P10 - Note méthodologique

Développez un chatbot pour réserver des vacances

Hourdin Charlène
Parcours Ingénieur IA
Mars 2023

Table des matières

1 Introduction **1**
1.1 Présentation du projet 1
1.2 Objectif de la note méthodologie 2
1.3 Contexte et enjeux du pilotage de la performance du modèle en production 2

2 Évaluation de la performance du modèle offline avec seuils de performance. **3**

3 Méthodologie de pilotage de la performance en production **4**
3.1 Dashboard : Tableau de suivi récaputilatif 5
3.2 Métrics : métriques de suivi de la performance 6

4 Évaluation de la performance du modèle en production **7**
4.1 Surveillance du nombre d'utilisateurs : effectuée en temps réel par Azure. 8
4.2 Surveillance du nombre de sessions : effectuée en temps réel par Azure. 8
4.3 Surveillance de la disponibilité : fondée sur la réponse de l'application au ping. 9
4.4 Le temps de réponse : la durée moyenne pour répondre aux requêtes. 10
4.5 Critère de dégradation de la performance 10

5 Schéma du mécanisme d'évaluation du modèle en production **12**

6 Dérive du modèle *(Drift)* **13**

7 Modalités de mise à jour du modèle **14**

8 Conclusion **15**

1 Introduction

1.1 Présentation du projet

Le projet lancé par **Fly Me** consiste en la création d'un chatbot pour aider les utilisateurs à choisir une offre de voyage.

Dans un premier temps, un **MVP *Minimum Viable Product*** doit être construit pour permettre aux employés de Fly Me de réserver facilement un billet d'avion pour leurs vacances.

La V1 du Chatbot doit être capable d'identifier cinq éléments dans la demande de l'utilisateur :

- ville de **départ**,
- ville de **destination**,
- **date aller** souhaiter du vol,
- **date retour** souhaitée du vol,
- **budget** maximum pour le prix total des billets.

Si un des éléments est manquant, le chatbot doit **poser les questions pertinentes** à l'utilisateur pour comprendre complètement sa demande. Le chatbot doit également **reformuler la demande de l'utilisateur** et lui demander de valider sa compréhension.

La stack technique utilisée est :

- **Azure**,
- le code source du framework de développement Python **Microsoft Bot Framework SDK v4 for Python**,
- le service cognitif **LUIS** d'Azure,
- le service **Web App d'Azure**,
- **Bot Framework Emulator**.

Le projet doit être travaillé de manière itérative et la chaîne de traitement, d'intégration et de développement doit être automatisée dès le début.

L'accès au dépôt GitHub de l'application est possible en utilisant le lien suivant : **https ://github.com/charleneho flyme**.

1.2 Objectif de la note méthodologie

L'objectif de cette note méthodologie est de décrire :

- les critères d'évaluation,
- le schéma du mécanisme d'évaluation,
- les modalités de mise à jour du modèle.

La mise en production d'un modèle d'apprentissage automatique ne se termine pas avec sa création et son entraînement. Il est crucial de suivre de près ses performances et de s'assurer qu'il est en mesure de résoudre les problèmes de manière satisfaisante.

Cette note méthodologie vise donc à fournir une approche structurée pour évaluer la performance et l'efficacité du modèle en production sur le long terme, ainsi que de détecter les éventuels problèmes et de définir les mesures correctives nécessaires.

En outre, cette note méthodologie énonce les modalités de mise à jour du modèle, telles que la fréquence et les seuils, qui permettent de garantir la pertinence et la précision du modèle en fonction des évolutions de l'environnement.

1.3 Contexte et enjeux du pilotage de la performance du modèle en production

La surveillance et l'analyse de la performance d'un modèle en production sont essentielles pour garantir son efficacité et sa pertinence dans un environnement opérationnel.

Cela permet de mesurer et d'analyser les résultats du modèle en temps réel, de détecter rapidement les erreurs et les problèmes de performance, et de prendre des mesures correctives pour améliorer la qualité et l'efficacité du modèle.

La performance du modèle en production est un enjeu important pour les entreprises, car elle impacte directement leur productivité et leur rentabilité. Il est donc important de garantir sa fiabilité et sa stabilité afin d'assurer **une expérience utilisateur de qualité** et d'**optimiser les coûts**.

