Développement d'un chatbot pour un Escape Room numérique sur la santé mentale

Stage de fin d'études - Rapport intermédiaire

Jalal IZEKKI

jalal.izekki@enseirb-matmeca.fr

Tuteur de stage : M. Laurent ANDRIAMIFIDY
Enseignant référent : M. Julian ALLALI

16 juillet 2022

Enseirb-Matmeca
Département Informatique
Option intelligence artificielle

Table des matières

1 Introduction

1.1 Présentation de l'entreprise

Tricky est une entreprise universitaire qui développe la prévention expérientielle dans le domaine de santé. Créée en 2017, **Tricky** propose des *escape games* physiques et digitaux afin de favoriser les comportements vertueux en santé, diminuer les arrêts maladie et réduire les accidents de travail. Un *escape game* est composé d'un *Escape Room* dont les participants vivent une immersion individuellement ou en groupe afin de résoudre un enigme qui met à l'épreuve leurs émotions et raisonnements. Celui-ci est suivi d'un *débriefing* pendant lequel les participants discutent avec un médiateur en santé, l'objectif étant de libérer la parole des participants pour lever les doutes sur les enjeux de la santé au travail. Les participants sont suivis dans le temps afin d'évaluer leur changement de comportement.

L'équipe de **Tricky** est composée de 16 personnes (dont 4 stagiaires), réparties sur 4 services principaux : produit, ventes, exploitation et ressources humaines (figure 1).

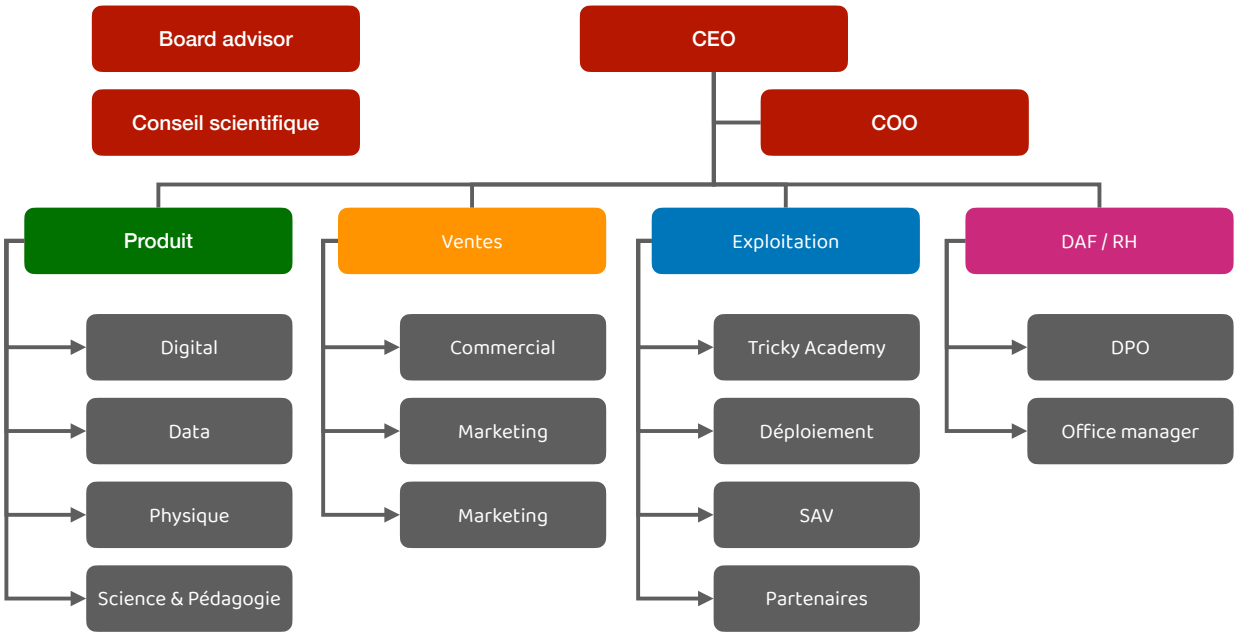

FIGURE 1 – Organigramme de l'entreprise **Tricky**

1.2 Service occupé durant le stage

Le service Produit de **Tricky** est composé de plusieurs équipes, à savoir :

- une équipe digitale qui travaille sur le développement de la partie informatique des jeux virtuels et des sites d'administration.
- une équipe data dont la mission est de collecter, traiter et analyser les données.
- une équipe physique qui développe la partie mécatronique des *escape rooms* physiques.

- une équipe science & pédagogie dont le rôle est l'analyse des sujets de prévention, la scénarisation et la *gamification* des scénarios ainsi que la rédaction des recommandations personnalisées selon les résultats des participants.

En tant que stagiaire en *data science*, j'interviens dans les deux équipes ; digitale et data afin de concevoir, développer et déployer un chatbot qui servira pour les sessions virtuelles d'un jeu collaboratif afin d'accompagner les utilisateurs dans les différentes étapes, répondre à leurs questions techniques, les aider à s'orienter dans les étapes du jeu en leur donnant des indices en cas de besoin et leur présenter des informations qui leur permettront de s'apercevoir des enjeux de santé.

