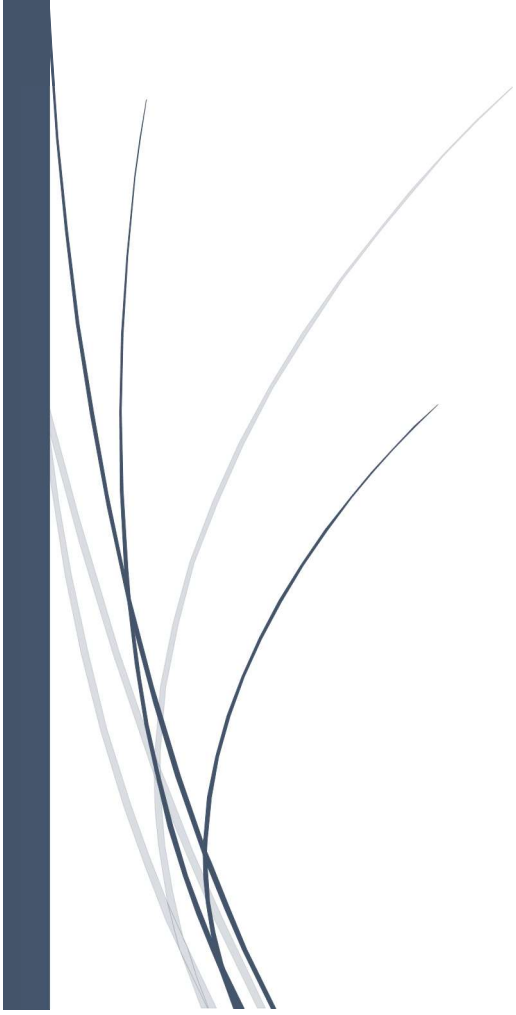
22/03/2024

Turbo-couscous

Assistant IA (ChatBot) en Java



Par ADDI Mohammed, BAKER Mohammad
et ODIN Thomas

Table des matières

Introduction :

Dans le paysage dynamique de la technologie numérique, les chatbots se sont imposés comme des outils essentiels pour améliorer l'interaction entre les humains et les systèmes informatiques. Ce projet vise à concevoir et développer un chatbot en Java, capable de simuler des conversations humaines pour fournir des informations, répondre à des questions et assister les utilisateurs dans diverses tâches. L'objectif est de créer un système modulaire, intégrant des fonctionnalités de traitement du langage naturel avancées et offrant une interface utilisateur conviviale. En tirant parti des outils de développement Java modernes et des bibliothèques NLP, ce chatbot se propose d'être une solution à la fois robuste et agile pour divers scénarios d'utilisation, allant de l'assistance technique au service client.

Analyse des Besoins :

La phase d'analyse des besoins est cruciale pour le succès de tout projet de développement logiciel. Elle implique l'identification précise des fonctionnalités que le système doit offrir ainsi que les contraintes opérationnelles auxquelles il doit s'adapter. Pour notre chatbot Java, les besoins ont été classés en fonction de leur pertinence et de la faisabilité de leur mise en œuvre. Les cas d'utilisation identifiés englobent des réponses aux requêtes standards, l'assistance en réservation et la fourniture de solutions techniques simples.

Les besoins réalisés incluent la capacité du chatbot à :

- Informer les utilisateurs sur ses fonctions propres.
- Fournir des recettes de cuisine, un exemple concret étant des recettes de couscous.
- Résoudre des problèmes techniques courants, comme le changement d'une ampoule ou l'installation d'une application.
- Gérer l'agenda de l'utilisateur.
- Présenter la météo.
- Envoyer des emails en nom de l'utilisateur.
- Fonctionner en anglais et en français, ce qui permet une accessibilité élargie.

En raison de contraintes de temps et de ressources, certains besoins n'ont pas été réalisés. Ceux-ci comprennent :

- La commande vocale : Un aspect complexe qui requiert une intégration plus profonde des capacités de reconnaissance vocale et de traitement du langage naturel.
- La domotique : La gestion de dispositifs domestiques intelligents qui nécessite une interface avec des protocoles et des standards variés.

