Documentation **Robotic Watch App**

Myriam Achour

Juin – Août 2025

Table des matières

1. Résumé du Projet

- Nom de l'application : Robotic watch app
- Objectif de l'application : Application de veille technologique offrant une vision globale du secteur de la robotique humanoïde. Elle centralise les informations disponibles sur

Internet, fonctionne automatiquement et permet de visualiser les données collectées dans une base alimentée par des agents IA à chaque requête effectuée.

- Stack principal :
    - Backend : Python
    - Frontend : Streamlit
    - Agents IA : LangChain, LangGraph, OpenAI, Tavily Search API
    - Base de données : CSV

2. Structure du Projet

```
robotic-watch-app/
├── requirements.txt
├── documentationApp.md
├── src/
│       ├── app.py
│       ├── pages/
│               ├── .env
│               ├── __init__.py
│               ├── Home.py
│               ├── web_scrap_file.py
│               ├── website_retrieval.py
└── data/
        ├── images/
        └── csv files
```

3. Installation et lancement

Prérequis :

- Python >= 3.12
- Clés API nécessaires pour :
    - OpenAI
    - TavilySearch

Etapes d'installation et de lancement depuis le terminal de commande :

```
>> git clone https://github.com/MyriamA6/robotic-watch-app
>> python -m venv venv
>> source venv/bin/activate
>> pip install -r requirements.txt
>> cd C:\Users\jusqu\a\robotic-watch-app\src
>> streamlit run app.py
```

Sur Raspberry Pi 5 :

```
>> git clone https://github.com/MyriamA6/robotic-watch-app
>> python -m venv venv
>> source venv/bin/activate
>> pip install -r requirements.txt
>> cd C:\Users\jusqu\a\robotic-watch-app\src
>> Des installations supplémentaires peuvent être requises
>> python -m streamlit run app.py
```

4. Fonctionnement global

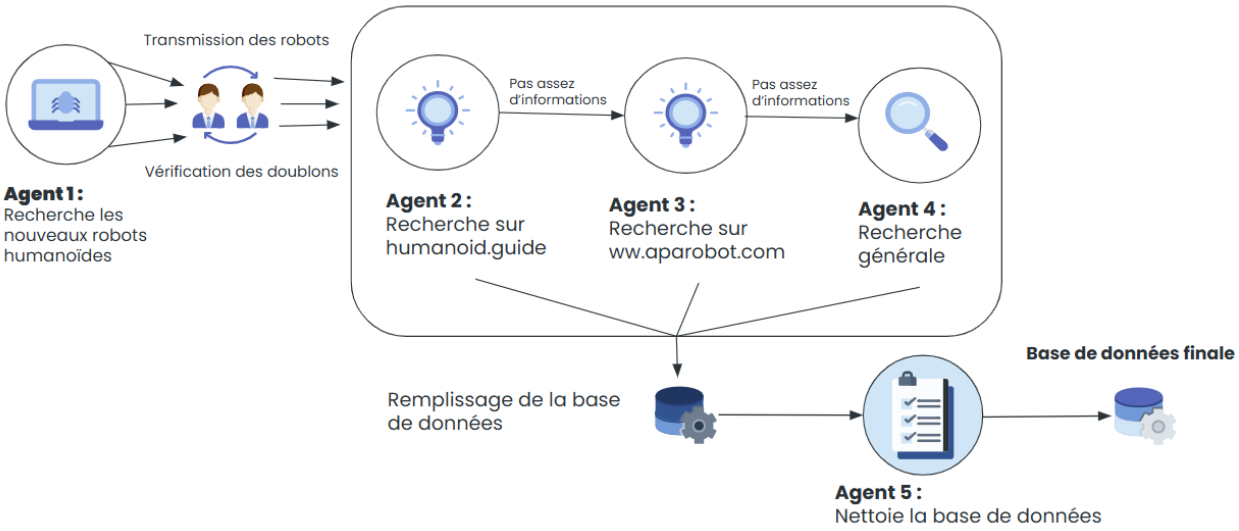
Pour une description détaillée de l'interface utilisateur et des fonctionnalités, consultez le guide d'utilisation fourni avec l'application.