2 Environnement de travail

2.1 Méthode de gestion du projet

La gestion du projet est effectuée à l'aide de *Github Projects* (figure 8 de l'annexe), qui est un système de suivi de projet sur **Github**. Il permet de séparer les tâches et de les attribuer à des *sprints* et suivre la progression du développement. *Github Projects* permet de construire un backlog basé sur des cartes ordonnées dans différentes étapes d'un flux de travail (exemple : à faire, en cours, terminé). Il dispose de déclencheurs qui déplacent automatiquement les tâches d'une étape du flux à la suivante en fonction de l'avancement de l'équipe. Cet outil a été utilisé en complément des outils suivants :

- *Github Issues* (figure 6 de l'annexe) : qui est un système permettant de décrire les tâches et de suivre leur progression et de communiquer autour des sujets qu'elles traitent en utilisant des commentaires et des étiquettes.
- *Github Pull requests* (figure 7 de l'annexe) qui permettent d'indiquer au reste de l'équipe les modifications apportées à une branche dans un dépôt sur *Github* et de discuter et examiner les modifications potentielles à apporter au code. Une *Pull request* peut être considérée comme une tentative d'apporter une solution à une problématique décrite dans la liste des tâches de *Github Issues*.
- *Github Actions* : une plateforme d'intégration continue et de livraison continue (CI/CD) qui permet d'automatiser le pipeline de construction, de test et de déploiement. Elle a été utilisée dans le cadre des différents projets afin d'analyser la cohérence et la qualité du code, lancer les tests dans un environnement indépendant (des machines virtuelles fournies par **Github**) et faciliter la tâche de déploiement.

Afin de fluidifier la communication entre les différentes équipes, l'outil **Slack** [1] est utilisé. Il s'agit d'une plateforme de communication collaborative qui permet d'organiser les communications dans des canaux correspondant à autant de sujets de discussions et de conserver une trace de tous les échanges.

1. Pour plus d'informations : https://fr.wikipedia.org/wiki/Slack_(plateforme)

2.2 Technologies utilisées

La réalisation de ce projet a fait appel à plusieurs technologies, que ce soit pour la partie de l'apprentissage automatique ou au niveau de l'infrastructure :

- **TensorFlow** : un outil open source utilisé pour l'apprentissage automatique, il a été utilisé dans le cadre du projet pour construire les réseaux de neurones profonds utilisés pour prédire les réponses du chatbot.
- **Django** : un *framework* open source de haut niveau utilisé pour le développement web en Python. Il a été conçu pour rendre la création des applications web plus rapide et avec moins de code. Un projet est composé de plusieurs applications qui implémentent chacune un groupe de fonctionnalités autonomes. Il est utilisé par certains sites web les plus fréquentés comme **YouTube**, **Spotify** et **Instagram**.
- **Docker** : un outil qui permet d'empaqueter une application et ses dépendances dans un conteneur virtuel isolé. Celui-ci peut s'exécuter sur n'importe quel ordinateur. Le but à terme est de déployer une image **Docker** avec l'application du chatbot sur un serveur de développement sous **Kubernetes** [2] et ensuite sur un serveur de production. La configuration de l'infrastructure **Kubernetes** sera commune à tous les projets en cours de développement et n'est pas inclue dans les missions du stage.
- **Poetry** : un outil de gestion de dépendances et leur version. Il permet de déclarer les dépendances d'un projet et de les installer/mettre à jour dans un environnement virtuel isolé. **Poetry** se caractérise par ses commandes intuitives et faciles à utiliser ainsi que par son comportement configurable.
- **Huey** : une librairie écrite en python qui permet de créer et gérer des tâches dans des files d'attente. **Huey** permet d'effectuer un traitement en temps réel des tâche tout en prenant en charge la planification des tâches et la création de tâches récurrentes.
- **Redis** : un outil *Open source* de mise en cache et stockage de données dans la mémoire avec une prise en charge de plusieurs structures de données. Cet outil a été utilisé avec un module supplémentaire nommé **RedisAI** pour permettre de servir des modèles **TensorFlow** entraînés.

Définition

RedisAI est un module de Redis qui fournit une implémentation de trois nouvelles structures de données (Tensor, Model, Script) afin de permettre d'exécuter des modèles de *machine learninig* et de gérer leurs données sur un serveur **Redis**. Il fournit une prise en charge prête à l'emploi pour les *frameworks* les plus utilisés (Tensorflow, Pytorch, ONNX [a]). **RedisAI** maximise à la fois le débit de calcul et réduit la latence, tout en simplifiant le déploiement des modèles de *machine learning*.

a. **ONNX** pour *Open Neural Network Exchange* est un format ouvert conçu pour représenter des modèles d'apprentissage automatique. Plus d'informations sur onnx.ai

2. **Kubernetes** est une solution d'orchestration de conteneurs applicatifs, il permet de déployer des applications conteneurisées sur tout type d'infrastructure IT et de gérer de façon centralisée les différentes ressources dont elles ont besoin. Certaines de ces informations sont extraites de cette page web.