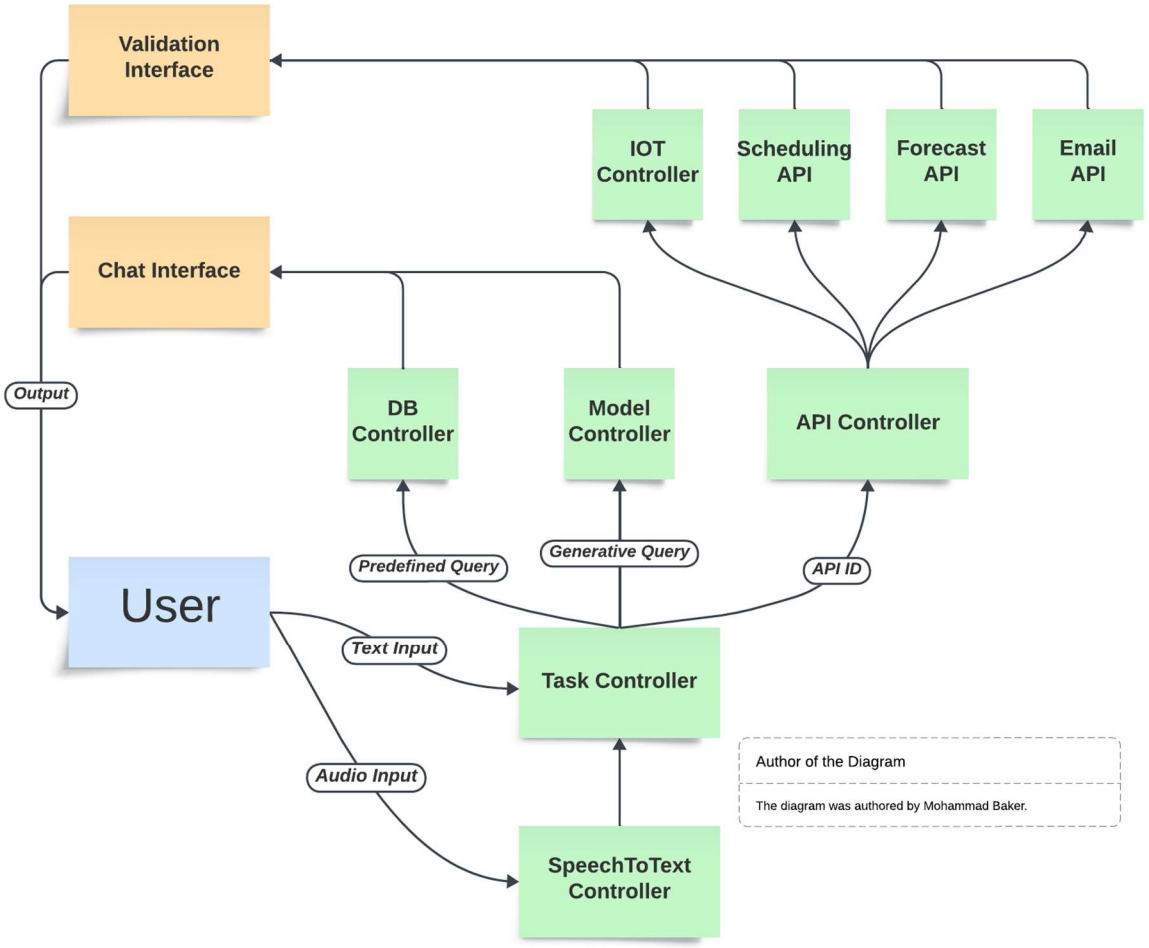
Conception :

Avec l'analyse des besoins en tête, nous avons créé une architecture de système qui catégorise les composants en interfaces utilisateur, contrôleurs de tâches et services. Chaque élément a été conçu pour s'interconnecter de manière fluide, garantissant une expérience utilisateur transparente.

