Fonctionnement de l'Architecture de "Brief My Press-AI"

L'architecture présentée est au cœur du projet Brief **My Press-AI**, dont l'objectif est de fournir une solution innovante et centralisée pour la collecte, l'analyse et la mise en forme d'articles de presse issus de diverses sources. Voici une description détaillée du fonctionnement de cette architecture.

Ce système s'appuie sur un processus automatisé, intégrant des étapes de récupération, transformation, et présentation des données.

1. Client Web (Navigateur)

L'utilisateur final (professionnel de la presse ou analyste) interagit avec le système via un navigateur.

- **Objectif principal** : Accéder facilement aux articles collectés, transformés et filtrés.
- **Action de l'utilisateur** : Il envoie une requête HTTP (comme demander les articles les plus récents ou les plus pertinents sur un sujet donné).

2. Gestion des Routes (url.py)

Le composant **url.py** gère les URL qui mènent aux différentes fonctionnalités du site.

- Exemple : Une URL comme `/articles/finance` permettrait à l'utilisateur de consulter les dernières actualités sur la finance.
- Cette gestion assure que chaque requête est redirigée vers le module Django approprié, garantissant une réponse rapide et ciblée.

3. Les Vues (views.py)

Le fichier **views.py** est le moteur principal de la logique métier.

- **Rôles des vues** :
    - Faire le lien entre la requête de l'utilisateur et les données stockées dans MongoDB.
    - Réaliser les traitements nécessaires sur les données, comme le tri, le filtrage ou l'enrichissement.
    - Retourner une réponse sous forme d'une page HTML dynamique grâce aux templates.
- **Enrichissement des données** : Les vues utilisent des données traitées par le pipeline ETL pour répondre précisément aux besoins de l'utilisateur.

4. Les Modèles (models.py)

Les modèles Django (ORM) permettent une interaction fluide avec la base de données MongoDB :

- **CRUD (Créer, Lire, Mettre à jour, Supprimer)** : Ils assurent que les données issues des APIs externes peuvent être stockées et gérées efficacement.
- Exemple : Un modèle peut représenter un **article de presse**, comprenant des champs comme le titre, la source, la date, et le contenu.
- **MongoDB comme base NoSQL** : Adaptée pour gérer des données semi-structurées (JSON), elle facilite l'intégration et la manipulation d'articles variés.

5. Base de Données (MongoDB)

- **Stockage** : Toutes les données collectées via les APIs (NewsAPI, Alpha Vantage, FMP) sont enregistrées dans MongoDB.
- **Avantage NoSQL** : La flexibilité permet de gérer des articles provenant de plusieurs sources sans besoin de schéma rigide.
- Ces données sont ensuite interrogées et structurées pour être renvoyées à l'utilisateur.

6. Processus ETL (Extract, Transform, Load)

Le processus **ETL**, géré par le script **etl.py**, est un élément clé de cette architecture.

- **Rôle principal** : Transformer des données brutes issues d'APIs en informations prêtes à être exploitées par les utilisateurs.
- **Étapes détaillées** :
    1. **Extraction** : Les articles sont récupérés automatiquement via des appels aux APIs (ex. NewsAPI pour la presse, Alpha Vantage pour les données financières).
    2. **Transformation** : Les données sont nettoyées, enrichies, et normalisées (par ex., suppression des doublons, ajout de mots-clés pertinents).
    3. **Chargement** : Les articles transformés sont insérés dans MongoDB pour une utilisation rapide et flexible.

7. Templates (HTML, CSS, JavaScript)

Le système utilise des templates pour afficher les données de manière claire et professionnelle.

- **Composants** : Les templates génèrent des interfaces utilisateurs dynamiques à partir des données traitées.
- **Technologies utilisées** : HTML, CSS et JavaScript pour une interface utilisateur moderne et interactive.
- **Personnalisation** : Les utilisateurs peuvent accéder à des vues spécifiques (exemple : articles par thématique ou par pertinence).

8. Flux Global du Système

1. **Requête utilisateur** : L'utilisateur accède à une URL via son navigateur.
2. **Traitement Django** :
    - Les requêtes sont dirigées vers les **vues** via `url.py`.
    - Les vues interrogent les **modèles** pour récupérer les données nécessaires dans MongoDB.
    - Les modèles exploitent les résultats du pipeline **ETL**.
3. **Réponse HTML** : Les templates génèrent une page web dynamique, qui est renvoyée au navigateur de l'utilisateur sous forme de **réponse HTTP**.

Conclusion

Cette architecture modulaire est parfaitement alignée avec les objectifs de **My Press AI** : fournir un outil de veille performant et facilement extensible. L'intégration de Django, MongoDB, et le pipeline ETL assure une gestion robuste des données tout en offrant une interface utilisateur optimisée pour des utilisateurs exigeants.