2.3 Environnements de développement

Comme tout projet informatique, le développement de l'application du chatbot a nécessité d'avoir plusieurs environnements indépendants afin de développer, tester et déployer l'application. Les environnements mis en place sont :

- **Un environnement local :** C'est l'environnement sur lequel l'application est développée et testée par les développeurs. L'environnement est configuré sur des machine locales, et la gestion des versions est effectuée avec l'utilisation d'un dépôt **Git**.
- **Un environnement de développement :** Il permet de rendre l'application fonctionnelle et consultable en la déployant sur un serveur interne afin de permettre aux intervenants internes (médiateurs, responsables, développeurs) de la tester et de faire remonter les problèmes éventuels ou les demandes de nouvelles fonctionnalités.
- **Un environnement de production :** Il s'agit de la destination finale du processus de développement. C'est le serveur qui héberge l'application finale qui peut être utilisée par les clients.

Afin de faciliter le déploiement, des scripts **Shell** ainsi que des tâches (*jobs*) dans *Github Actions* ont été introduits pour publier des images **Docker** contenant le code source de l'application sur **Dockerhub** [3] et les récupérer par le serveur **Kubernetes**.

3 Présentation du sujet

Une session d'un *escape game* virtuel se fait en 3 étapes :

- Une première partie pendant laquelle les participants sont accueillis par un médiateur et invités à remplir un questionnaire.
- Une deuxième étape où les joueurs entrent dans le jeu et rassemblent des indices qui leur permettent de passer d'une salle de jeu à l'autre.
- Une dernière étape où le médiateur demande aux participants de répondre à des questions et donner leur avis sur les différentes thématiques du jeu. Le médiateur essaye de corriger les fausses idées que peuvent avoir les participants et leur proposer des astuces qui leur permettront de favoriser des comportements vertueux de santé.

Le but du projet est de construire un chatbot pouvant dialoguer avec les utilisateurs pendant les sessions virtuelles d'un *escape game* par le biais d'un service de conversations automatisées, il permettra de réduire l'intervention des médiateurs dans les sessions de jeu. Le chatbot devra tourner dans un serveur **Django** et fournir une interface d'**API** permettant de communiqer avec lui. Le projet devra être composé de plusieurs services :

- Un service pour la collecte des questions et réponses qui serviront à l'entraînement des modèles de l'apprentissage automatique utilisés par le chatbot. Les questions seront

3. **Dockerhub** est un service fourni par **Docker** pour trouver et partager des images de conteneurs avec la possibilité de gérer l'accès aux référentiels privés. https://docs.docker.com/docker-hub/

catégorisées et enregistrées dans une base de données. Le service fournira un **CRUD** [4] qui permettra d'ajouter, consulter, supprimer et mettre à jour les données. Ce service permettra également d'ajouter des données en téléchargeant un fichier de données (en format `.csv` par exemple). Ce service sera destiné aux médiateurs afin qu'ils puissent rassembler les questions qui sont souvent posées pendant l'étape de *debriefing* des *escape games*.

- Un service pour l'entraînement du chatbot, celui-ci contiendra tout le flux de l'apprentissage automatique. Il devra permettre de récupérer les données pertinentes de la base de données, prétraiter les questions et les tags et construire et entraîner des modèles. Ce service fournira aussi un système de versionnage des modèles entraînés ainsi qu'une fonctionnalité permettant de déployer un modèle entrainé sur le serveur **Redis** dédié.
- Un service pour la gestion des prédictions. Celui-ci permettra d'effectuer des prédictions auprès du chatbot, les stocker dans la base de données et collecter le feedback des utilisateurs afin de faire valider les prédictions auprès des médiateurs. Les prédictions validées serviront à enrichir la base de connaissances du chatbot.

Chacun de ces services devra être développé dans une application [5] autonome dans le même projet **Django**. Quand il est nécessaire, le service fournira une interface **API** permettant de l'utiliser dans des applications externes. Cela sera le cas notamment pour le troisième service qui permet d'effectuer des prédictions.

4 Infrastructure du projet

Afin de déployer un projet de *machine learning*, une infrastructure particulière doit être mise en place afin de prendre en charge toutes les étapes du flux de travail de l'apprentissage automatique (sélection des modèles, entraînements, tests, déploiements...) et garantir les meilleures performances en production. Celle-ci commence à prendre forme dans le projet du chatbot avec l'introduction plusieurs fonctionnalités.

4.1 Les applications du projet Django

Afin de mettre en œuvre les bonnes pratiques de l'ingénierie logicielle (notamment la séparation des responsabilités [6]), Le projet **Django** a été divisé en trois applications :

- Une application de gestion de données d'entraînement.
- Une application pour effectuer des entraînements et gérer les modèles entraînés.

