2023 - 2024

Safe Bear

LUTTER CONTRE LE CYBERHARCELEMENT

COMPTE RENDU D'ALTERNANCE

LEO LAROU-CHALOT
IPSSI | PARIS
Paris

Lutte contre le cyberharcelement

L'entreprise SafeBear s'est donnée comme mission de proposer aux particuliers comme aux professionnel la possibilité de lutter contre le cyberharcèlement, en développant une solution visant à rendre possible la traçabilité et l'intengibilité des comportements liés au cyberharcèlement en sauvegardant ceux-ci dans la Blockchain.

A une époque ou tout est numérique, ce genre de problématique doit être démystifiée et permettre la fin d'une omerta.

SafeBear proposera à terme une application proposant les outils nécessaire à la protection des victimes de cyberharcèlement. L'idée est d'analyser les messages reçus sur les différents réseaux et plateformes numériques afin de les analyser, et de prevenir en cas de comportement jugé toxique. Pour des raisons évidentes, les détails liés aux différentes missions auxquelles j'aurai participé ne pourront être abordés.

Pour tout renseignement supplémentaires, l'entreprise est présentée à cette adresse : https://safebear.ai

Cette documentation a pour objectif de vous parler de mon rôle au sein de cette entreprise, des problématiques rencontrées ainsi que les solutions qui ont été mises en place.

Table des matières

La preuve de concept

Avant toute chose, je pense qu'il est important de préciser le contexte de travail dans lequel j'ai évolué.

Durant mon apprentissage au sein de l'entreprise, j'ai eu l'opportunité de travailler sur des missions qui étaient structurées sous forme de preuves de concept. Ces projets étaient axés sur la validation de nouvelles idées et l'exploration de solutions innovantes. Plutôt que de se lancer immédiatement dans des développements complets, l'approche consistait à créer des démonstrations pratiques pour évaluer la faisabilité technique et fonctionnelle des concepts envisagés.

Cette approche a offert plusieurs avantages. Tout d'abord, elle a permis de tester rapidement différentes idées, en identifiant les forces et les faiblesses de chaque approche. En réalisant des itérations rapides, nous avons pu ajuster nos stratégies en fonction des résultats obtenus. De plus, cette méthode a contribué à minimiser les risques associés aux projets, en fournissant des insights concrets avant de s'engager dans des développements plus importants.

Ces missions en tant que preuves de concept ont également été un moyen efficace de convaincre les parties prenantes de l'entreprise de la pertinence des projets. Les démonstrations pratiques ont servi à illustrer concrètement les avantages potentiels et à obtenir le soutien nécessaire pour passer à la phase suivante du développement.

En résumé, mon expérience au sein de l'entreprise m'a permis de participer activement à des projets de preuves de concept, une approche qui a prouvé être à la fois agile et stratégique dans le processus d'innovation de l'entreprise.

Récolte de l’information et apprentissage supervisé

Le modèle d’intelligence développé et breveté par SafeBear a pour objectif l’analyse et la compréhension des comportements toxique sur Internet. Pour arriver aux résultats actuels, il aura fallu entraîner ce modèle en lui fournissant de l’information à analyser. Il existe pour cela des *Datasets* dédié à l’apprentissage.

Un "dataset" (ensemble de données en français) est simplement une collection de données. Dans le contexte de l'entraînement d'un modèle d'intelligence artificielle (IA), un dataset est une sélection organisée de données utilisée pour former et évaluer un modèle. Ces données peuvent prendre différentes formes en fonction du type de problème que vous cherchez à résoudre.

Dans le contexte SafeBear, ces données sont sous forme de texte pour le traitement du langage naturel **(NLP)**.

Il aura ensuite fallut procéder à un **apprentissage supervisé** de ce modèle.

L'apprentissage supervisé est un peu comme enseigner à un ordinateur en lui montrant des exemples étiquetés. C'est un peu comme apprendre à un enfant en lui montrant des images avec des descriptions. Dans le cas de la compréhension du langage naturel, on donne à l'ordinateur des morceaux de texte (peut-être des phrases ou des paragraphes) et on lui dit ce qu'ils signifient.

