Implémenter l’IA générative : agents, RAG et fine-tuning

Programme *(Mis à jour le 09/09/2026)*

1. Qu’est-ce qu’« implémenter » de l’IA générative?

- Les trois couches : infrastructure, middleware, applicatif — qui fait quoi
- Ce qu’on n’implémente jamais : pré-entraînement, alignement, RLHF
- Les quatre leviers d’adaptation : prompt, RAG, outils/agent, fine-tuning
- Rappel express du fonctionnement d’un LLM : tokens, contexte, prédiction, limites structurelles
- Ordres de grandeur : coût, latence, empreinte, effort de maintenance
- Mise en pratique : le même besoin métier traité successivement par les quatre leviers — constat des écarts de résultat, de coût et d’effort.

2. Le modèle comme brique logicielle

- Modèle ouvert en local (Ollama) ou modèle propriétaire via API : le même code
- Choisir un modèle : taille, quantization, empreinte mémoire, licence, langue
- Les paramètres qui comptent : température, top-p, fenêtre de contexte, graine
- Sortie structurée : imposer un schéma JSON et fiabiliser l’intégration
- Mesurer : tokens consommés, latence, coût par appel
- Mise en pratique : installation d’un modèle local, premiers appels en Python, extraction structurée d’une note de frais en JSON validé, mesure comparée local / API.

3. Donner des outils au modèle

- Le tool calling : ce que le modèle décide, ce que le code exécute
- Décrire un outil pour qu’il soit correctement appelé
- De l’appel d’outil à l’agent : la boucle perception ? décision ? action
- Les modes d’échec : boucles infinies, arguments inventés, outils mal décrits
- Mise en pratique : écriture d’un agent complet **sans framework** , en une quarantaine de lignes de Python — puis analyse de ses limites.

4. Agents : frameworks, état et garde-fous

- Ce qu’apporte un framework : état, mémoire, reprise, interruption, traces
- LangGraph : graphe d’états, nœuds, agent préconstruit, points d’interruption
- Mémoire courte et longue, persistance des conversations
- Garde-fous indispensables : validation humaine, budget d’itérations, périmètre des outils
- Code ou low-code : LangGraph face à n8n, quand chacun est le bon choix
- Panorama : LangGraph, CrewAI, plateformes managées (Bedrock Agent Core, Vertex Agent Engine, Azure AI Foundry)
- Mise en pratique : reconstruction de l’agent précédent avec LangGraph, ajout d’outils de support client, d’une mémoire de conversation et d’une validation humaine avant action sensible.

5. MCP : industrialiser l’accès aux outils

- Le problème que MCP résout : N intégrations × M applications
- Serveurs, outils, ressources, transports
- Écrire un serveur MCP et le brancher sur un agent, un IDE, un assistant
- Sécurité : périmètre, secrets, confiance dans les serveurs tiers
- Mise en pratique : exposition des outils métier du TP précédent sous forme de serveur MCP, puis consommation par l’agent et par un assistant du commerce.

Référence

THII3719

Durée

2 jours / 14 heures

Prix HT / stagiaire

1450€

Objectifs pédagogiques

- Arbitrer entre prompt, RAG, outils/agent et fine-tuning pour un besoin donné, avec les coûts et les limites de chaque option
- Implémenter un agent outillé et un pipeline RAG en Python, en local comme sur API
- Réaliser un fine-tuning LoRA d’un modèle ouvert et l’exploiter en production locale
- Évaluer la qualité d’un système génératif et instrumenter son coût
- Identifier les risques de sécurité et les obligations de conformité d’une mise en production

Niveau requis

- Pratique régulière d’un langage de programmation ; les travaux pratiques se font en Python (niveau lecture/adaptation de scripts suffisant)
- Aisance avec un terminal, un éditeur de code et l’installation de logiciels sur son poste
- Une connaissance d’usage des assistants IA (ChatGPT, Claude, Le Chat…) est un plus, non un prérequis

Public concerné

- Développeurs, lead devs, architectes logiciels et data
- Responsables techniques, chefs de projet et responsables innovation devant arbitrer des projets d’IA générative
- Équipes en charge d’un premier passage en production d’une brique d’IA générative

Formateur

Les formateurs intervenants pour Themanis sont qualifiés par notre Responsable Technique Olivier Astre pour les formations informatiques et bureautiques et par Didier Payen pour les formations management.

