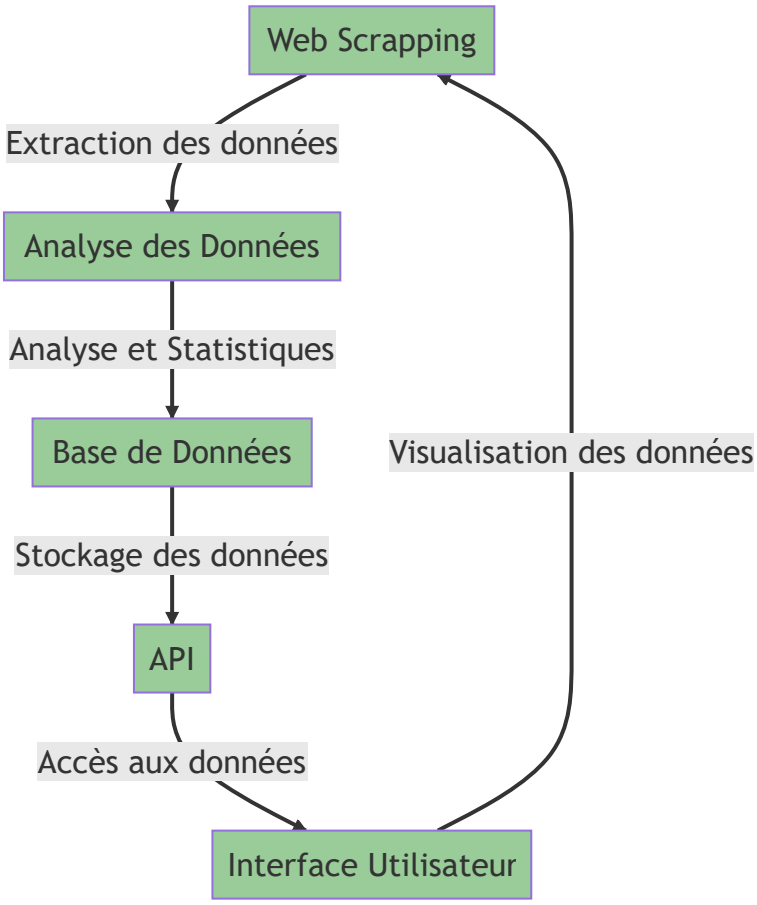
Smsar

Description du Projet

Ce projet vise à développer une application web qui fournit des statistiques et des informations sur le marché immobilier. L'application recueille des informations sur différentes annonces immobilières et les présente sous forme de statistiques. En outre, l'application offre des informations sur les quartiers des villes, y compris les établissements locaux et les articles de presse pertinents.

Sections du Projet



Web Scrapping

Le web scrapping est une technique essentielle dans ce projet, utilisée pour extraire des informations à partir de divers sites web d'annonces immobilières. Cette méthode nous permet de recueillir une grande quantité de données pertinentes pour notre analyse, y compris le type de propriété (par exemple, maison, appartement, terrain, etc.), le prix demandé, l'emplacement de la propriété, et d'autres détails qui peuvent être importants pour les acheteurs, les vendeurs, et les investisseurs immobiliers.

Dans le cadre de notre preuve de concept, nous avons mis en œuvre le web scrapping pour la catégorie "immobilier à vendre" sur Avito, un site web populaire d'annonces immobilières. Au total, nous avons réussi à scraper 128 000 annonces, ce qui nous a fourni une base de données initiale pour commencer notre analyse.

Cependant, il est important de noter que notre module de web scrapping est conçu pour être flexible et extensible. Bien que notre preuve de concept se concentre sur Avito, le module peut être adapté pour scraper des données à partir d'une variété de sites immobiliers, tels que Sarouty, Mubawab, et Maroc Annonces. Cela signifie que nous pouvons élargir notre base de données pour inclure des informations provenant de plusieurs sources, ce qui nous permettra d'avoir une vue plus complète et plus précise du marché immobilier.