5. Agents IA

5.1. Les Agents de recherche d'informations sur les humanoïdes

Afin de rendre les analyses à la fois dynamiques et complètes, des agents IA basés sur **LangChain** et **LangGraph** ont été intégrés. Ils constituent un système automatisé de récupération des robots humanoïdes récemment dévoilés sur le marché, ainsi que des informations relatives à leurs caractéristiques et spécifications techniques.

Ce système codé dans le fichier `web_scrap_file.py` est constitué ainsi :



AGENT 1 :

Avant de rechercher précisément les spécifications techniques pour chaque robot, il faut d'abord retrouver ces robots récemment révélés sur le marché. C'est là que l'Agent 1 (**`find_robots_agent`**) intervient :

➔ Il est guidé par un prompt pour chercher les plus récentes sorties sur les sites suivants :

- www.aparobot.com
- humanoidroboticstechnology.com
- www.linkedin.com
- x.com

Deux modes de recherche sont possibles :

1. **Recherche automatique** : l'agent effectue un balayage complet des sites listés.
2. **Recherche manuelle** : l'utilisateur indique directement dans le fichier texte `latest_humanoid_robots.txt` les robots à ajouter, au format suivant :

```
[{"name":"DR.01","company":"Deeprobotics"}, {"name":"L7","company":"RobotERA"}]
```

Important : à chaque lancement d'une nouvelle recherche, la sauvegarde des noms et entreprises trouvés précédemment est **écrasée** par les nouveaux résultats.

Système de vérification de doublons :

Après la recherche initiale, chaque robot identifié (nom + entreprise) est transmis à un système de vérification de doublons. Ce processus nécessite une **intervention humaine** pour valider ou non la similarité.

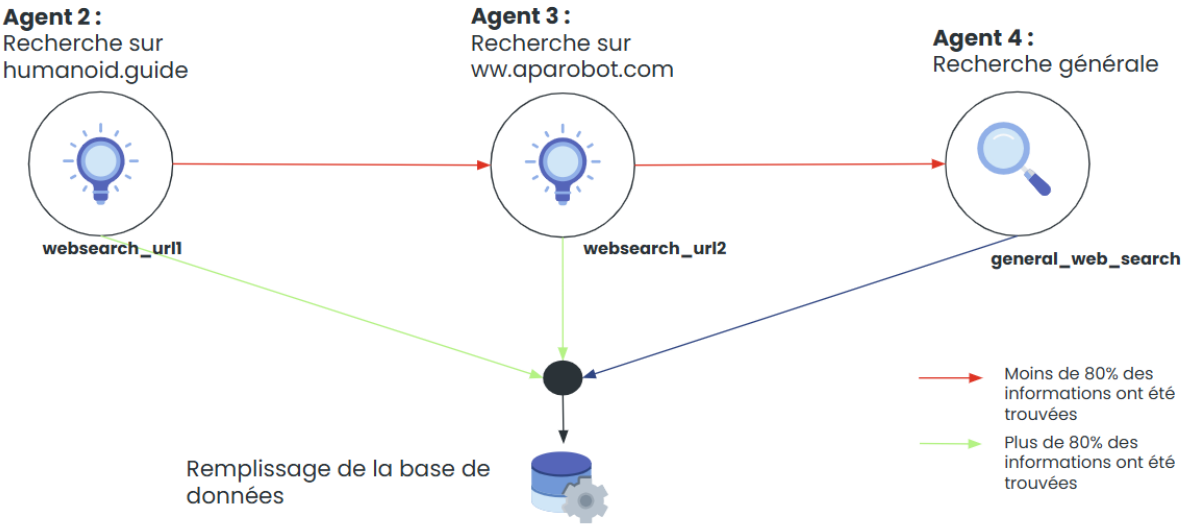
Deux cas de figure sont possibles :

- ➔ **Aucun doublon détecté** → la recherche des informations sur le robot se poursuit automatiquement.
- ➔ **Doublon potentiel détecté** → l'utilisateur décide si :
  - ➔ le robot est bien similaire (possibilités : d'ignorer ou de mise à jour des informations existantes), ou
  - ➔ le robot est distinct (ajout direct dans la base).

