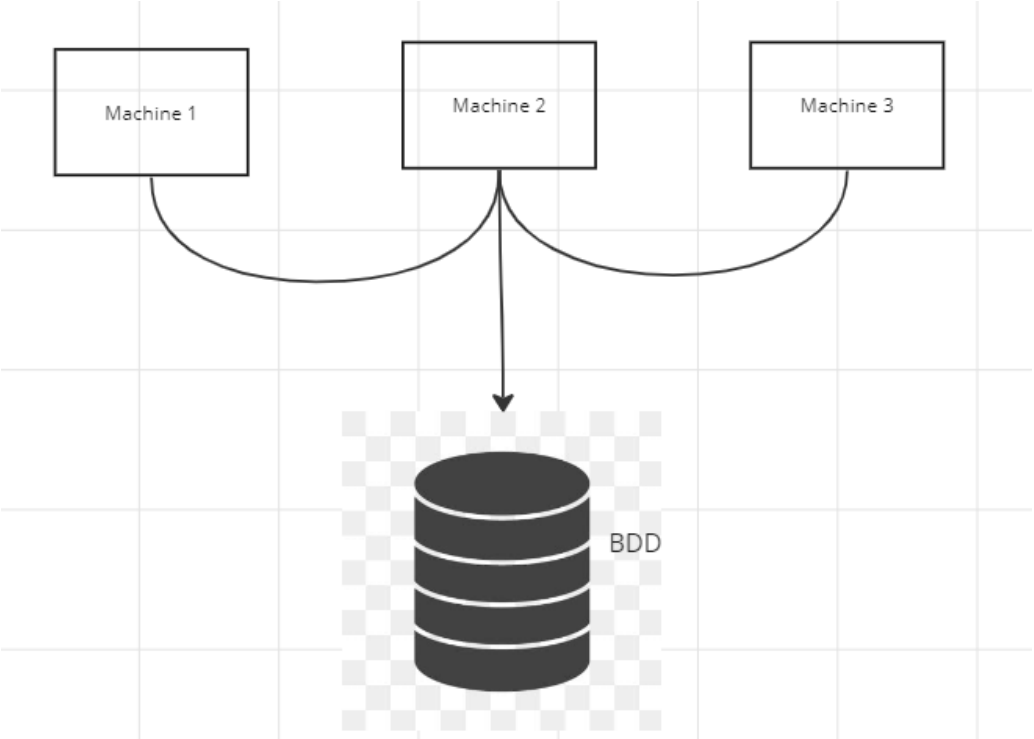
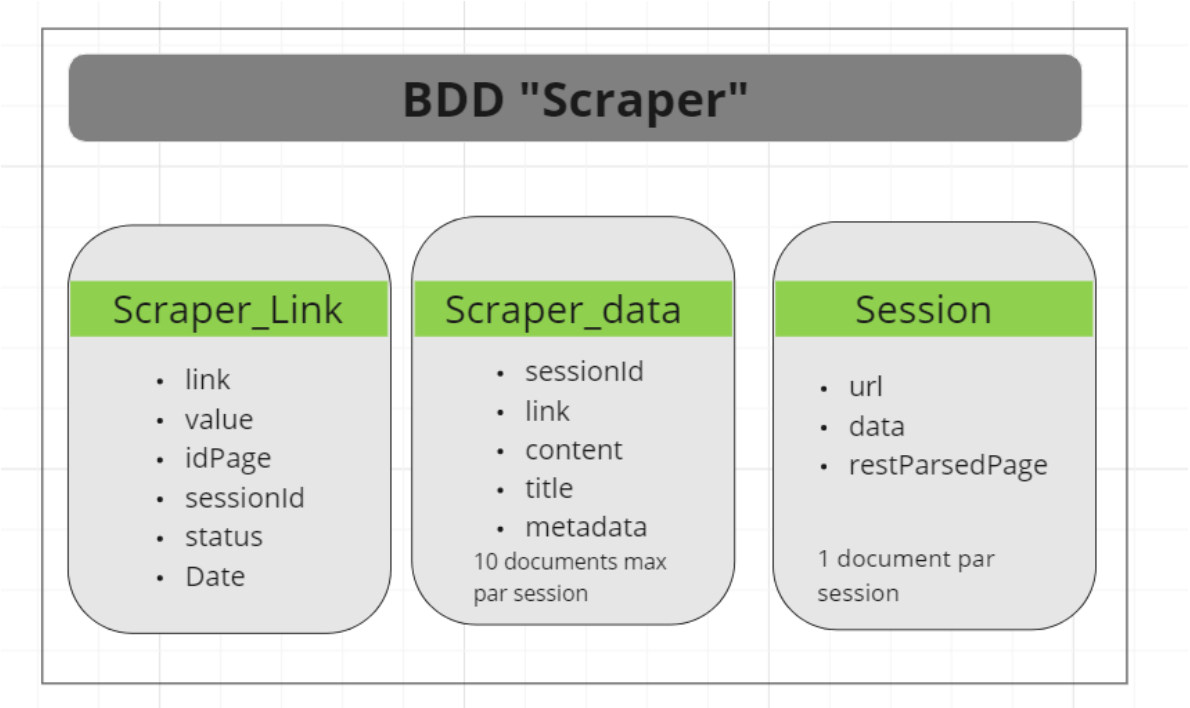
Architecture du Système de scraping