4. **CRUD** : acronyme en anglais (pour create, read, update, delete) qui désigne les quatre opérations de base permettant la gestion d'une collection d'éléments dans une base de données.

5. Dans **Django**, le terme application décrit un paquet Python qui fournit un certain ensemble de fonctionnalités. Les applications peuvent être réutilisées dans différents projets.

6. La séparation des responsabilités (ou des préoccupations) est un principe de conception qui vise à segmenter un projet en plusieurs parties, afin que chacune d'entre elles isole et gère un aspect précis de la problématique générale. Plus d'informations sur https://fr.wikipedia.org/wiki/Séparation_des_préoccupations.

- Une application de production et gestion des prédictions.

La figure 2 montre le rôle détaillé de chacune de ces applications.

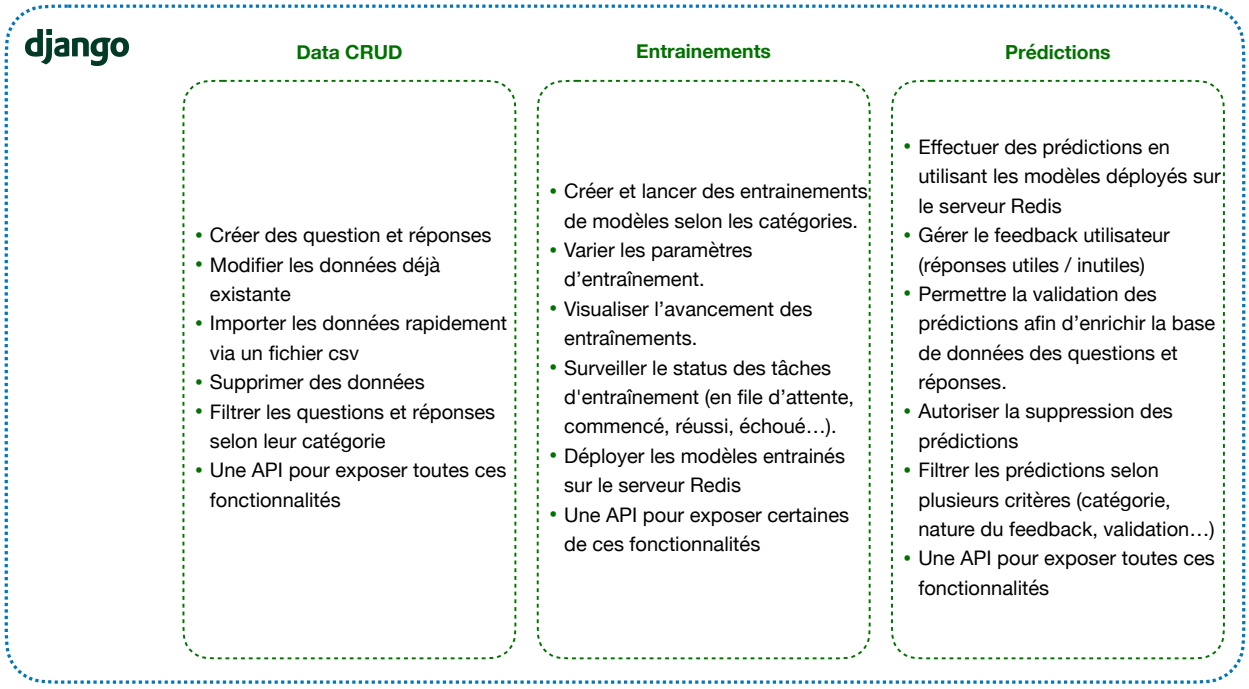

FIGURE 2 – Les différentes applications du projet du chatbot

4.2 Infrastructure des entraînements

La figure 3 présente le processus qui permet d'entraîner les modèles **tensorFlow** utilisés par le chatbot. Les étapes pour effectuer un ou plusieurs entraînements sont les suivantes :

- L'utilisateur sélectionne les catégories des modèles à entraîner et définit certains paramètres d'entraînement, les paramètres qui peuvent être définis actuellement sont le nombre d'epochs et une description de l'entraînement.
- Des tâches sont créées et insérées dans la base de données ainsi que dans la file d'attente du *worker* **Huey**.
- Pour chaque tâche, l'entraînement est effectué sur les données correspondantes en communicant l'avancement de l'entraînement à l'utilisateur (status de la tâche, barre de progression, problèmes éventuels...).
- Une fois un entraînement est terminé, l'utilisateur peut le déployer sur le serveur **Redis** ou le supprimer définitivement.

