Rapport pour le projet d'intégration de WhatsApp Business API avec GPT

Partie 1 : Familiarisation avec l'API WhatsApp Business

Objectifs :

L'objectif sera de rendre une API WhatsApp Business fonctionnelle, capable de gérer une conversation complète.

Pour cela, nous avons suivi les étapes suivantes :

1. Nous avons d'abord accédé à la plateforme Facebook Developers via le lien https://developers.facebook.com/ et nous nous sommes connectés à notre compte développeur que nous avions préalablement créé.
2. Une fois connectés, nous avons accédé au tableau de bord des développeurs, où nous avons créé une application nommée *BarkaLabGPT*. Pour le type d'application, nous avons sélectionné "Autre", puis opté pour l'option "Entreprise", adaptée aux entreprises qui utilisent l'API pour interagir avec leurs clients.
3. Après la création de l'application, nous avons ajouté WhatsApp comme produit.
4. Nous avons ensuite configuré un numéro de téléphone WhatsApp. Suite à cela, Facebook nous a fourni un identifiant de compte ainsi qu'un jeton d'accès, nécessaires pour utiliser l'API.
5. Enfin, pour gérer une conversation complète avec un utilisateur via l'API, nous avons utilisé une commande (voir ci-dessous) qui nous permet d'envoyer des messages à l'aide de l'API WhatsApp Business, intégrée dans l'API Graph de Meta. Nous utilisons un *endpoint* spécifique pour envoyer ces messages.

```
curl -X POST https://graph.facebook.com/v14.0/391996297338551/messages ^

-H "Authorization: Bearer
EAAKxPL6ZCFBgBOxta2U0ybHZCxHQralMFWICayXOqSF9kDUClRCOfGe3n51MdS00ZAUY4RE
4cKoAKuZA4djycl1dgUJJIZBftsOwGg9pQpGZAuELtZB2Eokf3CCI5PabW4okJWrDmVfY9BB0cESu
FL3o1G0IVUis3cNltgC8iBj6ZA9aRdPKtUMW4561ZAjtav1ahGjq4wXeJYUrZAuCXFmQWLF9i0V7
kZD" ^

-H "Content-Type: application/json" ^

-d "{\"messaging_product\": \"whatsapp\", \"to\": \"+22673693576\", \"type\": \"text\", \"text\":
{\"body\": \"Bonjour, c'est un message de test depuis l'API WhatsApp !\"}}"
```

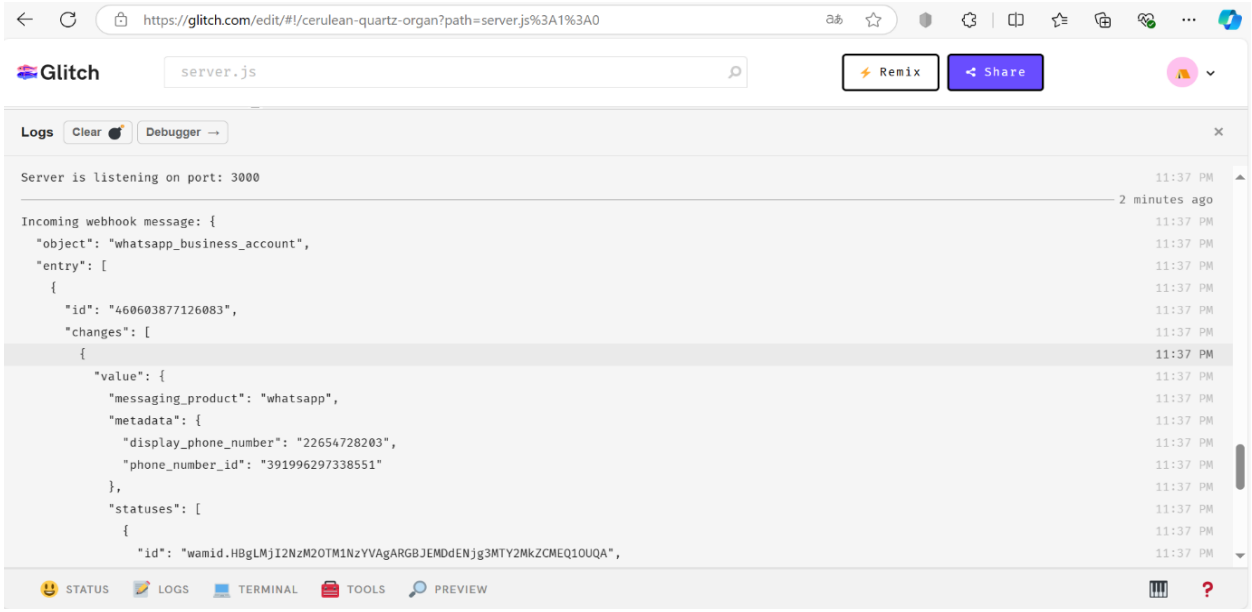
Pour la réception des messages via l'API WhatsApp Business, nous utiliserons des webhooks, ce qui nous permettra de recevoir des messages ainsi que des confirmations de lecture en temps réel concernant les modifications apportées au compte.

