Assistant IA low-cost pour la Nuit de l'Info 2025

Architecture et justification des choix

Équipe « ... »

5 décembre 2025

Table des matières

1 Contexte et objectifs

La Nuit de l'Info est un marathon de développement web de 16 heures où des équipes d'étudiant·es doivent concevoir et déployer une application complète en un temps très limité.

Notre projet est un **assistant de questions/réponses** dédié à la Nuit de l'Info et aux services publics numériques, avec les contraintes suivantes :

- **Faible connectivité** : certaines salles ont peu ou pas d'Internet.
- **Budget limité** : pas de dépenses importantes en crédits d'API LLM.
- **Bilingue** français / arabe.
- **Explicable aux jurys** (architecture claire, choix motivés).

Pour répondre à ces contraintes, nous avons conçu une architecture **multi-modes** : hors-ligne, hybride et en-ligne, qui s'adapte automatiquement à la qualité de la connexion et minimise les appels aux LLM distants.

2 Vue d'ensemble de l'architecture

L'architecture est composée de trois briques principales :

1. **Scraping et préparation des données** : un script Python collecte les textes, liens, PDF, vidéos et boutons du site officiel `https://www.nuitdelinfo.com`. Un deuxième script génère une base de FAQ bilingue et des embeddings légers.
2. **Frontend web (React)** : une application de chat qui tourne dans le navigateur, avec stockage local et trois moteurs IA : OFFLINE, HYBRID et ONLINE.
3. **Backend IA (FastAPI + RAG + OpenRouter)** : une API REST qui fait de la recherche sémantique (RAG) sur les FAQs puis appelle un modèle LLM distant (Llama 3.3 70B via OpenRouter) uniquement en mode en-ligne.

Flux simplifié

- L'utilisateur pose une question dans le chat.
- Le frontend choisit un **mode IA** en fonction de la connectivité (hors-ligne, hybride, en-ligne).
- Le moteur sélectionné interroge les FAQs locales et/ou l'API backend, puis renvoie une réponse courte et des sources.

3 Les trois modes IA

3.1 Mode OFFLINE (hors-ligne)

- Données : la base de FAQ est stockée dans `IndexedDB` et dans le bundle statique du frontend.
- Algorithme : simple recherche par mots-clés sur les questions et les mots-clés associés.
- Avantages :
  - Temps de réponse < 200 ms.
  - Fonctionne **sans Internet**.
  - Aucun coût d'API.
- Limites :
  - Sensibilité à l'orthographe et à la formulation exacte.
  - Ne gère pas les reformulations complexes.

3.2 Mode HYBRID (RAG local sans LLM)

- Données : même base de FAQ, complétée par un fichier `embeddings.json` contenant des vecteurs de dimension 384.
- Embeddings : vecteurs légers calculés par un *hash* stable des mots (pas de gros modèle de type BERT côté client).
- Algorithme :
  1. Calculer un embedding pour la requête utilisateur (même hash).
  2. Calculer la similarité cosinus avec toutes les FAQs.
  3. Retourner les k FAQs les plus proches comme contexte.
- Réponse : en mode hybride pur, la réponse reste celle de la FAQ la plus pertinente (forme simple mais déjà robuste).
- Avantages :
  - Comprend mieux les reformulations (« Quand est la nuit ? » vs « Quelle est la date de la Nuit de l'Info ? »).
  - Toujours **entièrement local** : aucun appel à un LLM distant.

3.3 Mode ONLINE (backend + RAG + LLM)

- Le frontend appelle `/api/chat` sur le backend FastAPI.
- Le backend effectue :
  1. RAG : recherche sémantique des FAQs les plus pertinentes.
  2. Construction d'un **prompt court** avec seulement quelques FAQs (top-k).
  3. Appel à un LLM hébergé via OpenRouter : `meta-llama/llama-3.3-70b-instruct:free`.
  4. Reformulation de la réponse dans la langue choisie (français ou arabe).
- Avantages :
  — Réponses plus naturelles et pédagogiques.
  — Capacité à combiner plusieurs FAQs dans une seule réponse.
  — Toujours **ancré sur les données officielles** du site (grâce au RAG).
- En cas d'erreur réseau, le frontend repasse automatiquement en mode HYBRID.