Cela permet d'éviter la détection de « faux positifs » comme Unitree R1 et Unitree G1 qui sont considérés similaires mais ne le sont pas.

AGENTS 2,3,4 :

Pour la collecte automatique des informations, un **système multi-agents conditionnel basé sur LangGraph** est utilisé. Il est composé de **trois nœuds** représentant les étapes successives de la recherche.:



Pour pouvoir sauvegarder les informations des robots plus facilement, une classe **Pydantic** `RobotSpec` est disponible.

Dans `LangGraph`, chaque agent doit manipuler un état courant (`GraphState`), qui correspond ici à l'avancement de la recherche sur le robot en cours d'analyse. Ses composants principaux sont :

- *`RobotSpec`* **robot**: instance contenant les informations actuelles du robot.
- *`Float`* **completion_rate** : taux de remplissage des informations du robot.

- *Str* **url** : url du site sur lequel les recherches doivent être poursuivies par le nœud suivant.

Tous les agents disposent également d'un outil commun pour accéder au contenu d'une page web à partir de son URL : `fetch_full_webpage`

Les **Agents 2 et 3 (`web_scraper`)** de ce système utilisent la même fonction `humanoid_guided_websearch`, dont le rôle est le suivant:

1. **Localiser l'URL exacte** de la page d'un robot sur un site donné.
2. **Effectuer une recherche approfondie** des informations disponibles sur ce robot.
    a. Pour effectuer cette recherche un prompt, `prompt_webscrape`, de contexte le plus précis possible a été créé et amélioré en utilisant une IA.

Afin de réduire l'utilisation des tokens OpenAI, des **arêtes conditionnelles** sont intégrées entre chaque nœud.

- Le système poursuit la recherche uniquement si le `completion_rate` (taux de complétion) est **strictement inférieur à 80 %**.
- Au-delà de ce seuil, le robot est considéré comme suffisamment renseigné et renvoyé tel quel.

AGENT 4 :

Si les deux premiers agents n'ont pas permis de collecter suffisamment d'informations, le robot est transmis à l'**Agent 4 (`web_search_worker`)**. Cet agent dispose d'un outil supplémentaire, `Tavily Search`, qui lui permet d'effectuer des recherches plus larges sur Internet. À travers la fonction `general_websearch`, il peut explorer librement divers sites afin de compléter au maximum les informations manquantes sur le robot.

Une fois la recherche effectuée le robot est ajouté dans la base de données `humanoid_data.csv`, ce pour éviter que la recherche soit reprise à zéro en cas de bug.

La base de données est ensuite traitée pour enlever les doublons identiques.

AGENT DE NETTOYAGE :

Un dernier agent (`cleaning_agent`) est chargé d'**harmoniser et de normaliser** le contenu de la base de données. Son rôle est d'assurer la cohérence et la lisibilité des informations collectées. Concrètement, il effectue notamment :

- La **standardisation des termes** : par exemple, si un élément est désigné à la fois par *Deep Learning* et *DL* dans une même colonne, tous les termes sont uniformisés sous la même appellation (*DL*).
- Le **regroupement par catégories** de certaines informations, comme les différents types de caméras.
- Divers autres traitements visant à éliminer les incohérences ou doublons sémantiques.

Les données sont enfin sauvegardées dans `humanoid_data_cleaned.csv` pour effectuer les analyses ensuite.

Pour lancer la recherche de robot se référer à la partie « **Administrateur** » du **guideUtilisateur.md**.

Difficultés/ Problèmes potentiels :