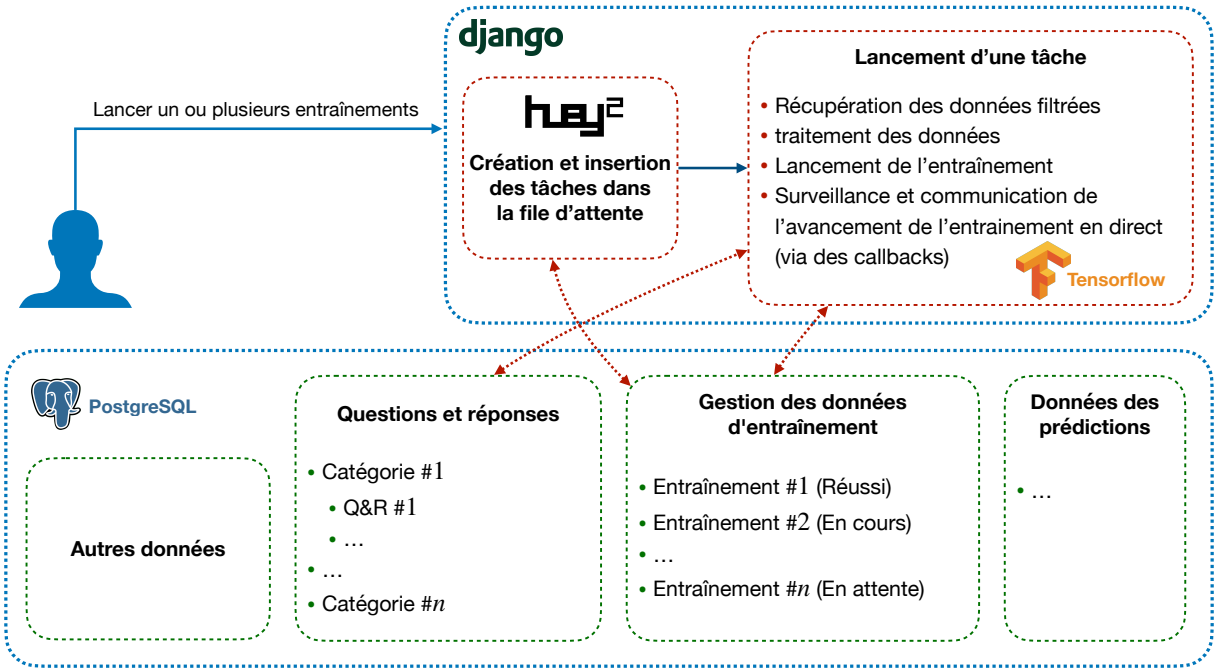

FIGURE 3 – Gestion des entraînements

Pour chaque entraînement, les données sont générées à partir des données collectées dans l'application de collecte des questions et réponses en suivant les étapes (représentées par la figure 4) suivantes :

1. D'abord, les données sont filtrées par l'**ORM** de **Django**[7] en fonction de la catégorie en cours d'entraînement.
2. Les questions sont obtenues à partir de l'attribut `text` des objets `Question` tandis que les tags (ou labels) sont générés à partir du champ `ID` des objets `Response`.
3. Les questions sont converties en vecteurs de nombres réels qui est le format pris en charge par les modèles **TensorFlow**.
4. Les *tags* sont convertis en représentation nombre / vecteur unique.
5. Le modèle **TensorFlow** est enfin alimenté avec ces données traitées.

7. **ORM** est l'acronyme en anglais de *object relational mapping*. Dans **Django**, il s'agit d'une bibliothèque qui automatise le transfert de données stockées dans des tables de base de données relationnelles vers des objets qui sont plus couramment utilisés dans le code d'application.