Pour afficher le contenu des webhooks, nous devrons ajouter une URL de rappel, que nous obtiendrons via notre serveur d'hébergement Glitch.

Concernant l'URL de rappel, nous avons utilisé Glitch pour héberger notre serveur. Nous avons configuré une application capable d'envoyer et de recevoir des messages en utilisant l'API WhatsApp Business. Après avoir obtenu un jeton d'accès et un identifiant de compte, nous avons réussi à envoyer un message de test avec succès. De plus, nous avons reçu des notifications de l'API pour confirmer la réception des messages.

Cependant, nous avons rencontré un problème de verrouillage de compte, ce qui nécessite de prouver la légitimité de notre entreprise pour continuer à utiliser l'API sans restriction.

Avec cette approche, nous pouvons dire que la conversation est entièrement gérée via l'API. Nous avons réussi à envoyer des messages et à recevoir des notifications de webhook en temps réel.

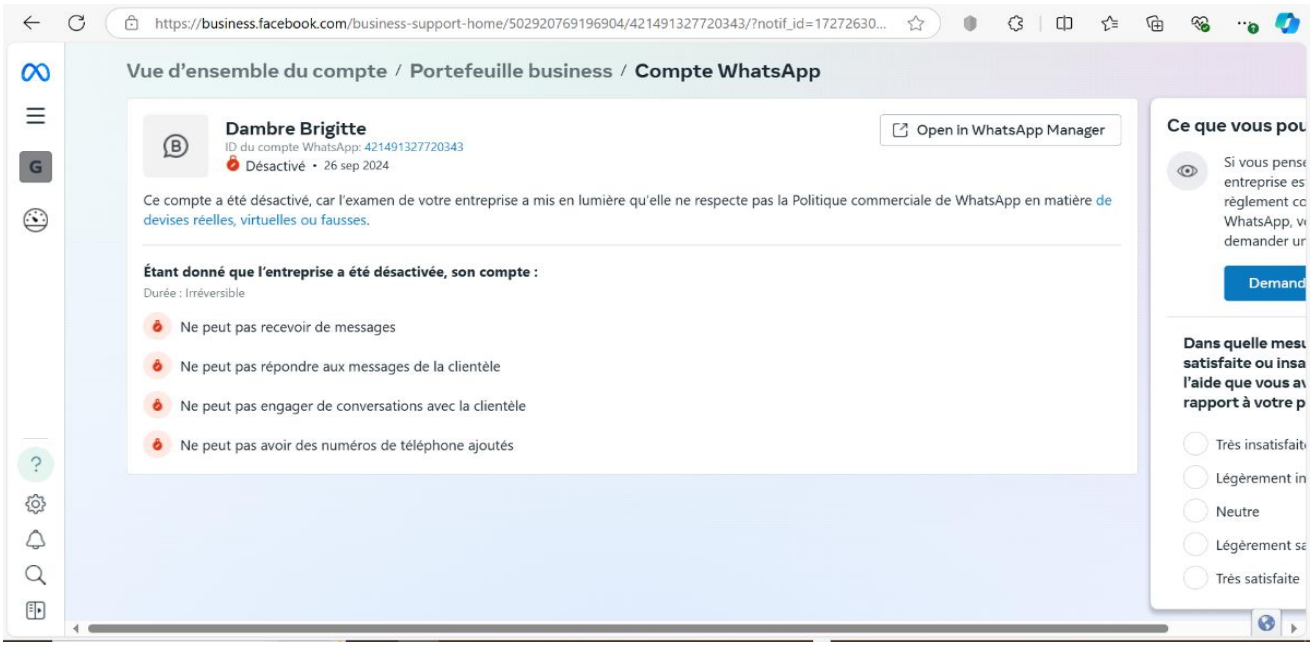


L'image ci-dessus illustre la preuve réelle que nous avons pu envoyer un message via l'API et recevoir la conversation depuis les journaux de notre serveur.

