

 Questions de réflexion (à rendre avec le TP)

1. Sécurité et RGPD

Q1.1 : Pourquoi est-il important de stocker les clés API dans un fichier .env et pas directement dans le code ?

Pour éviter de divulguer la clé si le code est partagé (GitHub, projet scolaire, etc.).

Pour sécuriser les informations sensibles : une clé API donne accès au service, donc un attaquant pourrait l'utiliser à ta place.

Pour faciliter la configuration par environnement (dev / prod) sans modifier le code.
→ C'est une bonne pratique DevOps et de sécurité.

Q1.2 : Quelles données personnelles sont collectées par notre agent ? Est-ce conforme au principe de minimisation du RGPD ?

On collecte *seulement* :

- la ville/destination donnée par l'utilisateur,
- éventuellement sa question ou commande.

Oui, c'est conforme au principe de minimisation du RGPD car l'agent ne collecte que les données strictement nécessaires pour fonctionner.

Il ne stocke pas de données supplémentaires (nom, email, IP, âge...) → donc minimisation respectée.

Q1.3 : Que se passerait-il si on stockait l'historique des conversations ? Quelles obligations RGPD s'appliqueraient ?

Si on stocke l'historique, alors on devient responsable de traitement.

Obligations :

- Informer l'utilisateur de ce qui est collecté et pourquoi (transparence).
- Possibilité pour l'utilisateur d'exercer ses droits (accès, suppression, rectification).

- Limiter la durée de conservation.
- Avoir une base légale (le consentement est souvent nécessaire).
- Sécuriser les données (chiffrement, accès limité).
En résumé : stockage = RGPD obligatoire.

2. Conception conforme CNIL

Q2.1 : Citez 3 recommandations de la CNIL que nous avons appliquées dans ce TP.

Transparence : Bannière indiquant qu'il s'agit d'un robot

Information : Popup explicative sur le traitement des données

Pas de stockage : Aucune conversation n'est enregistrée

Q2.2 : Comment l'utilisateur est-il informé qu'il parle à un robot ? Pourquoi est-ce important ?

- L'interface affiche un message du type "Je suis un agent conversationnel IA".
- Ou iconographie / nom explicite (ex : "Assistant IA").

Importance :

- Transparence : c'est une obligation CNIL.
- Évite la confusion avec un humain.
- Permet à l'utilisateur d'adapter sa manière de formuler ses demandes.
- Renforce la confiance.

Q2.3 : Proposez une amélioration pour ajouter une "supervision humaine" comme recommandé par la CNIL.

Ajouter un bouton "Contacter un humain" pour escalader la conversation.

Mettre un système où les réponses douteuses ou non comprises sont revues par un modérateur humain.

Ajouter un journal d'erreurs consulté par un administrateur humain.

Permettre à un humain de valider certaines décisions sensibles (réservations, achats, etc.).

3. Technique

Q3.1 : Expliquez le rôle de Mistral AI dans notre application

- Mistral AI fournit le modèle de langage utilisé par l'agent.
- C'est lui qui analyse la question, génère la réponse et comprend le contexte.
- L'app envoie un prompt → Mistral IA renvoie du texte.

En résumé : c'est le cerveau de l'agent conversationnel.

Q3.2 : Pourquoi utilise-t-on `response_format={"type": "json_object"}` dans la fonction `extraire_ville()` ?

Pour forcer le modèle à répondre en JSON structuré.

Cela permet d'avoir une sortie propre du style :

```
{"ville": "Paris"}
```

Cela évite que le modèle renvoie du texte non exploitable.

→ C'est crucial pour qu'un code Python puisse parse les données sans erreur.

Q3.3 : Comment pourrait-on gérer plusieurs langues dans l'agent conversationnel ?

DéTECTer automatiquement la langue avec un modèle (langdetect ou directement via Mistral).

Adapter les prompts selon la langue détectée.

Avoir des fichiers de traduction (i18n) pour l'interface.

Utiliser un modèle multilingue, capable de répondre dans la langue de l'utilisateur.

Demander à Mistral : "*Réponds dans la même langue que l'utilisateur.*"