L'amélioration de la performance du modèle en production permet également de **maximiser les bénéfices** et de **répondre aux besoins des clients de manière efficace**.

Pour atteindre ces objectifs, il est nécessaire de mettre en place une **surveillance continue, une analyse rigoureuse des données et une adaptation constante du modèle** en fonction des besoins et des évolutions du marché. Il est également important de **favoriser la collaboration étroite entre les différents acteurs** impliqués dans le processus de production et de maintenance du modèle.

2 Évaluation de la performance du modèle offline avec seuils de performance.

Nous avons utilisé le service de modélisation **Microsoft Azure LUIS** pour déterminer l'intention ***(Intent)*** de l'utilisateur en fonction de sa demande et identifier les informations pertinentes pour répondre à cette demande ***(Entity)***.

Afin de garantir la qualité de notre modèle, nous avons développé des fonctions spécifiques en Python pour évaluer les performances de LUIS dans le chatbot que nous avons déployé. Nous avons donc séparé le jeu de données initial en deux jeux distincts :

— un jeu pour entraîner le modèle **(train)**
— un jeu pour l'évaluation **(test)**

Nous avons retenu plusieurs critères d'évaluation pour mesurer la performance du modèle, notamment :

— **Précision de l'identification de l'intention *(Intent)*** : mesure la capacité du modèle à identifier correctement l'intention de l'utilisateur.
Résultats obtenus = un taux de détection de l'intention de 99%

— **Précision de l'identification des entités *(Entity)*** : mesure la capacité du modèle à identifier les informations importantes pour répondre à la demande de l'utilisateur.
Résultats obtenus = un taux de détection d'entités de 84%

— **Précision de l'identification de la catégorie salutations *(Greetings)*** : mesure la capacité du modèle à identifier correctement une simple salutation.
Résultats obtenus = un taux de détection d'intentions de 85%

Ces résultats témoignent de la qualité du modèle mis en place, ainsi que de sa capacité à généraliser à des données inédites.

Afin de savoir si un modèle fonctionne correctement, il est possible d'utiliser un seuil qui permet de définir une limite acceptable pour sa performance. Si la performance du modèle est en dessous de ce seuil, cela signifie que le modèle ne fonctionne pas aussi bien qu'il le devrait et qu'il est donc temps d'entraîner à nouveau le modèle.

En ré-entraînant le modèle, on peut ajuster les paramètres pour améliorer sa capacité à détecter les intentions de l'utilisateur, les entités ou les salutations.

L'utilisation d'un seuil permet également d'automatiser le processus de ré-entraînement du modèle, en lançant automatiquement une nouvelle phase de ré-entraînement si la performance du modèle est en dessous du seuil.

Pour chacun des critères d'évaluation, voici les seuils que nous pouvons définir :

— Précision de l'identification de l'intention (Intent) : **seuil de 90%**
— Précision de l'identification des entités (Entity) : **seuil de 80%**
— Précision de l'identification de la catégorie salutations (Greetings) : **seuil de 80%**

.

3 Méthodologie de pilotage de la performance en production

Pour le suivi de la performance du modèle en production, nous avons opté pour l'utilisation du service **Applications Insights** de Microsoft Azure.

La mise en place de ce service se fait en deux étapes simples.

- **Étape 1** : Création du service Applications Insights dans notre compte Azure
- **Étape 2** : Ajout des éléments nécessaires au code Python pour permettre l'envoi des informations à ce service.

Une fois le service Applications Insights mis en place, nous avons accès à plusieurs fonctionnalités pour piloter la performance de notre modèle en production, afin de faciliter l'identification et la résolution des problèmes éventuels.