Conditions d’accès à la formation

Délai : 3 mois à 1 semaine avant le démarrage de la formation dans la limite des effectifs indiqués

Moyens pédagogiques et techniques

Salles de formation (les personnes en situation de handicap peuvent avoir des besoins spécifiques pour suivre la formation. N’hésitez pas à nous contacter pour en discuter) équipée d’un ordinateur de dernière génération par stagiaire, réseau haut débit et vidéo-projection

Débrief de la journée

- Retour sur les briques construites, questions ouvertes, première version de la grille de décision.

6. RAG : brancher un modèle sur ses documents

- Le pipeline complet : découpage, vectorisation, index, recherche, génération
- Embeddings et similarité : ce qu'un vecteur représente vraiment
- Bases vectorielles : Chroma, Qdrant, pgvector, index intégrés — critères de choix
- Recherche hybride, reclassement, métadonnées et filtrage
- Citer ses sources et refuser de répondre : les deux exigences non négociables
- Les cinq causes d'échec d'un RAG et comment les diagnostiquer
- Mise en pratique : un moteur de RAG écrit de bout en bout sans bibliothèque spécialisée (pour tout voir), puis le même avec une base vectorielle ; dégradation volontaire du découpage et observation des effets.

7. Fine-tuning : spécialiser un modèle

- Fine-tuning complet, LoRA, QLoRA : ce qui change, ce que ça coûte
- Ce que le fine-tuning apporte (format, style, tâche) et ce qu'il n'apporte pas (connaissances fraîches)
- Constituer un jeu de données : volume réel nécessaire, qualité, séparation train/test
- Entraîner, quantifier, exporter, servir le modèle obtenu
- Coût total de possession : réentraînement, dérive, versionnage
- Mise en pratique : fine-tuning LoRA d'un petit modèle sur un jeu de données métier, export et exécution locale du modèle obtenu, comparaison mesurée avant / après.

8. Les autres modalités : image et diffusion

- Diffusion : le principe du débruitage progressif, et pourquoi il a remplacé les GAN
- Du texte à l'image : le rôle des modèles d'alignement texte-image
- La même mécanique LoRA appliquée à l'image : styles et personnalisation
- Ce qui change côté implémentation : coût, latence, droits

9. Évaluer et observer

- Pourquoi les tests classiques ne suffisent pas : non-déterminisme et jugement
- Construire un jeu de test : cas nominaux, cas limites, cas piégés
- Métriques utiles : exactitude factuelle, ancrage dans les sources, taux de refus, coût, latence
- Le modèle juge : intérêt, biais et précautions
- Traces, journalisation et suivi des coûts en production
- Outillage : Ragas, Langfuse, LangSmith — quand un harnais maison suffit
- Mise en pratique : construction d'un jeu de tests et d'un harnais d'évaluation, appliqué au RAG et au modèle fine-tuné produits dans la journée.

10. Servir et déployer

- Dimensionner : taille de modèle, quantization, VRAM, débit, utilisateurs simultanés
- Moteurs de service : Ollama, llama.cpp, vLLM — cas d'usage respectifs
- API managées et plateformes : Bedrock, Vertex AI, Azure AI Foundry, Mistral, OpenAI
- Conteneurisation et mise à l'échelle : ce qui est spécifique aux modèles génératifs
- Souveraineté, hébergement et données : arbitrer local / cloud européen / cloud américain
- Coûts réels : construire un budget à l'appel et au mois

11. Sécuriser et se conformer

- OWASP Top 10 pour applications LLM (édition 2026) : les risques qui comptent
- Injection de prompt directe et indirecte : la démonstration sur un système réel
- Agence excessive : le risque n° 1 des déploiements agentiques
- Fuites de données, secrets, journalisation, chaîne d'approvisionnement des modèles
- AI Act : calendrier applicable, obligations de transparence, classification des systèmes

- Mise en pratique : injection d’une instruction malveillante dans le corpus documentaire construit la veille, observation du détournement de l’agent, mise en place des contre-mesures.

12. Synthèse : choisir et se lancer

- La grille de décision complète : prompt, RAG, agent, fine-tuning, combinaisons
- Du prototype à la production : les étapes et les pièges habituels
- Compétences à réunir et arbitrages internalisation / externalisation
- Atelier final : chaque participant qualifie un cas d’usage réel avec la grille et repart avec une architecture cible argumentée