Difficultés rencontrées :

En ce qui concerne la réception des messages, nous avons rencontré des problèmes avec le numéro d'entreprise utilisé pour la création de notre application afin de pouvoir utiliser l'API.

Cependant, notre compte a été verrouillé, ce qui empêche le numéro du destinataire de recevoir le message que nous avons envoyé via l'API WhatsApp. Si le numéro (numéro d'entreprise) est actif et que le compte n'est pas verrouillé, le message devrait arriver sur l'application WhatsApp du destinataire. Cependant, dans notre cas, nous avons reçu une erreur indiquant que le compte est verrouillé, ce qui signifie que le message n'a pas pu être livré.



Partie 2 : Création et fine-tuning d'un modèle GPT

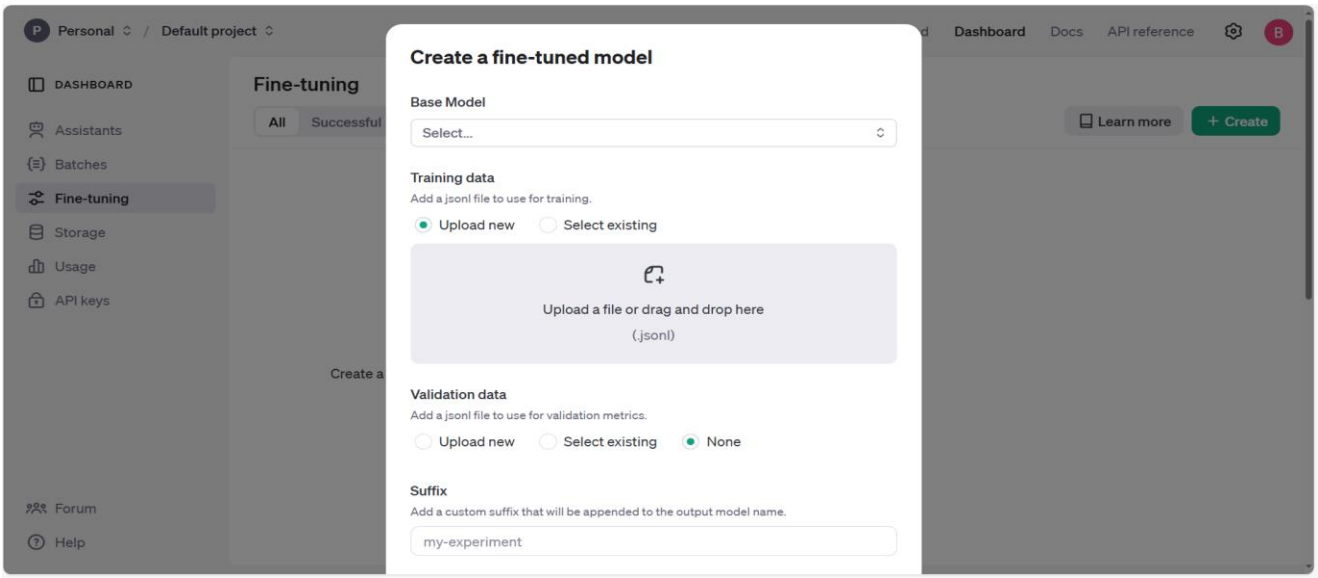
Objectif : Former et ajuster un modèle GPT capable de répondre aux questions des clients BarkaChange.

1. Le modèle GPT d'OpenAI est un modèle d'intelligence artificielle basé sur l'architecture des transformateurs, utilisé pour répondre à des questions et automatiser des conversations. Il convient de noter qu'OpenAI propose une API que nous pouvons intégrer dans notre projet pour tirer parti d'un modèle GPT déjà formé. Ce modèle offre des versions optimisées que nous pouvons affiner avec nos propres données.