Ces fonctionnalités comprennent notamment :

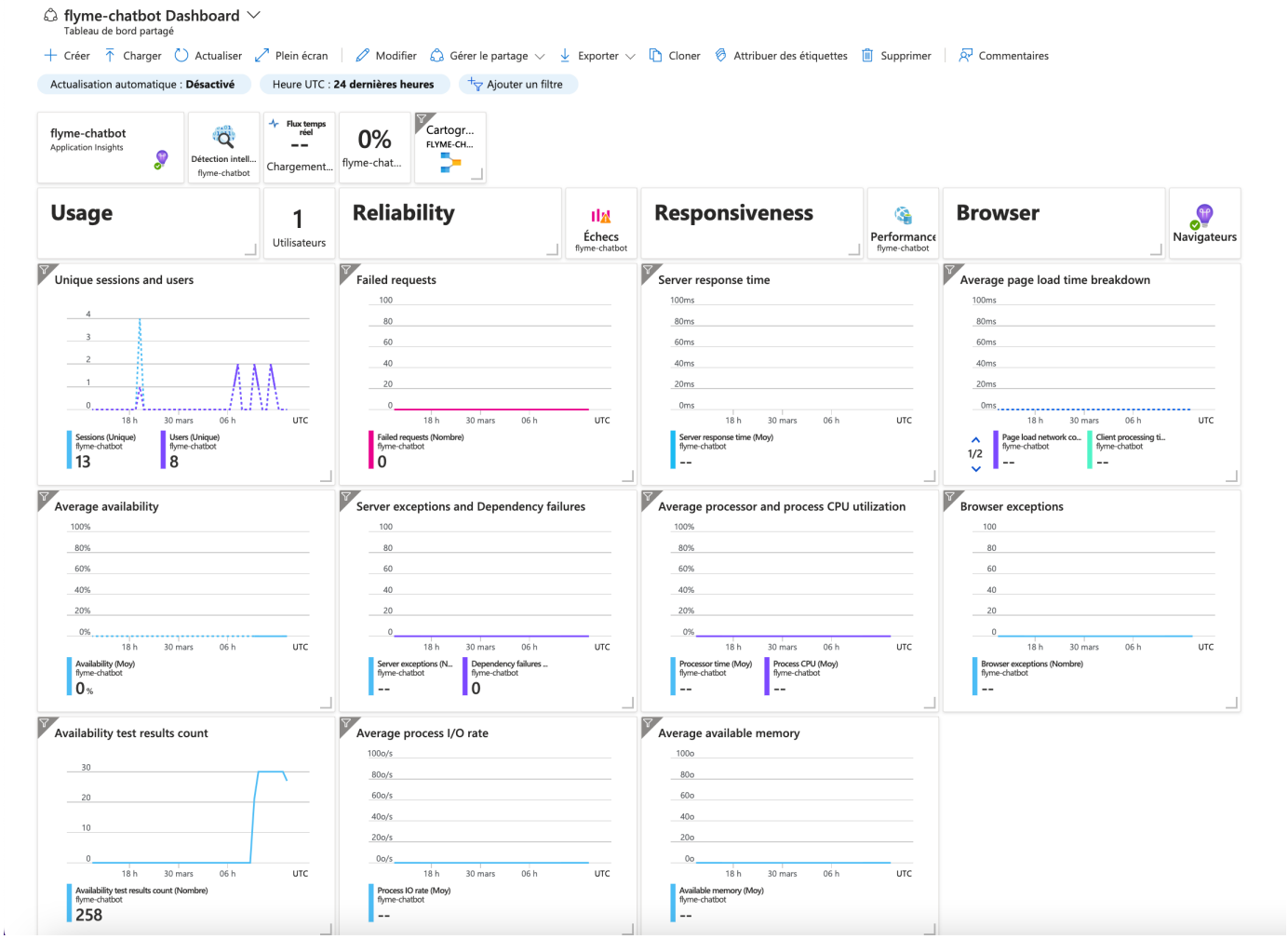
- la surveillance des métriques de performance en temps réel,
- la détection des anomalies,
- le suivi des erreurs et des exceptions,
- la collecte des traces de débogage

Applications Insights nous permet donc d'avoir une vue complète de la performance de notre modèle en production, ce qui nous permet d'apporter les ajustements nécessaires pour optimiser sa performance et améliorer l'expérience utilisateur.

3.1 Dashboard : Tableau de suivi récaputilatif

Le **dashboard** Azure Insights est un outil qui permet de suivre la performance des applications hébergées sur Azure. Il fournit des informations sur les mesures telles que la disponibilité, les performances et le nombre d'utilisateurs, ce qui permet de détecter rapidement les problèmes et d'optimiser l'application pour améliorer l'expérience utilisateur.