1. **Qualité de la recherche de nouveaux robots** : Le système de recherche ne détecte pas toujours les nouveautés du marché : il peut parfois renvoyer d'anciens robots ou en oublier certains récents.
    a. Une approche basée sur un nombre minimum de robots à identifier a été testée, mais elle s'est révélée trop restrictive : l'agent se limite alors artificiellement à ce seuil.
2. **Limitation de l'interface Streamlit** : L'utilisation de l'interface Streamlit pour gérer la recherche de nouveaux complexifie le code et provoque de nombreux bugs, dû à l'utilisation de boutons dont le fonctionnement oblige un rechargement complet de l'application.
    a. Une sauvegarde de l'état de session a été mise en place pour contourner en partie ce problème, mais elle ne résout pas toutes les erreurs.
    b. En cas de blocage critique lors de l'ajout des robots dans la base, une **version alternative utilisable en terminal** est disponible dans le fichier **terminal_web_scrap_file.py**.
3. **Limites du nettoyage de la base de données :** Le système de nettoyage automatique n'est pas toujours optimal. Certaines incohérences persistent et pourraient être mieux gérées. Une amélioration de ce système ou du prompt associé pourrait être envisagée. (voir **prompt_cleaning** de web_scrap_file.py)

5.2. Agent de recherche de données financières

Dans le fichier **companies_webscrape.py**, se trouve un dernier agent chargé de collecter les données financières des entreprises associées aux robots présents dans la base de données. Celui-ci dispose également des outils `Tavily Search` et `fetch_full_webpage`.

Lancement de la recherche :

- Via l'interface web_scrap_file de l'application, en cliquant sur le bouton correspondant.
- En indiquant la lettre **« c »** lorsque cela est demandé via le terminal dans le cas du lancement du fichier **terminal_web_scrap_file.py**.

Remarques :

- La recherche ne doit pas être lancée régulièrement car elle est **longue à exécuter**, l'agent traitant toutes les entreprises présentes dans la base.
- Les données financières pouvant évoluer indépendamment de la sortie d'un nouveau robot, il a été choisi de **mettre à jour ces informations séparément**.

Gestion des bases de données

- Toutes les informations sont sauvegardées dans le fichier **companies_data.csv**. Il s'agit de la base « dynamique ». Elle fournit par exemple la **somme totale des levées de fonds** depuis la création de chaque entreprise traitée, mais **pas le détail par année**.

- Une base de données **statique** conserve l'historique des levées de fonds des entreprises (données collectées manuellement via le site `Tracxn.com`) pour analyser les tendances sur plusieurs années.

Vigilance :

- En raison des limites d'utilisation de l'API OpenAI, le processus est conçu pour **ignorer une entreprise en cas de bug ou d'itérations excessives**, afin de ne pas bloquer le traitement global.
- Cette approche n'est pas optimale car certaines entreprises peuvent ne pas être mises à jour, mais elle garantit la continuité de l'exécution de l'agent.

6. Bases de données

- `Countries_co.csv` : contient un certain nombre de pays ou régions du monde avec leur coordonnées pour pouvoir les afficher sur une carte de l'interface Streamlit par la suite.
- `Humanoid_data.csv` : contient l'ensemble des informations « brutes » des robots humanoïdes, récupérées.
- `Humanoid_data_cleaned.csv` : contient ces mêmes informations, uniformisées et nettoyées.
- `Companies_data.csv` : Contient la base de données financières des entreprises ayant conçu des robots, citées dans la base de données `Humanoid_data_cleaned.csv` .
- `Humanoid_company_market_analysis.csv` : Contient les données statiques sur les entreprises remplies à partir du site `Tracxn.com`.

7. Interface Streamlit

Sur l'interface Streamlit deux pages sont présentes :

- `Home.py` (accessible via le logo *Renault Group*)
- `Web_scrap_file.py` (cachée) : utilisation décrite dans le **guide d'utilisation**

Page Home.py

Cette page effectue un **prétraitement des données**, puis génère un ensemble d'analyses et de graphiques sur les robots humanoïdes et les entreprises enregistrées dans les bases de données.

Dynamisme :

La page intègre des zones de **JavaScript** permettant de faire défiler automatiquement certaines parties de l'application :

- Slides sur les **caméras**
- Slides sur les **challenges**
- Slides sur les **informations récentes**

Chaque section affichée dispose d'un **titre (subheader) avec une ancre**. Ces ancres sont stockées dans une liste **anchors_ids**, et leur ordre dans la liste correspond à l'ordre d'affichage à l'écran.

Pour ajouter une nouvelle section, il faut **ajouter son ancre à cette liste**.

Pour des raisons d'affichage, chaque section comporte un **subheader vide** avec l'ancre, suivi du subheader réel sans ancre.

