Mon ChatGPT

Objectifs du Projet

Ce projet vise à valider les compétences suivantes :
• Tester de façon unitaire son code
• Tester une interface grâce aux tests e2e
• Maîtriser une méthodologie de test : Test Driven Development (TDD)
• Linter et méthode de refactoring

Description du Projet

L'objectif est de concevoir votre propre ChatGPT sous la forme d'une architecture DDD et Hexagonale. Vous devez donc réaliser une solution permettant d'ouvrir une conversation et d'y dialoguer avec un modèle type ChatGPT. Bien sûr, il doit être simple de différencier les demandes de l'utilisateur et les réponses du modèle.

Pour réaliser votre solution vous devez avoir 3 parties :

1. Frontend - Une interface graphique permettant d'écrire les messages à votre API avec une gestion par conversation.
2. Backend - Une API qui utilise un modèle d'IA et stocke en base de donnée les conversations pour en garder un historique.
3. Base de donnée - Permet de stocker l'historique des conversations.

Votre solution doit permettre côté Backend :

- Une authentification par utilisateur (N'hésitez pas à aller au plus simple en utilisant des solutions déjà existantes).
- La recherche de conversation par mot clé (Ex: 'NestJS' doit me montrer toutes les conversations où l'on parle de NestJS).
- La recherche de messages dans une conversation.
- Le partage d'une conversation via un lien unique en publique.

Votre solution doit permettre côté Frontend :

- De créer des conversations, de voir la liste de celles déjà existantes avec leur nom et la date de création.
- D'afficher correctement le code généré par l'IA (Facile à lire et à copier).
- De modifier un message et de le renvoyer à l'IA.

Le choix des technologies pour les différentes couches et la base de données est libre.
Pour le choix du modèle, je vous conseille d'utiliser Gemini de Google qui est l'un des seuls modèles utilisable sans paiement : Gemini API.
Vous pouvez cependant utiliser le modèle de votre choix si vous avez déjà une clé API ou en utilisant Hugging Face qui fournit certains modèles gratuitement mais qui ne sont pas très performant : https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407. Pour utiliser ce modèle, vous devez créer un compte Hugging Face et créer un Token que vous utiliserez comme une clé API.

Vous devez avoir au minimum l'architecture de fichier suivante côté Backend :

```
src/
    domains/
        <votre_domaine>/
            entities/
            services/
            repositories/
    application/
        usecases/
        ...
    infrastructure/
        adapters/
        ...
```

Pour `domains` :

- Contient toute la logique métier et un dossier par domaine.
- Chaque domaine contient ses entités, ses services et ses repositories sous forme de classe ou d'interface.

Pour `application` :

- Contient toute la logique applicative (routes, controllers, etc).
- Les `usecases` utilisent les entities, services et repositories pour réaliser une action en recevant en dépendances les adapters (ex: SendMessageUseCase).

Pour `infrastructure` :

- Contient toutes les abstractions de technologies externes
- Hérite des services et repositories pour implémenter la logique avec une technologie externe (Ex: MongoDBMessageRepository qui hérite de l'interface MessageRepository

dans domains/).

Vous pouvez bien sûr ajouter des dossiers / fichiers en fonction de vos besoins.
La logique métier doit être testée unitairement.
L'interface graphique doit être testée en E2E avec un outil comme Cypress ou Playwright.
L'ensemble du développement doit être réalisé en TDD.
Le projet doit avoir un linter permettant de vérifier la qualité du code.
Vous devez réaliser une pipeline CI / CD permettant d'automatiser la vérification de la qualité sur votre projet.

Chaque commit doit suivre la convention suivante : Conventional Commits pour faciliter la lecture, la compréhension et la gestion des versions.

Évaluation

Ce projet est individuel. Un lien vers votre dépôt de code est requis pour vérification avant la soutenance.

La soutenance comprendra :

1. Présentation du projet par l'étudiant (environ 7 minutes).
2. Questions sur les choix effectués et revue technique (environ 3 minutes).

L'évaluation se basera sur 20 points, portant sur les compétences évaluées.
Le versionnage progressif du code est obligatoire tout au long du projet.
Tout bonus innovant sera valorisé.