Ci-dessous une présentation du Dashboard Azure Insight :



3.2 Métrics : métriques de suivi de la performance

Metrics (ou métriques) sur Azure Insights est un outil de suivi des performances qui permet de collecter et de visualiser des données clés telles que :

— **Compteur de performance** : mesure les temps de réponse et de chargement de l'application, et permet d'identifier les problèmes de performance.

— **Disponibilité** : surveille la fréquence et la durée des temps d'arrêt de l'application, ce qui permet de détecter rapidement les problèmes d'indisponibilité.

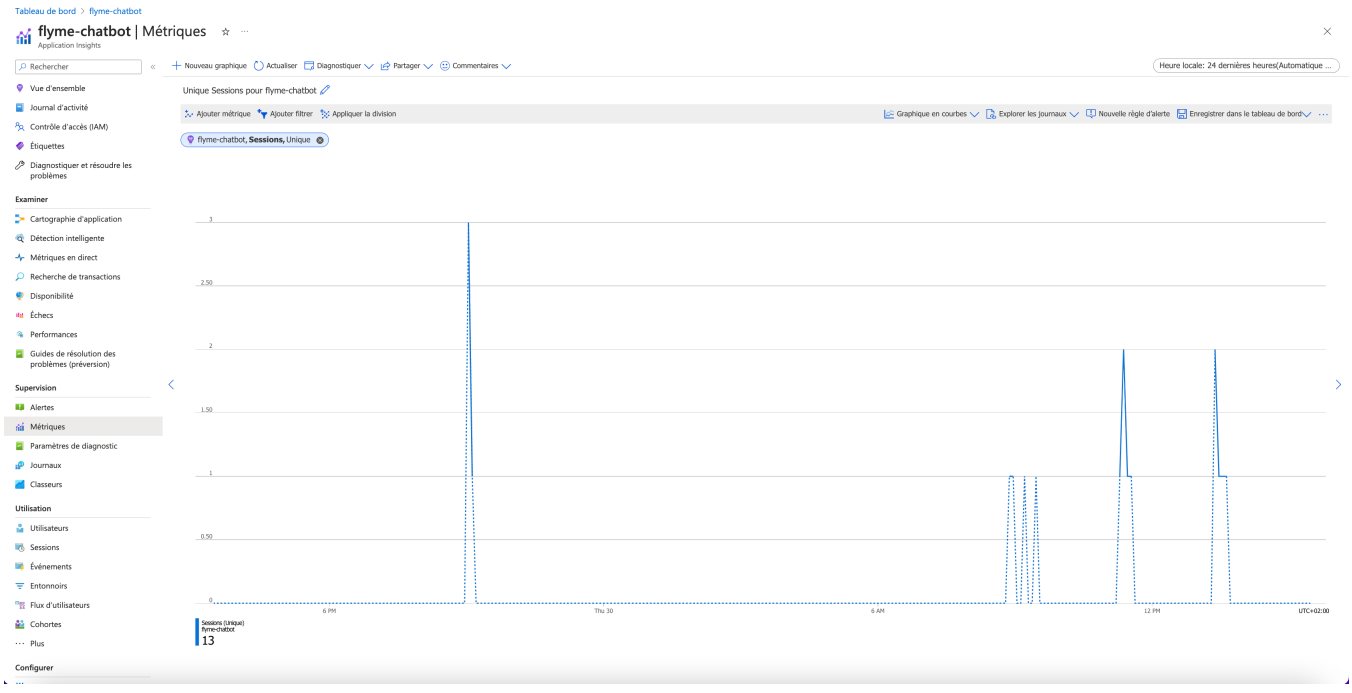
— **Navigateur** : suit les performances du navigateur des utilisateurs et peut être utilisée pour identifier les problèmes de compatibilité et optimiser l'expérience utilisateur.

— **Serveur** : suit les performances du serveur sur lequel l'application est hébergée, en surveillant les temps de réponse, les ressources utilisées et la disponibilité des ressources.

— **Utilisation** : mesure la fréquence d'utilisation de l'application, le nombre de sessions actives, le nombre d'utilisateurs et le taux d'engagement, afin de comprendre les interactions des utilisateurs avec l'application et d'améliorer son efficacité.