En outre, notre module de web scrapping est également capable de scraper les API d'information. Par exemple, nous pourrions utiliser l'API de Google Places pour obtenir des informations sur les établissements locaux dans un quartier spécifique, ou l'API de Twitter pour recueillir des tweets relatifs à l'immobilier. Ces API peuvent nous fournir des informations supplémentaires qui peuvent être pertinentes pour notre analyse, comme les tendances du marché, les nouvelles de l'industrie, et les commentaires des utilisateurs sur les propriétés spécifiques.

Ce module fournit un outil de web scraping flexible et configurable construit avec Python. Il utilise Selenium pour l'automatisation du navigateur et BeautifulSoup pour l'analyse du HTML. Le module est conçu autour du concept de machine à états, permettant de définir une séquence d'actions à effectuer pendant le processus de scraping.

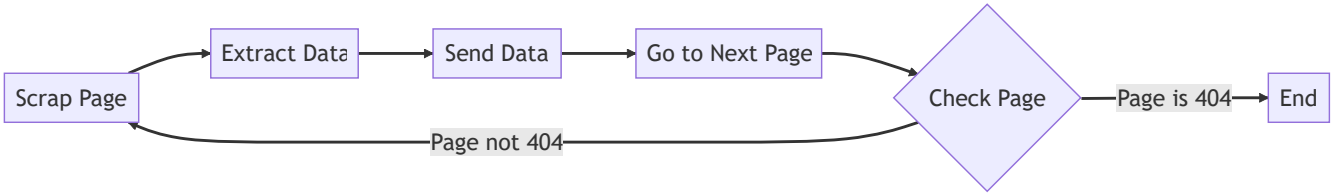
Caractéristiques

- **Configuration flexible** : Le module utilise un fichier de configuration JSON pour définir la séquence d'actions (états) à effectuer pendant le processus de scraping. Chaque état est mappé à une méthode de la classe `Scraper` et les paramètres de cette méthode.
- **Automatisation du navigateur** : Le module utilise Selenium pour automatiser un navigateur web. Cela permet au module d'interagir avec des sites web basés sur JavaScript et d'effectuer des actions telles que cliquer sur des boutons ou naviguer vers différentes pages.
- **Analyse HTML** : Le module utilise BeautifulSoup pour analyser le contenu HTML des pages web et extraire des données.
- **Concept de machine à états** : Le module est conçu autour du concept de machine à états. Cela vous permet de définir une séquence d'actions à effectuer pendant le processus de scraping. La séquence d'actions peut être facilement modifiée en changeant le fichier de configuration.

Utilisation

Pour utiliser le module, vous devez créer un fichier de configuration JSON qui définit la séquence d'actions à effectuer pendant le processus de scraping. Chaque action est représentée par un état dans la machine à états. Le fichier de configuration doit définir l'état initial et une correspondance des états aux méthodes de la classe `Scraper` et leurs paramètres.

Voici un exemple de fichier de configuration :



```
{
    "userAgent": "Votre User Agent",
    "base_url": "https://www.example.com/page1",
    "data_endpoint": "https://api.example.com/data",
    "initial_state": "scrap_page",
    "states": {
        "scrap_page": {
            "method": "scrap_page",
            "parameters": ["By.CSS_SELECTOR", ".listing"],
            "next_state": "extract_data"
        },
        "extract_data": {
            "method": "extract_data",
            "parameters": ["raw_data", "div", "sc-jejop8-0"],
            "next_state": "goto_next_page"
        },
         "call_api": {
            "method": "call_api",
            "parameters": ["api_url"],
            "next_state": "send_data"
        },
        "send_data": {
            "method": "send_data",
            "parameters": ["data"],
            "next_state": null
        },

        "goto_next_page": {
            "method": "goto_next_page",
            "parameters": [],
            "next_state": "check_page"
        },
        "check_page": {
            "method": "check_page",
            "parameters": [],
            "condition": "page_not_404",
            "state_true": "scrap_page",
            "state_false": "end"
        },
        "goto_link": {
            "method": "goto_link",
            "parameters": ["link"],
            "next_state": "scrap_page"
        },

        "end": {
            "method": "end",
            "parameters": [],
            "next_state": null
        }
    }
}
```