4 Justification des choix IA low-cost

4.1 Réduction drastique des appels LLM

- Les modes OFFLINE et HYBRID couvrent déjà une grande partie des questions, sans aucun appel à un LLM distant.
- Le mode ONLINE n'est utilisé que lorsque :
  — la connexion est bonne ;
  — l'utilisateur le demande explicitement (bouton ON) ;
  — ou le système détecte une question plus complexe où la reformulation par LLM apporte une vraie valeur.

Ainsi, le nombre d'appels à OpenRouter reste faible, ce qui limite :

- les coûts potentiels de tokens ;
- la dépendance à la connectivité.

4.2 Embeddings légers et reproductibles

- Plutôt que d'utiliser un modèle lourd type Sentence-BERT côté client, nous utilisons un schéma d'embedding léger basé sur un *hash* stable.
- Avantages :
  — Pas de poids de modèle à télécharger (important pour les réseaux lents).
  — Calculs simples (somme de compteurs, normalisation L2) ; facilement exécutables dans un navigateur ou en Python.

4.3 RAG avant LLM

- Nous effectuons toujours la sélection des documents (FAQs) **avant** d'appeler le LLM.
- Le prompt envoyé au LLM contient seulement :
  — la question utilisateur ;
  — un petit nombre de FAQs pertinentes (par exemple 3).
- Cela réduit la **taille du contexte**, donc :
  — moins de tokens facturés ;
  — temps de réponse plus court.

4.4 Dégradation gracieuse selon le réseau

- Si l'API backend est indisponible ou lente, le frontend bascule automatiquement sur HYBRID puis OFFLINE.
- L'utilisateur garde toujours une réponse fiable, même sans connexion, grâce à la base de FAQs embarquée.

5 Notice d'installation et de lancement

Cette section peut servir de base au document d'installation officiel.

5.1 Préparation des données

1. Lancer le script de scraping :

```
cd Script
python3 scraper_nuit_info.py
```

Le crawler visite un sous-ensemble de pages HTML de `www.nuitdelinfo.com` (limité à `max_pages`) et enregistre les données dans `Script/data/nuit`$_{i}nfo/$.

1. `Générer les FAQs et embeddings :`

```
cd ..
python3 process_all_data.py
```

`Ce script crée ou met à jour frontend/public/data/faqs.json et frontend/public/data/embe`

5.2 Lancement du backend

1. Depuis le dossier `backend`, créer l'environnement virtuel et installer les dépendances :

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

2. Créer un fichier `.env` avec la clé OpenRouter :

```
OPENROUTER_API_KEY=sk-or-...
OPENROUTER_MODEL=meta-llama/llama-3.3-70b-instruct:free
```

3. Démarrer le serveur :

```
uvicorn app.main:app --reload --port 5000
```

5.3 Lancement du frontend

1. Installer les dépendances :

```
cd frontend
npm install
```

2. (Optionnel) créer `frontend/.env` :

```
REACT_APP_API_URL=http://localhost:5000/api
```

3. Démarrer l'application :

```
npm start
```

4. Ouvrir `http://localhost:3000`, choisir la langue et le mode IA (OFFLINE / HYBRID / ONLINE).

6 Conclusion

Cette architecture démontre qu'il est possible de construire un assistant IA utile, bilingue et pédagogique tout en respectant des contraintes fortes de **faible connectivité** et de **budget limité**.

La combinaison des trois modes IA, du scraping ciblé du site officiel et de l'utilisation raisonnée d'un LLM distant via OpenRouter permet d'obtenir un compromis efficace entre :

- qualité des réponses ;
- coûts en tokens et dépendance réseau ;
- simplicité de déploiement pour une équipe étudiante.