— **Échec** : suit les erreurs qui se produisent dans l'application, telles que les exceptions, les erreurs de base de données et les erreurs de serveur, et permet de détecter rapidement les problèmes et de prendre des mesures pour les résoudre.

Ci-dessous une présentation d'une métrique de suivi de la performance des sessions utilisateurs :



4 Évaluation de la performance du modèle en production

Les critères de pilotage de la performance du modèle en production sont essentiels pour garantir la qualité de service et la satisfaction des utilisateurs.

Ainsi, nous avons sélectionné les critères suivants pour évaluer en continu la performance du chatbot :

— **Le nombre d'utilisateurs** de l'application est un indicateur important pour savoir combien de personnes utilisent notre chatbot. Il nous permet de mesurer la popularité de notre chatbot, et d'identifier les éventuels pics de trafic qui pourraient impacter la performance du modèle.

— **Le nombre de sessions** représente le nombre de sessions d'utilisation de l'application. Cet indicateur nous permet de comprendre le comportement des utilisateurs et d'identifier les tendances d'utilisation de l'application.

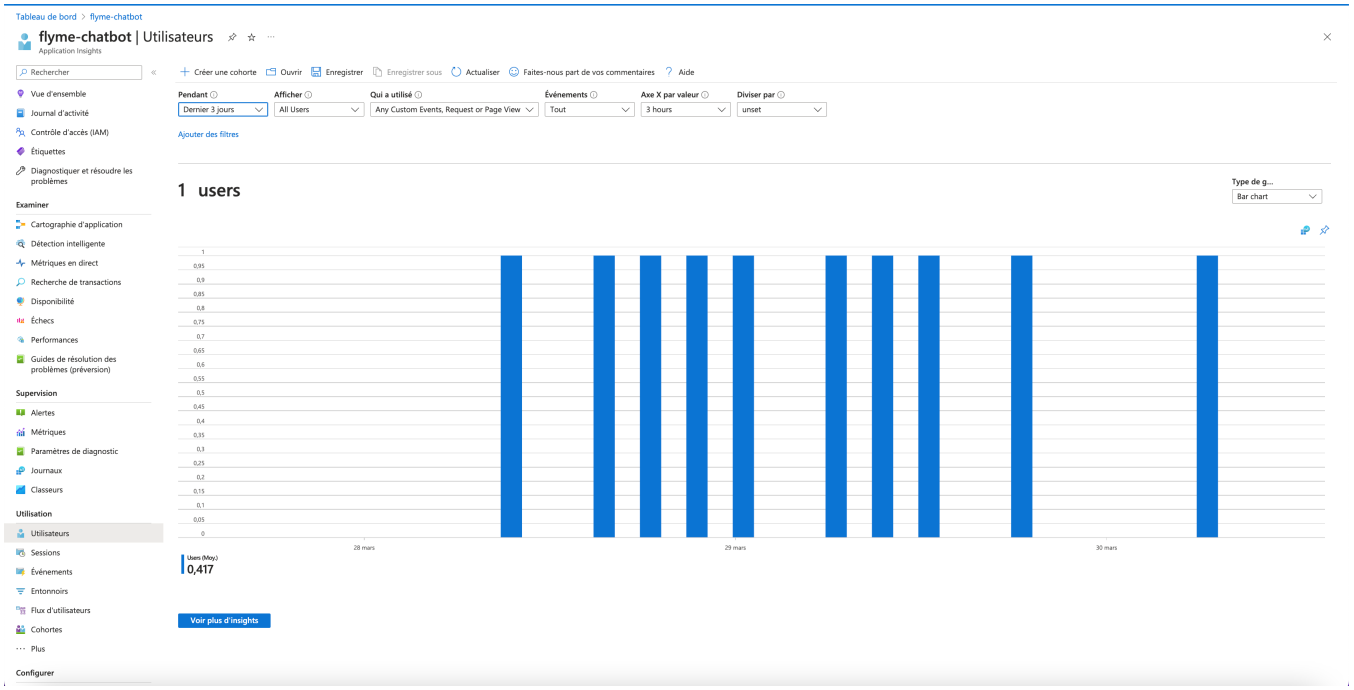
— **La disponibilité** est un critère clé pour s'assurer que notre chatbot fonctionne tout le temps. Nous avons mis en place un test pour détecter les problèmes et les résoudre rapidement afin que les utilisateurs puissent continuer à utiliser notre chatbot.