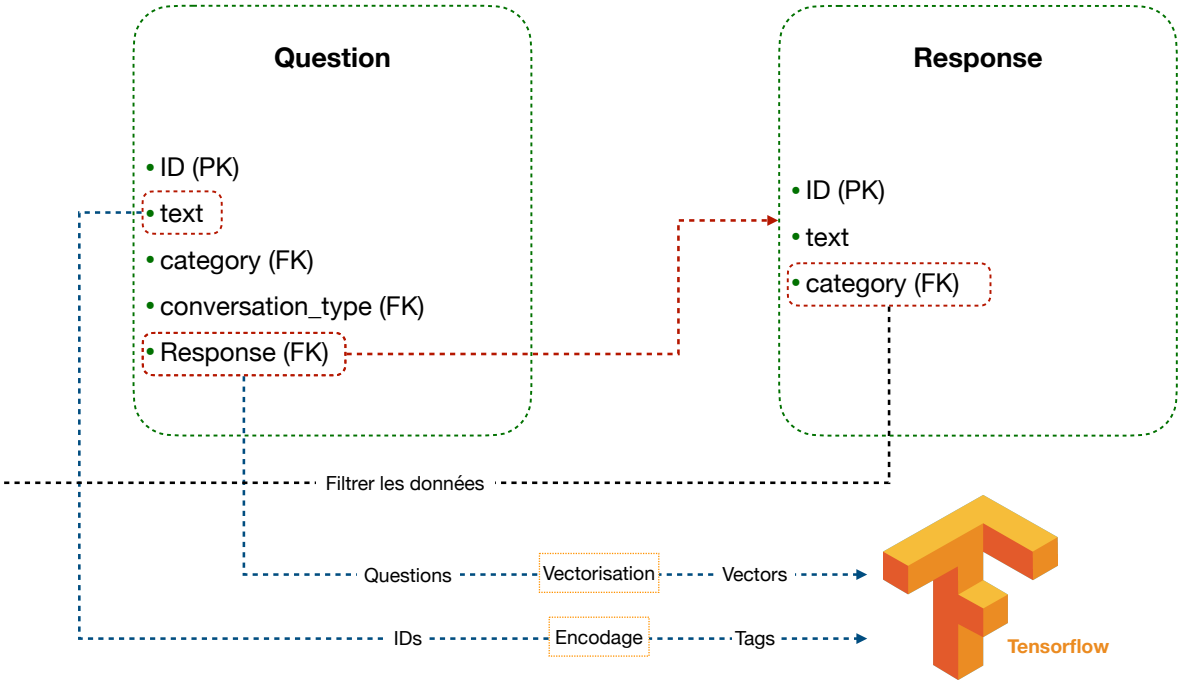

FIGURE 4 – Processus de traitement des données d'entraînement

4.3 Infrastructure de déploiement

Afin de fluidifier le processus des prédictions. Un serveur **Redis** a été mis en place, le module **RedisAI** est chargé sur le serveur afin que celui-ci soit capable de servir des modèles de *machine learning / deep learning*. Le projet **Django** du chatbot a également été connecté au service d'authentification unique qui permet aux utilisateurs d'accéder aux différentes applications en ne procédant qu'à une seule authentification. La figure 5 montre les différents services qui composent l'infrastructure de déploiement et la façon dont ils interagissent entre eux.

Une prédiction s'effectue donc en passant par les étapes suivantes :

- Le client pose la question dans la fenêtre du chatbot dans l'application du *frontend*. La détéction de la catégorie de la question est actuellement de la responsabilité de l'application du *frontend*.
- Ces informations sont envoyées à l'application du chatbot via une requête API protégée par un *token* généré d'une manière sécurisée via le standard **JWT** [8].
- La question passe dans un flux de traitement de langage naturel et est ensuite envoyée dans le serveur **Redis** par le biais d'une interface client.
- Le serveur **Redis** fournit dans son interface client des méthodes qui permettent de lancer la tâche de prédiction et récupérer le résultat. Une fois la prédiction est terminée, le résultat est renvoyé dans l'application **Django**.

8. **JWT** (pour *JSON Web Token*) est une stratégie d'authentification utilisée par les applications qui adoptent une architecture client / serveur. Elle permet l'échange sécurisé de *tokens* entre les différentes parties. Cette sécurité de l'échange se traduit par la vérification de l'intégrité et de l'authenticité des données.

- L'application insère les données de la prédiction dans la base de données et renvoie le résultat au client via une réponse API.

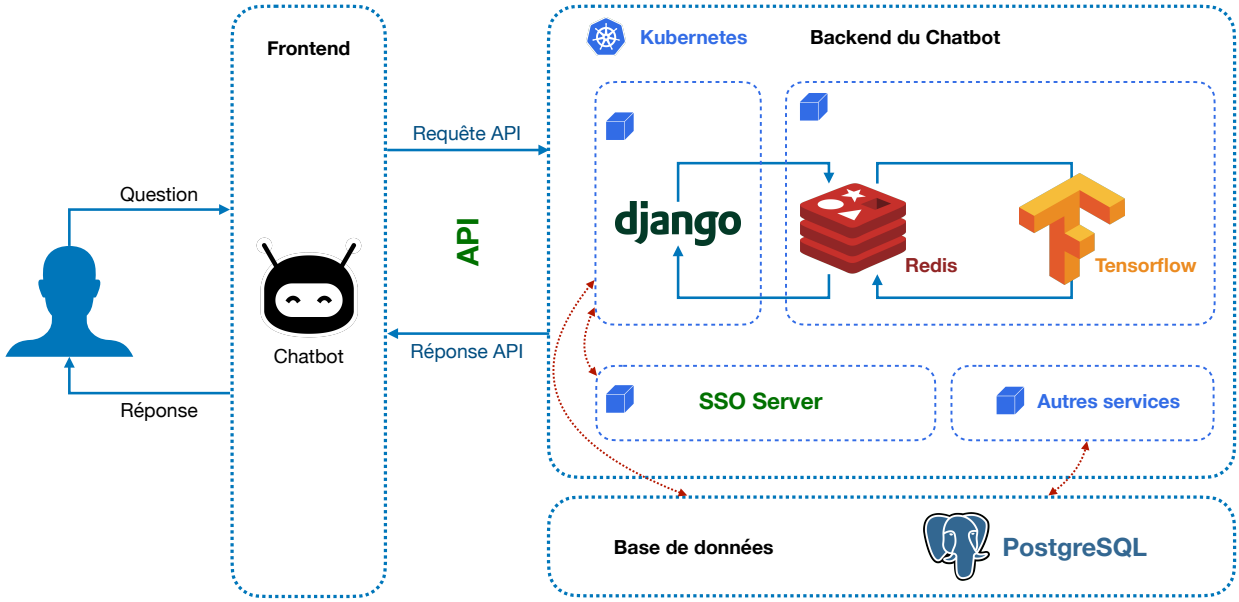

FIGURE 5 – L'infrastructure de déploiement du projet du chatbot

5 Prochaines étapes

Les étapes à venir dans le cadre du projet du chatbot comportent principalement les tâches suivantes :