Étape 1 : Dataset pour la compréhension du langage naturel :

On commence par rassembler un ensemble de données (dataset) qui contient des exemples de texte avec des informations sur ce que le texte signifie. Par exemple, si on veut apprendre à un modèle à comprendre la différence entre une phrase « toxique » et une « non toxique », le dataset aura des exemples de phrases étiquetées comme telles.

Étape 2 : Entraînement du modèle :

Ensuite, on utilise ce dataset pour entraîner le modèle d'IA. Le modèle apprend à associer les caractéristiques du texte (comme les mots et les phrases) aux étiquettes ou aux significations correspondantes. C'est comme si l'ordinateur apprend à reconnaître les schémas dans le langage naturel.

Étape 3 : Test et utilisation :

Une fois que le modèle est entraîné, on le teste avec de nouveaux exemples pour voir s'il peut comprendre et attribuer correctement les étiquettes. En fin de compte, le modèle devrait être capable de comprendre de nouveaux morceaux de texte qu'il n'a jamais vus auparavant, grâce à ce qu'il a appris pendant l'entraînement.

C’est sur cette 3ème étape que j’ai débuté lors de mon stage dans l’entreprise. Notre tâche, aux personnes travaillant sur ce sujet et moi-même était d’indiquer si oui ou non, le modèle *« comprenait »* correctement les exemples que nous lui fournissions.

Cette tâche peut sembler redondante, et c’est évidemment le cas, mais non moins importante pour autant. C’est sur cette base que le modèle affûtera sa compréhension du langage naturel.

Développement d’un bot Twitch

L’une des premières missions en autonomie aura été celle de réaliser un **bot** pour rendre possible la solution SafeBear sur certaines plateformes de messagerie et de streaming.

Cette première mission a été réalisée dans un concept de « preuve de concept ».

Une "preuve de concept" est une **démonstration pratique** ou une **réalisation partielle** d'un projet, conçue dans le but de **prouver la faisabilité** ou la **viabilité** d'une idée ou **d'une solution technologique**. C'est essentiellement une étape préliminaire avant de s'engager pleinement dans le développement d'un produit ou d'un système.

En l’occurrence, j’ai commencé par **Twitch**.

Proposition de solution d’implémentation

Initialement, l’idée était de pouvoir proposer la solution SafeBear sur la plateforme Twitch afin de l’utiliser pour récupérer les messages reçus par le propriétaire de la chaîne pour les analyser et le protéger.

Problèmes rencontrés

Au fil de ma reflexion, plusieurs problèmes se sont présentés.

1. La technologie à utiliser (langages, bibliothèques, Framework)
2. La récupération des messages au nom du propriétaire de la chaîne (autorisation).
3. La récupération des messages sur une plage de temps donné en cas d’interruption du service.

Au-delà de ces trois points, c’est toute la logique de communication et d’autorisation entre notre solution et la plateforme Twitch à l’aide de l’API proposée par la plateforme qu’il aura fallu prendre en main en un minimum de temps.

Solutions proposées

Après quelques échanges avec le développeur senior qui m’avait sous sa responsabilité, j’ai décidé d’utiliser Python pour mon implémentation pour deux raisons.