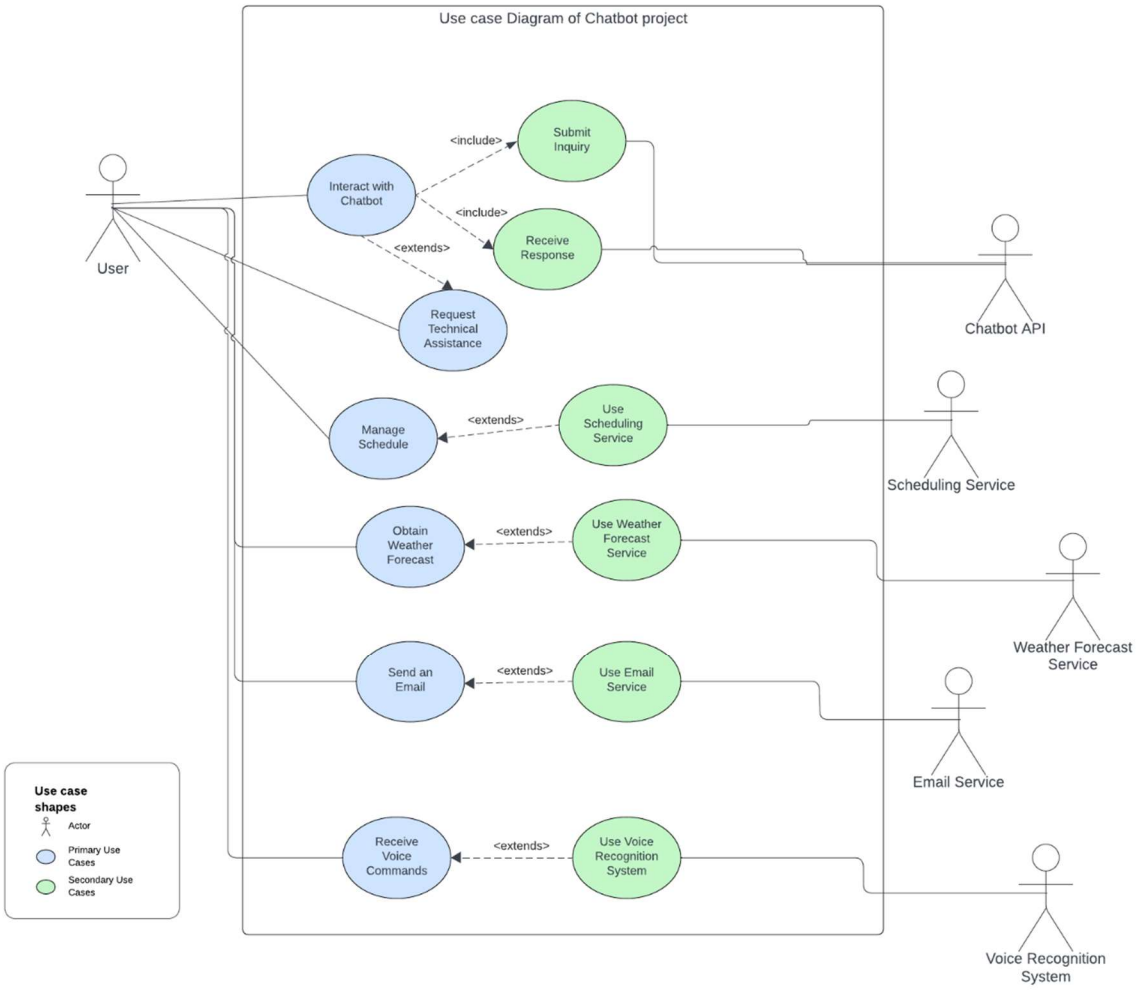
- Interfaces Utilisateur : Le Chat Interface et le Validation Interface sont les points de contact primaires pour l'utilisateur, offrant un retour visuel et une interaction directe.
- Contrôleurs de Tâches : Le Task Controller, le DB Controller, le Model Controller et le SpeechToText Controller orchestrent le traitement des entrées utilisateur et la logique métier.
- Services et API : Les différents services API, y compris l'IOT Controller, le Scheduling API, le Forecast API et l'Email API, étendent les capacités du chatbot au-delà de la simple interaction textuelle, en permettant une intégration avec d'autres systèmes et services en ligne.

Les Diagrammes UML — Use Case Diagram, Sequence Diagram — démontrent de manière visuelle comment les utilisateurs interagiront avec le chatbot et comment le chatbot traitera les requêtes, depuis l'entrée de l'utilisateur jusqu'à la réponse finale, en passant par les différents services et contrôleurs impliqués.

