

Manuel d'installation



Manuel d'installation

Prototype d'agent conversationnel pour les séniors

Les informations d'identification du document

Référence du document :	D4
Version du document :	1.01
Date du document :	20 juin 2025
Auteur(s) :	Marc Dieudonné Elie Beyeler Adrien Slifirski

Les éléments de vérification du document

Validé par :	Nom de l'encadrant
Validé le :	20/6/25
Soumis le :	20/6/25
Type de diffusion :	Document électronique (.odt)

Confidentialité :

Mots clés : Guide d'installation de l'agent conversationnel pour les Séniors.

1.Sommaire

1.Sommaire	4
2.Introduction (ou préambule)	4
2.1.Objectifs et méthodes	4
2.2.Documents de référence	5
3.Guide de lecture	5
4.Concepts de base	5
5.Installation du matériel	5
5.1.Installation de python	6
5.2.Installation de Ollama	6
6.Paramétrage du système	7
7.Installation du logiciel	7
8.Paramétrage du logiciel	8
9.Autres informations	8
10.Annexes	8
11.Glossaire	9
12.Références	9
13.Index	9

2.Introduction

Ce manuel fournit les étapes nécessaires à l'installation de l'agent conversationnel conçu pour les seniors. Il détaille l'installation du matériel, du logiciel, ainsi que le paramétrage pour un usage optimal.

2.1. Objectifs et méthodes

Fournir un guide pas-à-pas pour permettre à un technicien ou utilisateur avancé d'installer l'application sur une machine locale ou un serveur personnel. La méthode repose sur une architecture client-léger (navigateur) et un back end local (Serveur python + Large Langage Model via l'outil Ollama).

2.2. Documents de référence

- Maquettes Figma
- Cahier des charges du projet
- Documentation WeatherAPI
- Spécification du LLM Llama 2 (Meta AI)

3.Guide de lecture

- Techniciens : suivre les étapes d'installation et de déploiement.
- Utilisateurs confirmés : paramétrage du logiciel et des données.
- Développeurs : comprendre la configuration du back end et les dépendances.

4.Concepts de base

- **Agent conversationnel** : application web simulant une conversation.
- **localStorage** : mécanisme de stockage dans le navigateur pour les données utilisateurs.
- **LLM** : Modèle de langage génératif (Llama 2).

5.Installation du matériel

Prérequis :

- PC avec navigateur moderne (Chrome, Firefox, Edge)
- Connexion Internet

Le chatbot nécessite deux éléments logiciels externe afin de fonctionner correctement :

- Premièrement, une version de python supérieur ou égale à python 3.11.9
- Le logiciel Ollama, paramétré de manière à utiliser le LLM de Meta llama2.

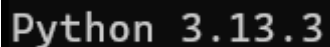
Ollama nécessite une action afin de pouvoir utilisé llama2, qui sera détaillé dans la partie suivante.

5.1. Installation de python

Si vous possédez déjà une version de python supérieur à python 3.11, vous pouvez passer cette étape.

Note : si vous souhaitez connaître votre version actuelle de python, suivez les étapes indiquées [ici](#).

1. Allez sur <https://www.python.org/downloads/> et télécharger la dernière version stable de python (à l'heure ou ce guide est écrit il s'agit de la version 3.13.4) pour Windows.
2. Une fois le téléchargement terminé, lancer l'installation à l'aide de l'installateur (*python-3.13.4-amd64.exe*).
 - . **IMPORTANT** : Vérifiez bien que la case « *Add python to PATH* » est cochée. Faites-le si ce n'est pas le cas.
3. Vérifiez que python est bien installé en effectuant les actions suivantes :
 - . Appuyez simultanément sur les touches Windows + R.
 - . Tapez « cmd ».
 - . Si python est correctement installé, vous devriez voir un affichage correspondant à la version de python que vous avez installé (ici Python 3.13.3) :

A screenshot of a terminal window with a black background and white text. The text displayed is "Python 3.13.3".

5.2. Installation de Ollama

Si vous possédez déjà une version d'Ollama, vous pouvez passer cette étape.

Note : si vous souhaitez connaître votre version actuelle de Ollama, suivez les étapes indiquées [ici](#).