Attribut/Méthode	Type	Description
`config`	Attribut	Un dictionnaire contenant la configuration du scraper.
`driver`	Attribut	Un objet WebDriver de Selenium utilisé pour automatiser le navigateur.
`__init__(self, config)`	Méthode	Le constructeur de la classe. Il initialise les attributs `config` et `driver`.
`init_driver(self)`	Méthode	Initialise le WebDriver de Selenium avec les options appropriées.
`scrap_page(self)`	Méthode	Scraper le contenu d'une page web en utilisant le sélecteur CSS fourni dans la configuration.
`extract_data(self, raw_data)`	Méthode	Extraire les données de la page web scrapée en utilisant BeautifulSoup.
`goto_next_page(self)`	Méthode	Naviguer vers la page suivante en utilisant l'URL fournie dans la configuration.
`goto_link(self, link)`	Méthode	Naviguer vers un lien spécifique.
`call_api(self, api_url)`	Méthode	Appeler une API spécifique et retourner la réponse.
`send_data(self, data)`	Méthode	Envoyer les données extraites à un endpoint spécifique en utilisant une requête POST.
`scrape_site(self, config)`	Méthode	Commencer le processus de scraping en utilisant la configuration fournie.

Voici une liste des dépendances principales et une brève explication de leur utilisation :

- **Requests** : Cette bibliothèque est utilisée pour envoyer des requêtes HTTP. Elle est principalement utilisée pour interagir avec les API, par exemple pour envoyer les données extraites à un endpoint spécifique.
- **JSON** : Cette bibliothèque est utilisée pour manipuler les données JSON. Elle est utilisée pour convertir les données extraites en format JSON avant de les envoyer à un endpoint.
- **Time** : Cette bibliothèque est utilisée pour introduire des délais dans le script, par exemple pour attendre que la page web soit chargée ou pour simuler le comportement d'un utilisateur humain.
- **Random** : Cette bibliothèque est utilisée pour générer des nombres aléatoires, par exemple pour introduire des délais aléatoires entre les requêtes afin de simuler le comportement d'un utilisateur humain et d'éviter d'être bloqué par les sites web.
- **Datetime** : Cette bibliothèque est utilisée pour manipuler les dates et les heures, par exemple pour enregistrer le moment où les données ont été extraites.

- **BeautifulSoup** : Cette bibliothèque est utilisée pour analyser le contenu HTML des pages web et extraire des données. Elle permet de naviguer dans la structure de la page et de trouver les éléments en fonction de leurs balises, classes, id, etc.
- **Selenium** : Cette bibliothèque est utilisée pour automatiser un navigateur web. Elle permet d'interagir avec des sites web basés sur JavaScript et d'effectuer des actions telles que cliquer sur des boutons ou naviguer vers différentes pages.