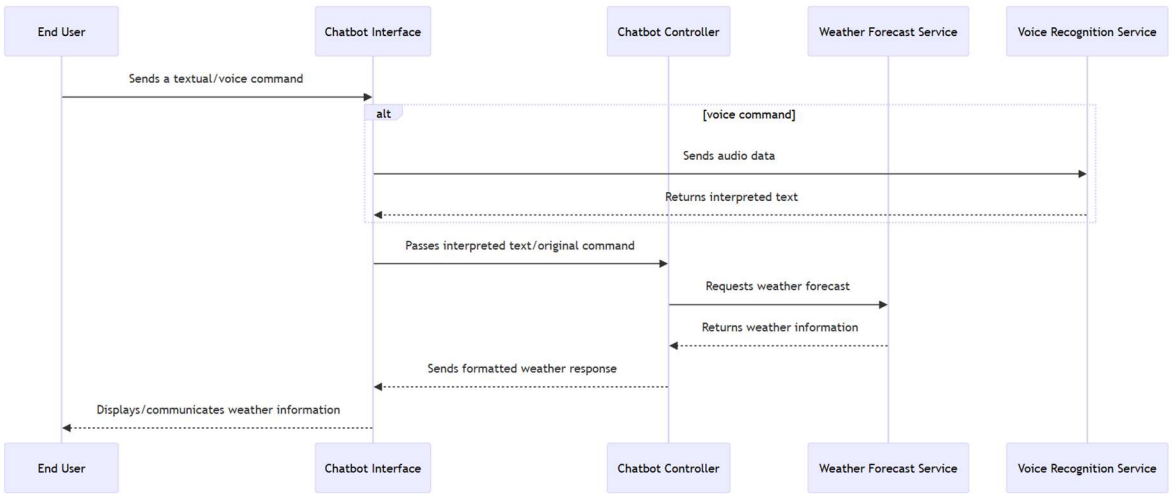
MVC Diagram :



Use Case Diagram :



Sequence Diagram :



Développement :

1. Gestion de la Base de Données (DB)

Le TaskController est le composant central qui orchestre le flux des tâches et la logique métier du chatbot. Voici comment le développement a été abordé pour certains de ses aspects clés :

- Gestion des Emails : Lorsqu'il est en mode collecte d'informations pour envoyer un email, le TaskController recueille séquentiellement l'adresse destinataire, le sujet et le corps de l'email. Une fois ces informations recueillies, il utilise la classe GmailSend pour envoyer l'email via l'API Gmail. La gestion des exceptions est en place pour traiter les éventuelles erreurs d'envoi.
- Gestion des Rendez-Vous : Une logique similaire est utilisée pour la programmation des rendez-vous avec l'API Google Agenda. Le TaskController recueille les informations nécessaires comme le résumé, la description, et les horaires de début et de fin avant d'ajouter l'événement au calendrier.

L'ingénierie du traitement du langage naturel (NLP) est prise en charge par NlpEngine, qui utilise des modèles OpenNLP pour la détection de phrases, la tokenisation et l'attribution des parties du discours (POS tagging).

- Détection des Intentions : La logique d'interprétation des intentions est codée de manière à reconnaître les intentions telles que la programmation de rendez-vous, l'envoi d'emails, la prévision météorologique, etc., en fonction des tokens reçus.
- Interrogation du Dataset : Si une intention n'est pas reconnue immédiatement, NlpEngine consulte un dataset JSON pour trouver une correspondance potentielle, en utilisant un traitement des expressions régulières et une correspondance de motif. S'il ne trouve pas de réponse dans le dataset, il délègue la demande à OpenAIChatbot, qui peut générer une réponse plus contextuelle.

2. API Google et météo

API Google : utilisez l'API OAuth de Google pour récupérer et stocker le jeton d'accès, puis envoyer une demande d'envoi d'un e-mail ou récupérer l'agenga à l'aide de l'API Google avec le jeton d'accès, puis renvoyer les données.

API Météo : utilisez l'adresse IP pour obtenir l'emplacement à l'aide d'une API, puis obtenez la météo à partir d'une API utilisant l'emplacement.