Gestion des temps d'affichage

Les temps d'affichage des différentes sections peuvent être modifiés selon vos besoins :

1. **`newsDelay` (JS)** : durée d'affichage d'une slide de la section *Information (News)*, en millisecondes.
2. **`percepDelay` (JS)** : durée d'affichage d'une slide de la section sur les **capteurs de perception**, en millisecondes.
3. **`challengeDelay` (JS)** : durée d'affichage d'une slide de la section *Challenges*, en millisecondes.

Le dernier script JavaScript gère l'affichage dynamique global :

- **`display_time_seconds`** : durée d'affichage d'une section référencée par une ancre dans la liste anchors_ids, en secondes.
- Les variables **`specialAnchor/2/3/4`** correspondent aux sections affichées plus longtemps que les sections classiques, définies par **`specialDelay`** (en millisecondes).
- Pour ajouter une section affichée plus longtemps :
    1. Référencer la section dans une nouvelle variable specialAnchorX.
    2. Ajouter dans la ligne `const nextDelay` la condition :

       `|| targetId === specialAnchorx`

8. La recherche des informations récentes sur Internet

Le fichier **`website_retrieval.py`** est dédié à la collecte des **informations les plus récentes** sur la robotique humanoïde.

- Le script utilise du **web scraping classique** avec les bibliothèques **BeautifulSoup** et **requests**.
- Actuellement, les données sont extraites du site humanoidsdaily.
- Remarque : le site n'a pas été mis à jour depuis le 26 juin. Il pourrait être nécessaire de **changer de source** pour disposer d'un site actualisé et compatible avec le web scraping.

9. Points de vigilance

Limitations de l'application :

- Les quotas mensuels de chaque API :
    - OpenAI : pas de plan gratuit
    - Tavily Search API : 1000 crédits gratuits
- La relance automatique de l'application à chaque appui de bouton.
- Le nettoyage de la base de données, dans le but d'uniformiser les informations pour les analyses qui suivent.

10. Améliorations possibles

1. Utilisation des agents IA :

Il serait intéressant de tester d'autres modèles OpenAI ou d'autres LLM pour optimiser les performances. Les modèles testés jusqu'à présent sont :

- **Ollama** : Gemma, Mistral, Llama 2, Llama 3 → forte utilisation du processeur, résultats lents et parfois incorrects.
- **Gemini** : Gemini-2.5-flash-preview-04-17 et autres versions gratuites.
- **GPT-4o et GPT-4o mini** → meilleurs résultats à ce jour.
    - Les modèles **O1** et **O1-Pro** sont plus performants mais également plus coûteux.

Disposer d'un accès à l'interface pour suivre la consommation de tokens permettrait de choisir un modèle offrant le meilleur compromis qualité/prix.

Changer d'interface web :

L'utilisation de Streamlit impose un **rechargement complet de la page** lors des clics sur les boutons.
→ Remplacer Streamlit par un autre framework pourrait permettre une **interaction plus fluide**, sans rechargement intégral de l'application.

Améliorer la recherche de données financières :

Pour améliorer la recherche de données financières sur les entreprises, en particulier les levées de fonds qu'elles ont réalisées depuis **leur création**, il faudrait **obtenir un accès à des plateformes spécialisées disposant d'une API**. Les plus intéressantes trouvées sont majoritairement payantes. Parmi celles disponibles, il y a :

1. Tracxn -> Modèle premium/commercial access (prix non défini)
2. Crunchbase
3. Pitchbook

Améliorer le nettoyage de la base de données :

Une **révision et amélioration du prompt** de l'agent de nettoyage pourrait résoudre les problèmes de cohérence et d'hétérogénéité dans la base, permettant d'obtenir des données plus fiables et uniformisées.

11. QRcode et sources utilisées

La dernière section de l'application présente un QR code. Une fois scanné, il mène vers un document contenant la liste des sources utilisées pour l'application. Ce document est rempli manuellement et peut être mis à jour.

Remarque : vous pouvez modifier l'URL du document en mettant à jour la variable `sheet_url` dans le code avec l'URL d'un Google Docs de votre choix.