

warning "Mise à jour du backend (octobre 2026)" L'inférence GitHub Models a été retirée le 30 juillet 2026. Les références ci-dessous sont conservées uniquement comme contexte historique pendant la migration de ce lab. Utilisez le Lab 013 ? Backends LLM gratuits pour configurer Foundry Local ou Ollama avant de continuer.

Lab 027 : Patterns de mémoire d'agent

Niveau : L200 Parcours : Pro Code / SK Durée : ~35 min ? Coût : Gratuit (local)

Ce que vous apprendrez

Les 4 types de mémoire d'agent et quand utiliser chacun Mémoire en contexte (gestion de l'historique de conversation) Mémoire externe (magasin de vecteurs + SQLite pour les données structurées) Mémoire épisodique (ce qui s'est passé dans les sessions précédentes)

Construire un agent conscient de la mémoire qui retient les préférences utilisateur

Introduction

Sans mémoire, chaque conversation d'agent repart de zéro. Un agent de service client qui oublie le nom du client en pleine conversation, ou un assistant qui ne peut pas se rappeler de ce que vous avez décidé la semaine dernière, crée une expérience frustrante.

Les quatre types de mémoire :

Type | Stockage | Durée de vie | Exemple | En contexte | Fenêtre de contexte LLM | Requête unique | Historique de conversation | Sémantique externe | Base vectorielle | Persistante | Anciens tickets de support | Structurée externe | Base SQL | Persistante | Préférences utilisateur, historique de commandes | Épisodique | Clé-valeur / fichier | Session ou persistante | « Lors de la dernière session, nous avons discuté de X » |

Prérequis

Python 3.11+ pip install openai chromadb GITHUB_TOKEN configuré

tip "Données exemple incluses" Ce lab utilise orders.csv et products.csv du dépôt. [diagram or code sample]

Exercice du lab

Étape 1 : Mémoire en contexte (historique de conversation)

La mémoire la plus simple ? gardez simplement l'historique des messages. Le défi : les fenêtres de contexte ont des limites.

[diagram or code sample]

Étape 2 : Mémoire structurée avec SQLite

Pour les faits nécessitant de la précision (préférences, commandes, paramètres), utilisez une base de données, pas la fenêtre de contexte.

[diagram or code sample]

Étape 3 : Mémoire épisodique sémantique avec ChromaDB

ChromaDB est une base de données vectorielle gratuite et locale ? pas besoin de Docker.

[diagram or code sample]

Étape 4 : Agent conscient de la mémoire

Combinez les trois types de mémoire en un seul agent :

[diagram or code sample]

Guide de décision pour les patterns de mémoire

[diagram or code sample]

Prochaines étapes

Le RAG agentique utilise la mémoire de récupération : ? Lab 026 ? RAG agentique SK dispose d'abstractions de mémoire intégrées : ? Lab 023 ? Plugins et mémoire SK

Kit d'accompagnement

Le build du site génère des téléchargements PDF et ZIP stables. Le ZIP contient le guide du lab et tous les fichiers d'accompagnement déclarés.

[Télécharger le PDF généré](#) [Télécharger le ZIP généré](#)

Supporting file: CHEATSHEET.md

Fichiers d'accompagnement: Lab 027 : Patterns de mémoire d'agent

Commencez par le parcours local ou gratuit documenté. Consignez les hypothèses et les preuves de validation. Gardez les services hébergés facultatifs sauf exigence explicite du lab.

Supporting file: README.md

Lab 027 : Patterns de mémoire d'agent

Utilisez ces fichiers avec le guide du lab et le kit généré.