3. API OpenAI

Configuration de l'API

- HttpClientHelper : Une classe d'assistance pour gérer les requêtes HTTP vers l'API OpenAI. Elle encapsule la complexité de l'envoi des requêtes et de la réception des réponses HTTP.

- API_KEY : La clé API est configurée en tant que variable d'environnement, ce qui renforce la sécurité et facilite la configuration sans modification du code.

Envoi des Requêtes

- Les requêtes POST sont envoyées à l'URL de l'API avec les données requises encapsulées au format JSON, y compris les messages de l'utilisateur.

Traitement des Réponses

- Les réponses JSON de l'API sont désérialisées dans des objets Java pour faciliter leur traitement. L'objet OpenAIResponse contient la structure attendue de la réponse, y compris la liste des choix.
- OpenAIChatbot : Utilise HttpClientHelper pour envoyer des messages à l'API OpenAI et reçoit des réponses que nous transformons en réponses textuelles pour l'utilisateur.
- La logique de gestion d'erreurs robuste assure que les exceptions sont capturées et que des messages d'erreur utiles sont renvoyés à l'utilisateur.

Gestion des Erreurs

Des exceptions inattendues sont capturées et un message d'erreur approprié est retourné, assurant ainsi que l'utilisateur est informé en cas de défaillance de la communication avec l'API OpenAI.

4. Fonctionnement global :

- MainApp : C'est la classe principale de l'application. Elle contient la méthode main, qui lance l'application JavaFX en appelant Application.launch avec la classe ChatbotGUI comme argument.
- ChatbotGUI : Cette classe étend Application et surcharge la méthode start, qui est le point d'entrée de l'interface utilisateur JavaFX. La méthode start initialise et affiche l'interface graphique du chatbot.

Composants de l'Interface Utilisateur

- VBox chatPane : Contient les messages échangés entre l'utilisateur et le chatbot.
- ScrollPane scrollPane : Permet de faire défiler le contenu de chatPane lorsque les messages dépassent la hauteur disponible.
- TextField userInput : Permet à l'utilisateur de saisir ses messages.
- Button sendButton : L'utilisateur peut cliquer sur ce bouton pour envoyer son message ou appuyer sur Enter après avoir saisi son message.

Traitement des Messages

Lorsqu'un message est envoyé par l'utilisateur, la méthode sendMessage est appelée. Cette méthode ajoute d'abord le message de l'utilisateur à chatPane, puis demande à TaskController de traiter le message.

- TaskController : C'est le cœur logique de l'application. Il traite l'entrée de l'utilisateur, interagit avec divers services et API, et génère une réponse appropriée.

- Il utilise ModelController pour traiter l'entrée de l'utilisateur en utilisant des techniques de traitement du langage naturel (NLP).
- Il peut gérer des tâches complexes comme envoyer un e-mail ou planifier des rendez-vous en demandant des informations supplémentaires à l'utilisateur, en utilisant des services tels que GmailSend et agendaGoogle.
- Il gère également les interactions avec l'API météo et d'autres services personnalisés selon les besoins.

API Controller

- apiController : C'est une classe qui peut être utilisée pour initialiser et gérer des connexions avec différentes API externes que le chatbot pourrait utiliser.

Cycle de Vie d'un Message

- L'utilisateur saisit un message et appuie sur Enter ou clique sur le bouton Send.
- sendMessage est appelée, qui ajoute le message à l'interface graphique et passe le message à TaskController.
- TaskController détermine l'intention de l'utilisateur et effectue l'action requise.
- Une réponse est générée et renvoyée à ChatbotGUI.
- ChatbotGUI affiche la réponse dans chatPane.

Intégration des Services

- Le chatbot est conçu pour être extensible, avec la capacité d'intégrer de multiples services via des API.
- Que ce soit pour récupérer des prévisions météorologiques, envoyer des e-mails, ou gérer des rendez-vous, chaque fonctionnalité est gérée de manière modulaire.

Tests :

Tests d'Intégration avec l'API OpenAI

Procédure de Test :

- Vérification Manuelle dans le Terminal : Des tests manuels ont été réalisés en mode de développement, avec des logs de console configurés pour afficher les détails des requêtes et des réponses. Cette étape a permis une vérification visuelle et instantanée de l'intégration de l'API.
- Utilisation d'Outils Tiers : Des outils tels que Postman ont également été employés pour effectuer et analyser des appels d'API indépendamment du code de l'application, fournissant une couche supplémentaire de validation de nos requêtes et de la gestion des réponses.