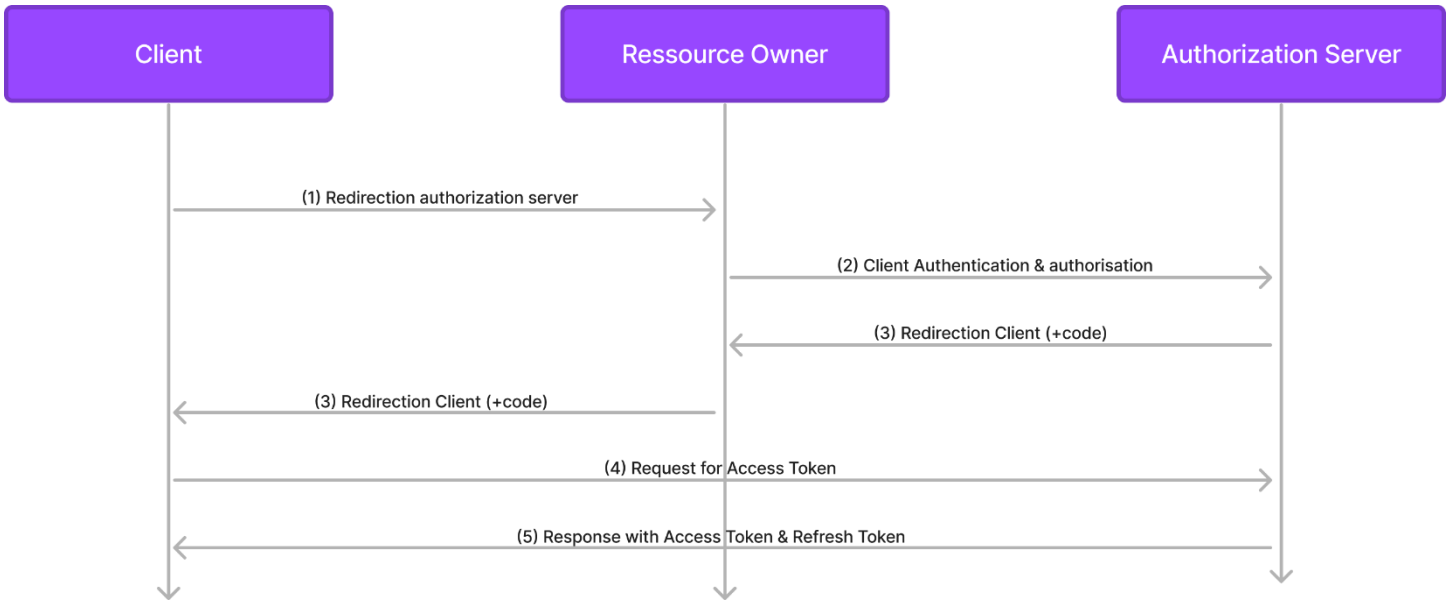
La première était que tout le reste de l’application était développé en python en ce qui concerne la récupération et la manipulation de l’information. Il me semblait donc judicieux de rester sur la même technologie. La seconde raison est qu’une bibliothèque Python était déjà spécialement dédiée à cette utilisation. L’utilisation de NodeJs aurait pu être possible, mais me semblait plus complexe, notamment sur l’utilisation des web sockets (ceux-ci étant déjà partiellement exploité en profondeur dans la bibliothèque python).

Mon projet s’est donc articulé autour d’un conteneur Docker de développement afin de créer un environnement de développement immuable et plus simple à mettre en place.

A l’intérieur, le module **Poetry** a été installé pour ma gestion des dépendances des différents modules utiles à ce projet. L’outil de formatage de document python **Black** a été utilisé afin de suivre les règles de mise en forme des différents fichiers du projet. Plus largement, le plugin **Ruff** disponible dans la collection de VsCode, plugin contenant entre autres **Black** et **iSort** a été installé.

En ce qui concerne l'accès aux ressources de l'utilisateur par l'intermédiaire de l'application, l'API proposée par Twitch utilise le protocole d'authentification **OAuth2.0** (une documentation disponible sur mon portfolio passe en revue ce protocole plus en détail) qui permet au client d'agir au nom de l'utilisateur qui s'est authentifié auprès des serveurs de ressources afin de lui permettre l'utilisation de celle-ci, sans avoir à connaître les identifiants personnels de cet utilisateur.

Cette technologie (**OAuth2.0**) est très largement utilisée dans ce genre de situation. Cette technologie permet de garantir la confidentialité des identifiants de l'utilisateur, grâce à un « *jeton d'accès* » (token) déchiffrable par le serveur de ressources.



Le dernier souci venait du fait que Twitch utilise des serveurs IRC.

Un serveur IRC (Internet Relay Chat) est un système qui facilite la communication en temps réel entre utilisateurs via des canaux de discussion. IRC est un protocole de communication textuelle qui permet aux utilisateurs de discuter les uns avec les autres dans des canaux thématiques en temps réel. Voici quelques points clés pour comprendre ce qu'est un serveur IRC :

- **Protocole IRC :** IRC est basé sur un protocole de communication spécifique qui définit comment les messages sont échangés entre les clients (utilisateurs) et les serveurs IRC. C'est un protocole ouvert et standardisé.
- **Canal de discussion :** Les utilisateurs se connectent à un serveur IRC et rejoignent des canaux de discussion. Chaque canal est un espace de discussion dédié à un sujet particulier, et les utilisateurs peuvent échanger des messages en temps réel dans ces canaux.
- **Serveur IRC :** Un serveur IRC est une machine qui exécute un logiciel de serveur IRC. Il gère la connexion des utilisateurs, les canaux de discussion, les messages, etc. Les serveurs IRC sont interconnectés, ce qui signifie que les messages peuvent être relayés entre différents serveurs pour permettre une communication à l'échelle mondiale.
- **Client IRC :** Les utilisateurs se connectent aux serveurs IRC à l'aide de clients IRC, qui peuvent être des applications spécifiques, des clients web ou des clients intégrés à des logiciels de messagerie.