1. Allez sur <https://ollama.com/download> et téléchargez la dernière version pour Windows.
2. Une fois le téléchargement terminé, lancez l'installation en double-cliquant sur l'installer *OllamaSetup.exe* et suivez les étapes de l'installation.
3. Vérifiez l'installation en réalisant les étapes suivantes :
 - . Appuyez simultanément sur les touches Windows + R.
 - . Tapez « cmd ».
 - . Dans l'invite de commande ainsi ouvert, tapez la commande suivante :
Ollama list
 - . Si Ollama est correctement installé, vous devriez voir un affichage correspondant à la version que vous avez installé tel que celui-ci :

NAME	ID	SIZE	MODIFIED
mistral:latest	f974a74358d6	4.1 GB	5 days ago

6.Paramétrage du système

Afin de pouvoir utiliser le chatbot, il est essentiel de charger le modèle de langue initial que nous allons utiliser. Pour cela, effectuez les étapes suivantes :

1. Commencez par ouvrir l'invite de commande en utilisant Windows + R, puis en tapant « cmd » dans la fenêtre qui apparaît.
2. Tapez la commande suivante : *Ollama pull llama2*. Vous devriez voir le lancement du téléchargement du modèle suite à cela.

Note : Le modèle est disponible dès la fin de cette étape. Ainsi, dès lors que l'étape d'installation du logiciel est terminée (partie suivante) le chatbot devrait être utilisable. Il est recommandé de tester le chatbot avant de commencer son entraînement dans la partie 8.

7.Installation du logiciel

Rendez-vous sur <https://github.com/Adrien-Sli/TER> . Une fois présent sur cette page, deux solutions s'offrent à vous :

1. Clonez le dépôt dans le répertoire que vous souhaitez. L'ensemble des fichiers nécessaires au fonctionnement de l'application seront présent.
 2. Téléchargez le répertoire sous la forme d'un fichier compressé .zip (bouton <> Code → Download zip).
- . Une fois l'archive téléchargé, extrayez le dossier du document .zip et déplacez-le à l'emplacement de votre choix (le dossier devrait être nommé *TER-main*).

Une fois le dossier présent à l'emplacement que vous souhaitez, vous pouvez dès lors commencer à utiliser l'agent conversationnel. Pour cela, assurez-vous qu'à la fois Ollama et le serveur local Flask sont actifs. Pour ce faire, ouvrez deux invite de commande distinct à l'aide des raccourcis précédents (Windows + R puis « cmd »). Dans le premier, taper la commande

Ollama serve

Vous devriez avoir un affichage indiquant qu'ollama est actif tel que celui-ci :

```
time=2025-06-18T16:21:39.046+02:00 level=INFO source=routes.go:1235 msg="server config" env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:4096 OLLAMA_DEBUG:INFO OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://127.0.0.1:11434 OLLAMA_INTERRUPT_GPU:false OLLAMA_KEEP_ALIVE:5m0s OLLAMA_KV_CACHE_TYPE: OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:C:\\Users\\adrie\\.ollama\\models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NEW_ENGINE:false OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost:* http://127.0.0.1 https://127.0.0.1 http://127.0.0.1:* https://127.0.0.1:* http://0.0.0.0 https://0.0.0.0 http://0.0.0.0:* https://0.0.0.0:* app://* file://* tauri://* vscode-webview://* vscode-file://*] OLLAMA_SCHED_SPREAD:false ROCR_VISIBLE_DEVICES:]"
time=2025-06-18T16:21:39.102+02:00 level=INFO source=images.go:476 msg="total blobs: 10"
time=2025-06-18T16:21:39.106+02:00 level=INFO source=images.go:483 msg="total unused blobs removed: 0"
time=2025-06-18T16:21:39.108+02:00 level=INFO source=routes.go:1288 msg="Listening on 127.0.0.1:11434 (version 0.9.1)"
time=2025-06-18T16:21:39.108+02:00 level=INFO source=gpu.go:217 msg="looking for compatible GPUs"
time=2025-06-18T16:21:39.108+02:00 level=INFO source=gpu_windows.go:167 msg="packages count=1"
```

Dans le second, veuillez indiquer le chemin menant à l'emplacement que vous avez choisi pour le répertoire à l'aide de la commande suivante :

Cd C:/Users/Chemin/Vers/le/repertoire/TER-main

Une fois dans le répertoire principal, placez-vous dans le dossier *back* à l'aide de *cd back/*