1. Architecture de connexion à la base de donnée



2. Architecture de la bdd “Scraper”



La BDD “ Scraper ” est constitué de 3 collections :

1. Scraper_data
    a. Champs
        _id : Identifiant unique du document.
        sessionId : Identifiant de session, correspondant à la collection “Session”.
        link : Lien de la page.
        content : Contenu HTML de la page.
        title : Titre de la page.
        metadata : Liste des métadonnées de la page, contenant les en-têtes et les emphases.

    b. Description
        Cette collection contient les données extraites de chaque page scraper. Chaque document représente une page et contient des informations telles que le lien, le contenu, le titre et les métadonnées de la page, le nombre de documents est limité à 10 documents max par session.

2. Scraper_Link
    a. Champs
        _id : Identifiant unique du document.
        link : lien dans une page.
        value: description du lien.
        idPage :Identifiant de la page à laquelle le lien est associé, correspondant à la collection "Scraper_data".
        sessionId: Identifiant de session, correspondant à la collection “Session”.
        status: status des liens qui peut être “en attente, en cours, terminé”.
        Date: Date et heure d’ajout des liens .

    b. Description
        Cette collection contient les liens extraits à partir des pages scrapées. Chaque document représente un lien et contient des informations telles que l'URL du lien, sa valeur, l'identifiant de la page à laquelle il est associé et l'état "parsed" (analysé ou non).

3. Session
    a. Champs
        _id : Identifiant unique du document.

url : Url de la session.
date : Date et heure de création de la session.
restParsedPage : Nombre restant de pages à analyser dans la session.

b. Description
Cette collection enregistre les sessions de scraping. Chaque document représente une session et contient des informations telles que l'URL de la session, la date de création et le nombre restant de pages à analyser dans la session.

3. Architecture du code

- La classe " **MongodbManager**" : Gère la connexion à la base de données MongoDB et fournit des méthodes pour insérer, récupérer et mettre à jour les données. Elle utilise la bibliothèque PyMongo pour interagir avec la base de données.

- La classe "**WebScraper**" : Représente un scraper web. Elle prend une URL et une limite de liens à extraire. Elle utilise la bibliothèque Requests pour effectuer les requêtes HTTP et la bibliothèque BeautifulSoup pour analyser le contenu HTML et extraire les informations nécessaires. Elle contient également des méthodes pour extraire le titre, les en-têtes, les liens et les emphases d'une page HTML.

- Gestions d'erreurs : La fonction "**retry"** est un décorateur qui enveloppe une fonction avec une logique de réessai en cas d'erreur de requête. Cela permet de gérer les éventuelles erreurs de connexion réseau lors du scraping.

- Fonction principale : La fonction "**run**" est la fonction principale qui effectue le scraping des pages. Elle utilise la classe "**MongodbManager**" pour récupérer la session à partir de la base de données.Elle récupère la page initiale, puis itère sur les liens à extraire pour chaque page. Elle utilise les méthodes de la classe "**WebScraper**" pour extraire les informations et les insérer dans la base de données à l'aide de la classe "**MongodbManager**".

- Fonction "**statusManager**" : cette fonction est utilisée pour gérer les statuts en cours de session lorsqu'une machine tombe en panne et reprend la prise en charge, et vérifie si une page existe déjà dans la base de données et met à jour le statut en conséquence

- **API Flask** : Une route /api/scrape est définie pour lancer le scraping en utilisant une requête POST. La fonction scraper est associée à cette route et elle effectue le scraping en appelant la fonction run avec les paramètres appropriés.

- Gestion de conflits d'exécutions entre les machines : Le module “**argparse**” est utilisé pour récupérer un argument en ligne de commande (args.first) qui détermine si la machine actuelle est celle qui exécute en premier le scraping initial. Cela permet de différencier le comportement de la machine initiale des autres machines.

L'architecture globale implique l'utilisation d'une base de données MongoDB centralisée pour stocker les données extraites, tandis que chaque machine exécute le scraping de manière parallèle. Chaque machine récupère un lien non traité à partir de la base de données, effectue le scraping de la page correspondante, insère les données dans la base de données et met à jour l'état du lien pour indiquer qu'il a été traité. Cela permet d'éviter les doublons et de synchroniser les différentes machines lors de l'extraction des données. L’API Flask permet à une autre application d’utiliser le script en son sein, il n’est donc plus possible de scraper en passant par la ligne de commande. Un outil comme Postman permet de faire des requêtes avec par exemple :
http://127.0.0.1:5000/api/scrape?url=https://fr.wikipedia.org/wiki/France&first=True ;
qui permet de lancer une session de scraping sur la page wikipedia France.