IRC a été développé dans les années 1980 et a été l'un des premiers moyens de communication en ligne en temps réel. Bien que son utilisation ait diminué avec l'avènement des messageries instantanées et des réseaux sociaux, IRC reste populaire dans certains contextes, notamment dans les communautés open source, les discussions techniques et les jeux en ligne.

Prise en main de GraphQL

Introduction

Ce projet avait pour objectif de mettre en œuvre et d'explorer plusieurs technologies modernes telles que Poetry, Django, Strawberry et pyTest. Le principal défi était de développer une API GraphQL fonctionnelle en utilisant ces outils.

Contexte

Le projet SafeBear Django Strawberry est un exemple concret de développement d'API GraphQL. Il vise à créer une plateforme robuste et moderne en utilisant des technologies populaires du monde Python et du développement web.

Objectifs de la mission

- Mettre en place une structure de projet Django avec Poetry.
- Implémenter un schéma de données GraphQL avec Strawberry.
- Gérer les fixtures pour peupler la base de données avec des données de test.
- Réaliser des tests unitaires avec pyTest.
- Documenter le projet de manière complète et compréhensible.

Réalisation des objectifs

Structure de projet Django avec Poetry

La mise en place de la structure du projet avec Poetry a été réalisée avec succès, facilitant la gestion des dépendances et l'environnement virtuel.

Schéma de données GraphQL avec Strawberry

Les schémas de données GraphQL ont été définis avec Strawberry, permettant la création d'une API flexible et adaptée aux besoins du projet.

Gestion des fixtures

Les fixtures ont été utilisées pour préremplir la base de données avec des données de test. Cela a été essentiel pour garantir le bon fonctionnement des tests unitaires.

Tests unitaires avec pyTest

Des tests unitaires complets ont été mis en place pour vérifier le bon fonctionnement des différentes parties du projet, assurant une qualité logicielle élevée.

Documentation exhaustive

La documentation du projet a été rédigée de manière complète, couvrant tous les aspects du développement, de l'installation à l'utilisation en passant par la configuration.

Difficultés rencontrées

- L'utilisation de fixtures et la gestion des données de test ont posé quelques défis, mais ont été résolues avec succès.
- Des erreurs liées à l'absence de modules Django ont été résolues en ajustant l'environnement de développement.

Résultats obtenus

- Une API GraphQL fonctionnelle avec une base de code propre et organisée.
- Des tests unitaires exhaustifs assurant la stabilité du projet.
- Une documentation claire et accessible, facilitant la compréhension et la prise en main du projet.

Perspectives d'amélioration

- Explorer des fonctionnalités avancées de Strawberry pour optimiser les requêtes GraphQL.
- Ajouter des fonctionnalités supplémentaires à l'API pour enrichir l'expérience utilisateur.
- Continuer à suivre les meilleures pratiques de développement et de test pour maintenir la qualité du code.

Conclusion

La mission sur ce projet a été une expérience enrichissante, permettant d'approfondir les connaissances sur le développement d'API GraphQL et d'acquérir une expertise pratique sur des technologies modernes. Le projet est fonctionnel, bien documenté et prêt à être présenté au jury.

Templating de mails

MJML, acronyme de "Mailjet Markup Language", est un langage de balisage conçu spécifiquement pour simplifier la création de templates d'e-mails. Contrairement au HTML traditionnel, MJML offre une approche plus intuitive et simplifiée, permettant aux développeurs et aux concepteurs de concevoir des e-mails réactifs et esthétiquement attrayants sans avoir à jongler avec les complexités du rendu sur les différents clients de messagerie.