- La mise en place d'une technique d'évaluation des performances des modèles entraînés suivant plusieurs critères (exemple : *accuracy* et *loss* d'entraînement et de validation).
- Le développement d'une fonctionnalité qui permettra de déployer constamment les modèles avec les meilleures performances.
- L'automatisation de la détection de la catégorie de la question posées par l'utilisateur à l'étape des prédictions.
- La mise en œuvre d'une fonctionnalité de suggestion d'indices et conseils pendant le jeu. Cette fonctionnalité est indépendante de l'architecture actuelle du chatbot et sera intégrée directement dans les jeux.

6 Conclusion

Mon stage à **Tricky** est une expérience pleine d'apprentissage. Il m'a permis de mettre en pratique les connaissances apprises le long des trois années à l'école afin de mener à bien les différentes tâches qui me sont confiées. Pendant les trois premiers mois, j'ai pu découvrir les différentes technologies et méthodes utilisées en entreprise afin de faciliter la tâche de

développement et la rendre à la fois efficace et rigoureuse. Les sessions de revue de code font partie de ces méthodes, il s'agit de séances hebdomadaires de quelques heures dont le but est de présenter aux autres des méthodes astucieuses utilisées pour résoudre une problématique, ou faire de la programmation en groupe (mob programming). J'ai également eu l'occasion de découvrir les activités de l'entreprise notamment en expérimentant certains services fournis par celle-ci.

Annexes

A Outils de gestion de projet

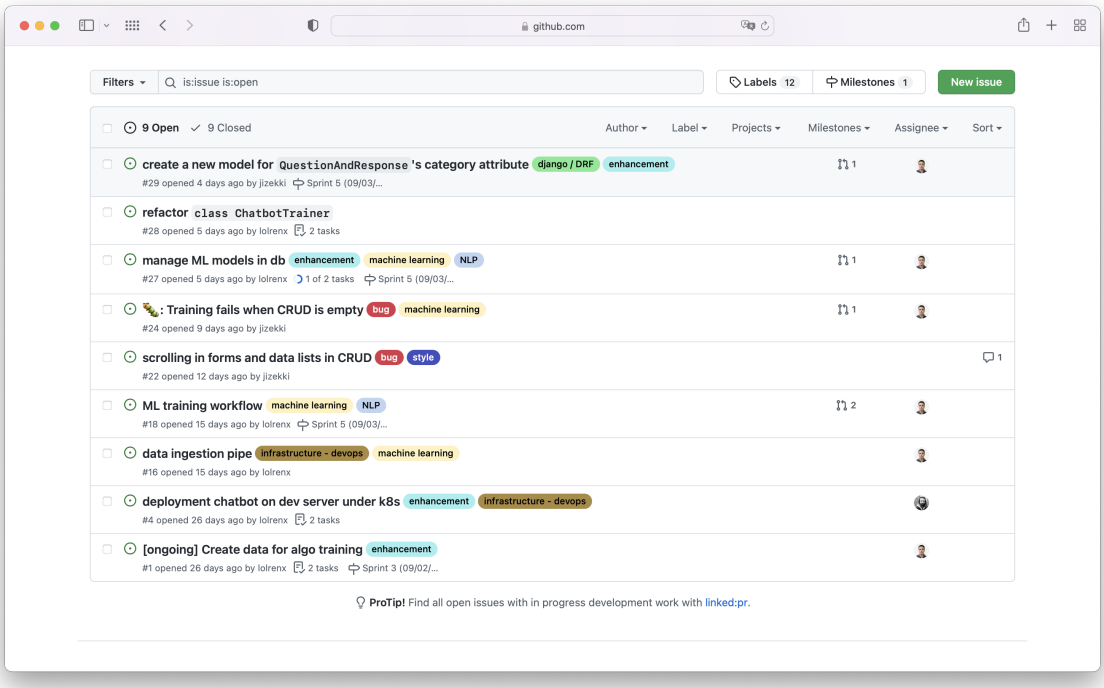

FIGURE 6 – Github Issues (04/03/2022)