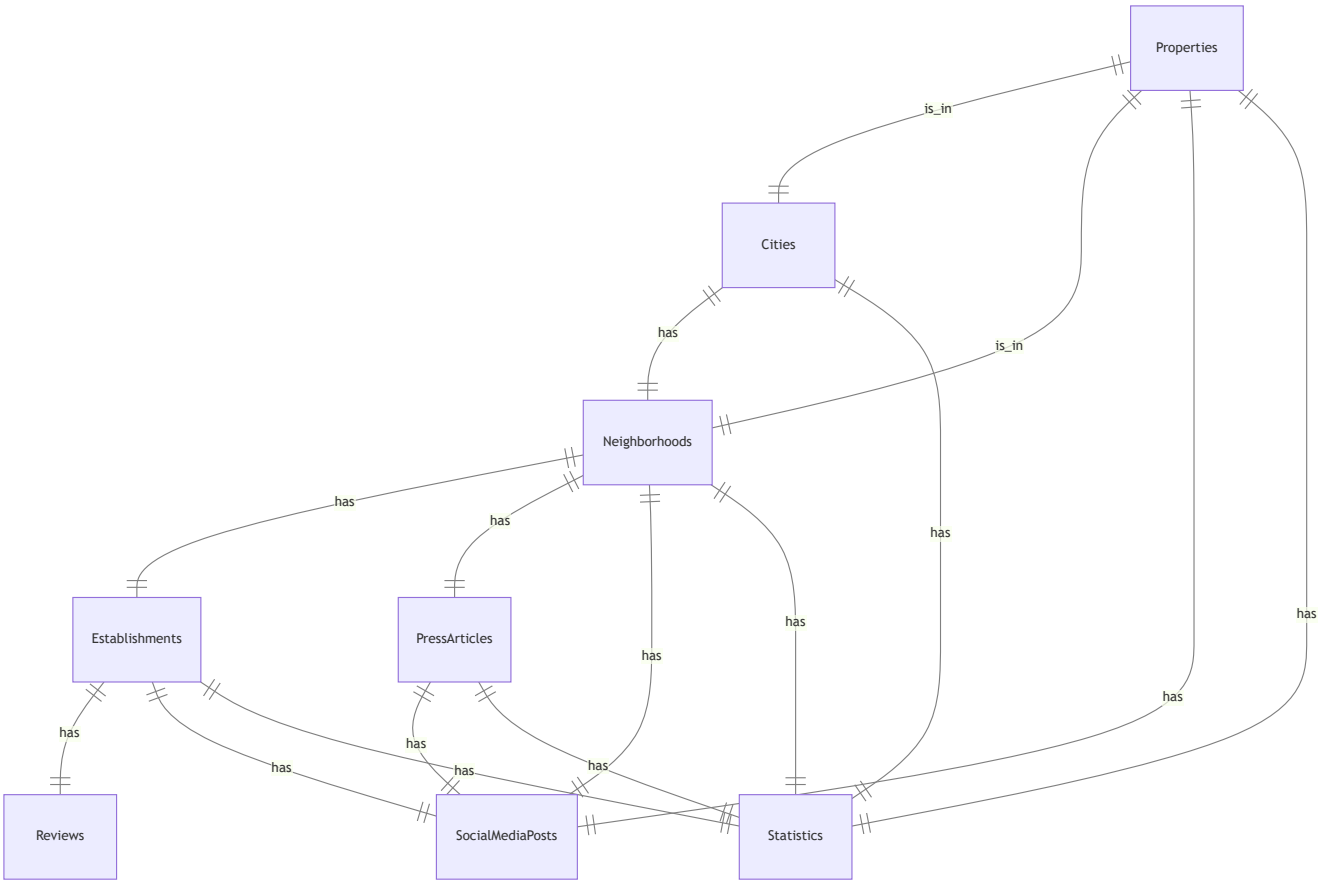
Analyse des Données

Les données recueillies sont analysées pour produire des statistiques sur le marché immobilier. Ces statistiques peuvent inclure des informations sur les tendances des prix, les types de propriétés les plus courants, et d'autres informations utiles pour les acheteurs, les vendeurs, et les investisseurs immobiliers.

AD

Base de Données

La base de données est utilisée pour stocker les informations recueillies et les résultats de l'analyse des données. La base de données comprend les tables suivantes :



Nom de la Table	Attributs
Propriétés	ID, Type, Ville, Quartier, Prix, Date de publication, URL de l'image, Nombre d'images, Nombre de chambres, Nombre de salons, Nombre de salles de bain, Numéro d'étage, Surface habitable, Surface totale, Âge de la propriété, URL de l'annonce

Nom de la Table	Attributs
Villes	ID, Nom
Quartiers	ID, Nom, ID de la ville
Établissements	ID, Nom, Type, Évaluation, ID du quartier
Avis	ID, Sentiment, Texte, ID de l'établissement
Articles de Presse	ID, Titre, Sentiment, Texte, Date de publication, ID du quartier
Posts sur les Réseaux Sociaux	ID, Sentiment, Texte, Date de publication, ID du quartier, ID de la ville, ID de l'établissement, ID de la propriété, ID de l'article de presse
Statistiques	ID, Date, Type de Statistique, Valeur, ID de la Ville, ID du Quartier, ID de l'Établissement, ID de la Propriété

API

L'API est utilisée pour fournir un accès aux informations et aux statistiques stockées dans la base de données. L'API peut être utilisée par divers clients, y compris une application web, une application mobile, ou d'autres services. AD

Interface Utilisateur

L'interface utilisateur permet aux utilisateurs d'interagir avec l'application. Elle peut inclure des visualisations de données, des outils de recherche, et d'autres fonctionnalités pour aider les utilisateurs à comprendre le marché immobilier. AD

Technologies Utilisées

Le projet utilise une variété de technologies, y compris Python pour le web scrapping et l'analyse des données, ...