L'adoption de MJML présente plusieurs avantages pour une entreprise, notamment dans le contexte d'un service dématérialisé avec un cycle de vie d'abonnement :

Gain de temps et d'efficacité

En utilisant MJML, les équipes peuvent concevoir et personnaliser rapidement des templates d'e-mails, réduisant ainsi le temps nécessaire pour mettre en place des campagnes de mailing. Cela permet également de réagir plus rapidement aux besoins des clients et aux changements dans le cycle de vie de l'abonnement.

Compatibilité multiplateforme

Les e-mails conçus avec MJML sont optimisés pour être compatibles avec une large gamme de clients de messagerie, garantissant ainsi une expérience cohérente pour les utilisateurs quel que soit le dispositif ou le logiciel qu'ils utilisent pour accéder à leurs e-mails.

Réactivité et adaptation

Les templates MJML sont conçus pour être réactifs, c'est-à-dire qu'ils s'ajustent automatiquement à la taille de l'écran sur lequel ils sont consultés. Cela est particulièrement important dans le contexte des services dématérialisés, où les utilisateurs peuvent accéder à leurs e-mails depuis différents appareils, tels que des smartphones, des tablettes ou des ordinateurs de bureau.

Personnalisation et suivi des performances

Grâce à la possibilité d'intégrer des données dynamiques dans les templates MJML, les entreprises peuvent personnaliser facilement le contenu des e-mails en fonction des préférences et du comportement des utilisateurs. De plus, les fonctionnalités de suivi intégrées permettent d'analyser les performances des campagnes de mailing et d'ajuster les stratégies en conséquence pour maximiser l'engagement et la rétention des abonnés.

Résumé

En résumé, l'utilisation de MJML pour la conception de templates d'e-mails offre aux entreprises une solution efficace et flexible pour gérer les communications avec leurs clients tout au long du cycle de vie de l'abonnement à un service dématérialisé. En simplifiant le processus de création, en garantissant la compatibilité multiplateforme et en permettant la personnalisation et le suivi des performances, MJML contribue à améliorer l'efficacité et l'impact des campagnes de mailing, tout en offrant une expérience utilisateur optimale.

Playwright : Automatisation de Tests End-to-End pour une Fiabilité Accrue

Dans le cadre de mon apprentissage en informatique, j'ai investi du temps et des efforts dans l'exploration et la maîtrise de Playwright, un outil puissant pour automatiser les tests end-to-end. Playwright est une bibliothèque open source développée par Microsoft, conçue pour faciliter la validation de l'intégrité et de la fonctionnalité d'une application web à travers divers navigateurs.

Comprendre Playwright

Playwright offre une approche polyvalente pour l'automatisation des tests, en prenant en charge plusieurs navigateurs tels que Chrome, Firefox, Safari et Edge. Ce qui le distingue, c'est sa capacité à interagir avec les éléments de la page web de manière réaliste, reproduisant ainsi le comportement utilisateur de manière précise. Cela inclut des actions telles que le clic, la saisie de texte, le défilement et même la manipulation des périphériques de géolocalisation et de capteurs.

Avantages pour l'entreprise

Adopter une approche de tests end-to-end avec Playwright offre plusieurs avantages pour une entreprise axée sur la prestation de services dématérialisés, notamment :

1. Fiabilité accrue : En automatisant les tests end-to-end, les entreprises peuvent garantir une meilleure fiabilité de leur application en identifiant rapidement les problèmes potentiels à chaque étape du processus.
2. Réduction des coûts : En automatisant les tests, les entreprises peuvent réduire les coûts associés aux tests manuels répétitifs, tout en garantissant une couverture exhaustive des scénarios d'utilisation.
3. Gain de temps : Playwright permet l'exécution rapide et simultanée de tests sur plusieurs navigateurs et plates-formes, ce qui accélère le processus de validation et de déploiement des fonctionnalités.
4. Meilleure expérience utilisateur : En détectant et en corrigeant les bogues potentiels avant qu'ils n'affectent les utilisateurs, les entreprises peuvent offrir une expérience utilisateur plus fluide et fiable, renforçant ainsi leur réputation et leur fidélité.

Résumé

En conclusion, mon acquisition de compétences avec Playwright représente un investissement précieux dans la qualité et la fiabilité des applications web, offrant à l'entreprise une base solide pour répondre aux attentes croissantes des utilisateurs dans un environnement numérique en constante évolution.