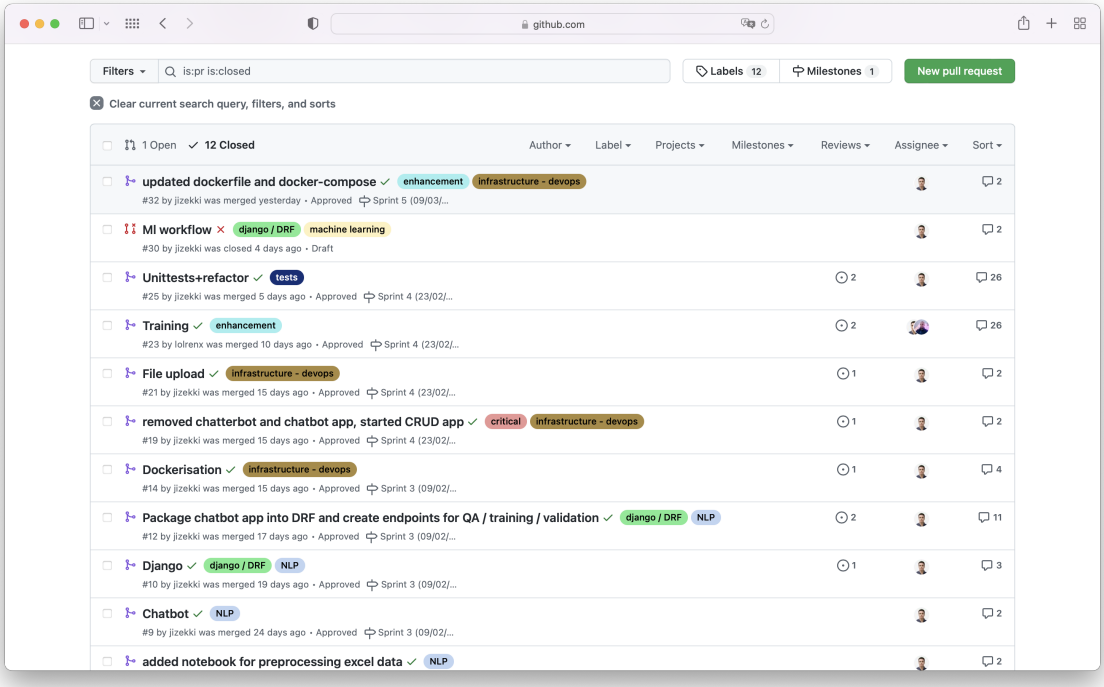

FIGURE 7 – Github Pull requests (04/03/2022)

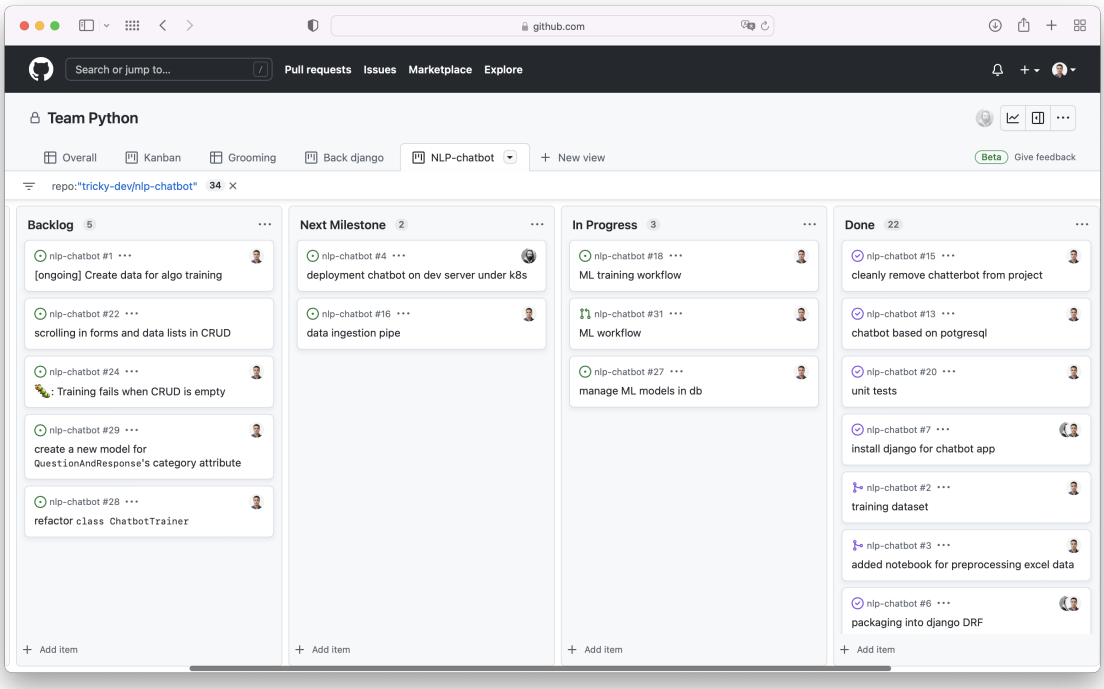

FIGURE 8 – Github Projects (05/03/2022)