Tests des Interactions avec la Base de Données

Pour assurer l'intégrité et la performance de notre base de données en interaction avec l'application, des tests spécifiques ont été conçus. Ces tests comparent les opérations effectuées sur la DB aux données de référence contenues dans un fichier JSON, en plus de vérifier les fonctions de création, mise à jour, et suppression de données.

Procédure de Test :

- Comparaison avec un Fichier JSON : Un script de test a été développé pour extraire les données de la base de données et les comparer aux informations du fichier JSON de référence. Ce processus garantit que les données stockées dans la DB correspondent exactement à nos données de référence.
- Vérifications Manuelles : Des requêtes directes ont été effectuées à l'aide d'outils de gestion de base de données pour inspecter manuellement l'état des données après diverses opérations. Cette approche offre une vérification ponctuelle et détaillée de la cohérence des données.

Documentation :

Prérequis

Avant de commencer, assurez-vous que les prérequis suivants sont remplis :

- Environnement Java : Une version récente de Java Development Kit (JDK) doit être installée sur votre système.
- JavaFX : JavaFX est requis pour l'interface utilisateur. Il doit être installé et correctement configuré avec votre JDK.

Installation et Configuration

1. Cloner le Répertoire : Commencez par cloner le répertoire Git contenant le code du projet sur votre machine locale.
2. « git clone Todin13/turbo-couscous: A chat bot project made in java. (github.com)»
3. Lancer l'Application : Ouvrez votre environnement de développement intégré (IDE) préféré, importez le projet et lancez MainApp.java.

Utilisation du Chatbot

- Interactions générales : Vous pouvez interagir avec le chatbot en saisissant des questions ou des commandes dans l'interface utilisateur. Le chatbot répondra si la requête est comprise et prise en charge.
- Envoi d'email : Pour envoyer un email, tapez des phrases telles que "email", "send email", ou "I want to send an email". Le chatbot vous guidera ensuite à travers les étapes nécessaires pour recueillir l'adresse du destinataire, le sujet et le corps de l'email.
- Affichage du calendrier : Saisissez "calendar summary" pour que le chatbot affiche le résumé des événements à venir dans votre calendrier.
- Ajout d'un événement au calendrier : Tapez "add event" pour commencer le processus d'ajout d'un nouvel événement dans le calendrier. Le chatbot demandera les détails de l'événement, y compris le résumé, la description, et les horaires de début et de fin.

Organisation de Travail :

Conclusion :

En conclusion, le développement de ce chatbot en Java a été une entreprise rigoureuse et instructive qui a touché à plusieurs aspects importants du génie logiciel, de l'intelligence artificielle et de l'interaction homme-machine. À travers la conception modulaire, l'application de principes de traitement du langage naturel, et l'intégration avec des API extérieures, nous avons créé un outil interactif capable d'assister les utilisateurs dans leurs tâches quotidiennes, de répondre à leurs interrogations et de faciliter une communication fluide et naturelle.

L'interface utilisateur, développée avec JavaFX, offre une expérience conviviale qui rend l'interaction avec le chatbot à la fois intuitive et agréable. Le système de gestion des tâches, conçu pour être robuste et extensible, permet une grande variété de fonctionnalités, depuis l'envoi d'e-mails jusqu'à la gestion d'agenda, prouvant ainsi l'efficacité du chatbot en tant qu'assistant personnel virtuel.

Tout au long de ce projet, nous avons été confrontés à des défis significatifs, notamment en matière d'interprétation des intentions de l'utilisateur et de gestion des états du chatbot. Ces obstacles ont été surmontés par des itérations de conception et de développement agiles, et ont été une source d'apprentissage précieuse.

Alors que le domaine des chatbots et des assistants virtuels continue d'évoluer rapidement, notre projet sert de fondation solide pour l'exploration future et l'expansion dans ce domaine dynamique. Nous sommes convaincus que le travail effectué ici apportera de la valeur aux utilisateurs finaux et inspirera d'autres développements dans l'avenir de l'intelligence artificielle conversationnelle.