   Tout d'abord, il faut :

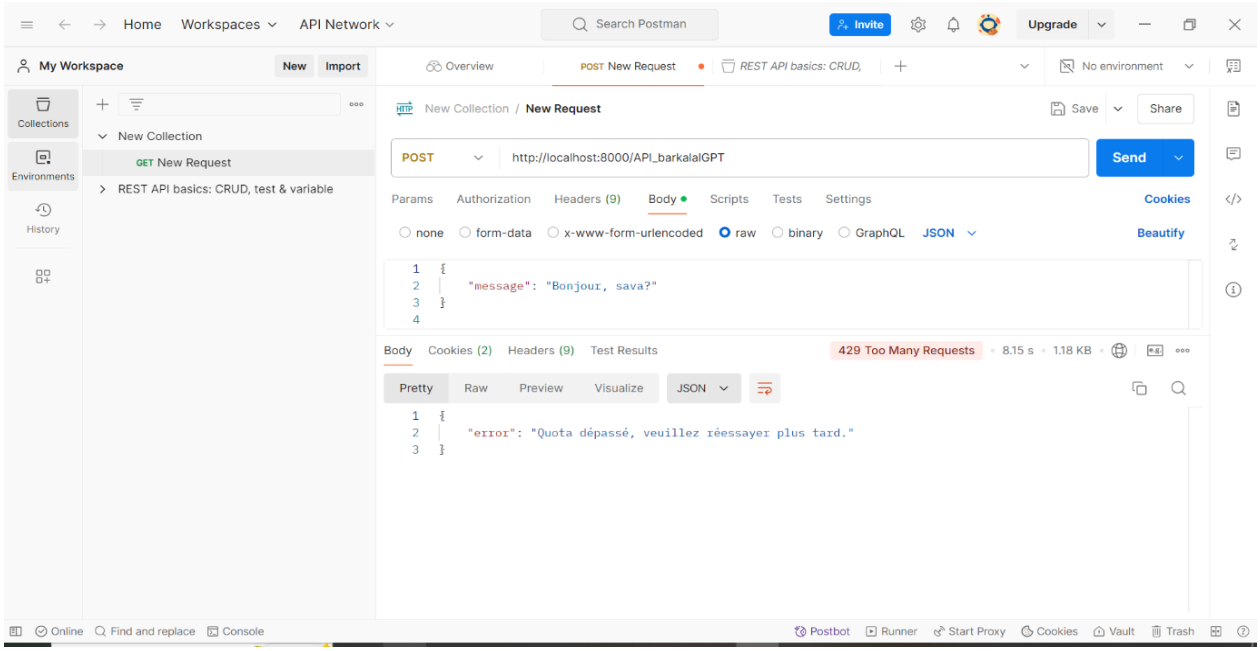
- Créer un compte OpenAI pour accéder à l'API GPT : une clé API nous sera fournie pour effectuer des appels API et tester le modèle.
- Envoyer des requêtes à l'API : nous enverrons des messages (prompts) au modèle et recevrons des réponses textuelles.
- Stocker des données d'entraînement : avec les ressources de BarkaChange, comme la FAQ, le guide, les services possibles et d'autres informations, nous allons exporter ces données dans un format structuré (CSV ou JSON).

2. Pour le **fine-tuning**, OpenAI nous permet d'ajuster un modèle GPT préexistant avec nos propres données (les ressources et la documentation disponibles sur BarkaChange. Cette fonctionnalité se trouve dans notre tableau de bord OpenAI, où nous pouvons choisir le modèle et uploader les ressources de BarkaChange au format JSON.



3. En ce qui concerne le test du modèle en simulant des interactions client, nous nous sommes connectés à notre compte OpenAI et avons généré une nouvelle clé API que nous avons intégrée dans le fichier. env de notre projet Laravel. Ensuite, nous avons installé OpenAI via la commande suivante : **« composer require openai/openai »**

Nous avons également créé un contrôleur pour gérer les interactions et défini une route correspondante. Par ailleurs, nous avons installé Postman, qui est un outil permettant de tester des API, et cela nous a permis de simuler des requêtes vers notre modèle.
Normalement, la compilation de nos requêtes devrait s'exécuter correctement, mais en raison de notre plan gratuit, nous avons rencontré des erreurs de quota, ce qui nous a empêchés de finaliser nos tests.



Conclusion :

En conclusion, notre projet a progressé significativement, en configurant avec succès l'API WhatsApp Business et en intégrant le modèle GPT via Laravel. Nous avons atteint nos premiers objectifs, mais des difficultés subsistent, notamment liées à des limitations de quota sur le plan gratuit de l'API OpenAI, ainsi qu'à des contraintes de temps pour finaliser l'ensemble des fonctionnalités.