Effectuer la commande *pip install flask flask-cors requests*

Dès lors, tapez la commande *Python server.py* pour lancer le serveur python.


```

Project root: C:\Users\adrie\OneDrive\Bureau\M1\TER
Frontend directory: C:\Users\adrie\OneDrive\Bureau\M1\TER\front
HTML directory: C:\Users\adrie\OneDrive\Bureau\M1\TER\front\HTML
Static files available at: C:\Users\adrie\OneDrive\Bureau\M1\TER\front
* Serving Flask app 'server'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8000
* Running on http://192.168.1.108:8000
Press CTRL+C to quit
* Restarting with stat
Project root: C:\Users\adrie\OneDrive\Bureau\M1\TER
Frontend directory: C:\Users\adrie\OneDrive\Bureau\M1\TER\front
HTML directory: C:\Users\adrie\OneDrive\Bureau\M1\TER\front\HTML
Static files available at: C:\Users\adrie\OneDrive\Bureau\M1\TER\front
* Debugger is active!
* Debugger PIN: 107-860-172

```

Note : le rôle du serveur est de coordonner l'application et de mieux gérer les échanges de requêtes avec le modèle utilisé.

Une fois ces deux étapes effectuées, aller dans votre navigateur et tapez « *Localhost :8000* ».

Vous devriez alors voir un formulaire apparaître dans votre navigateur. Ce formulaire est essentiel pour la personnalisation des réponses, aussi remplissez-le avec soin. Dès que vous en aurez terminé, vous devriez voir la page d'accueil de l'agent ; vous pouvez dès à présent interagir avec lui !

Pour un usage optimal dans le cadre de ce projet, nous avons entraîné le modèle sur les risques domestiques ; pour cela, référez-vous à la partie qui suit.

8. Paramétrage du logiciel

Afin d'accroître les compétences du modèle concernant les risques domestiques, nous avons confectionné plusieurs fichiers de données permettant à l'agent conversationnel d'être à la fois plus précis et fiable sur les questions relatives à certains risques.

Il s'agit du fichier *training.modelfile* contenu dans le dossier *Data*. Pour entraîner le modèle, effectuez les actions suivantes :

1. Ouvrez un invite de commande et placez-vous dans le dossier *Data* du répertoire du projet (voir [ici](#))
2. Tapez la commande *Ollama create nom_du_modele -f training.modelfile*

3. Vous devriez voir apparaître une succession d'instruction dans la console avec *success* qui les concluent :

```
$ ollama create test -f training.modelfile
using existing layer sha256:ff82381e2bea77d91c1b824c7afb83f6fb73e9f7de9dda631bcdca564aa5435
using existing layer sha256:43070e2d4e532684de521b885f385d0841030efa2b1a20bafb76133a5e1379c1
using existing layer sha256:1ff5b64b61b9a63146475a24f70d3ca2fd6fdeec44247987163479968896fc0b
using existing layer sha256:62014a9915cf76c24dcf6660f0b4fdf16cfd1019fffa09785c7b576c0fbcd3f
using existing layer sha256:6bb7ade3d508964d26dd11c22cdc3167deef5bb83d5956d69cf1c0b1b2d343
using existing layer sha256:1180af37f68d5b2bd204f4e8f9f4d4d545ed16d3a230db1542add6a2a9b6b31c
writing manifest
success
```

Note : Les actions précédentes sont les actions permettant d'entraîner le modèle. Il est ainsi tout à fait envisageable d'effectuer ces actions avec d'autres fichiers qui semblent pertinent pour accroître les capacités de réponse du modèle initiale sur d'autres risques ou d'autres éléments.

9. Autres informations

Pour les utilisateurs de linux, suivez les étapes suivantes :

1. `curl -fsSL https://ollama.com/install.sh | sh`
2. `ollama --version`
3. `sudo systemctl enable ollama`
4. `sudo systemctl start ollama`
5. `ollama pull llama2`
6. `ollama run llama2 "Bonjour, comment ça va ?"` note : cette commande vise à tester la présence du modèle.
7. `pip install flask flask-cors requests`
8. `python server.py`

10. Glossaire

- **localStorage** : stockage local persistant
- **LLM** : modèle de langage de grande taille

11. Références

- MDN localStorage
- WeatherAPI
- Documentation interne projet

12. Index

Agent conversationnel,
localStorage,
Llama 2,
WeatherAPI,
accessibilité