— **Le temps de réponse** moyen aux requêtes correspond à la durée que met l'application pour répondre aux requêtes des utilisateurs. Il est essentiel pour garantir la satisfaction des utilisateurs, car un temps de réponse élevé peut entraîner une expérience utilisateur médiocre. Ainsi, en surveillant cette métrique, nous pouvons nous assurer que notre application répond rapidement et efficacement aux requêtes des utilisateurs.

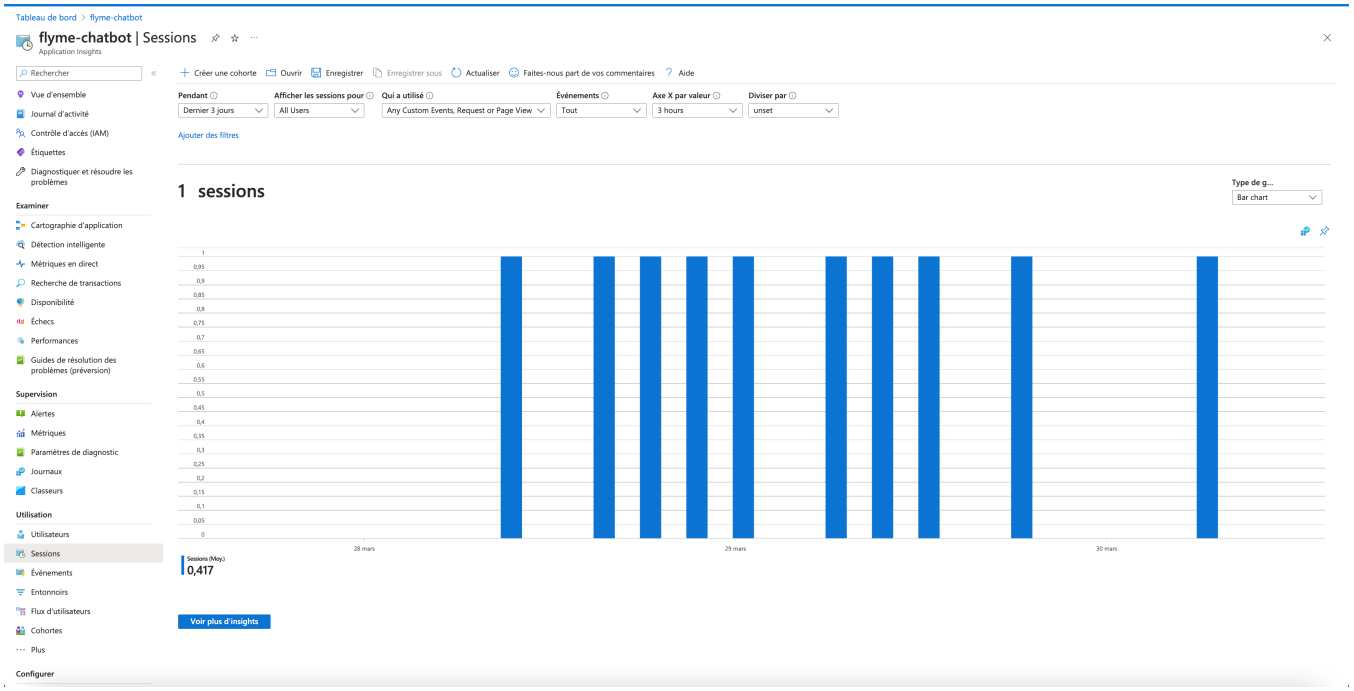
— **La dégradation de performance** est une alerte déclenchée lorsque le chatbot rencontre trois erreurs ou plus en une heure. Cela peut indiquer un dysfonctionnement du modèle et nécessiter une intervention rapide pour rétablir la performance du chatbot.

Les résultats de ces évaluations sont résumés dans les graphiques ci-dessous :

4.1 Surveillance du nombre d'utilisateurs : effectuée en temps réel par Azure.



4.2 Surveillance du nombre de sessions : effectuée en temps réel par Azure.



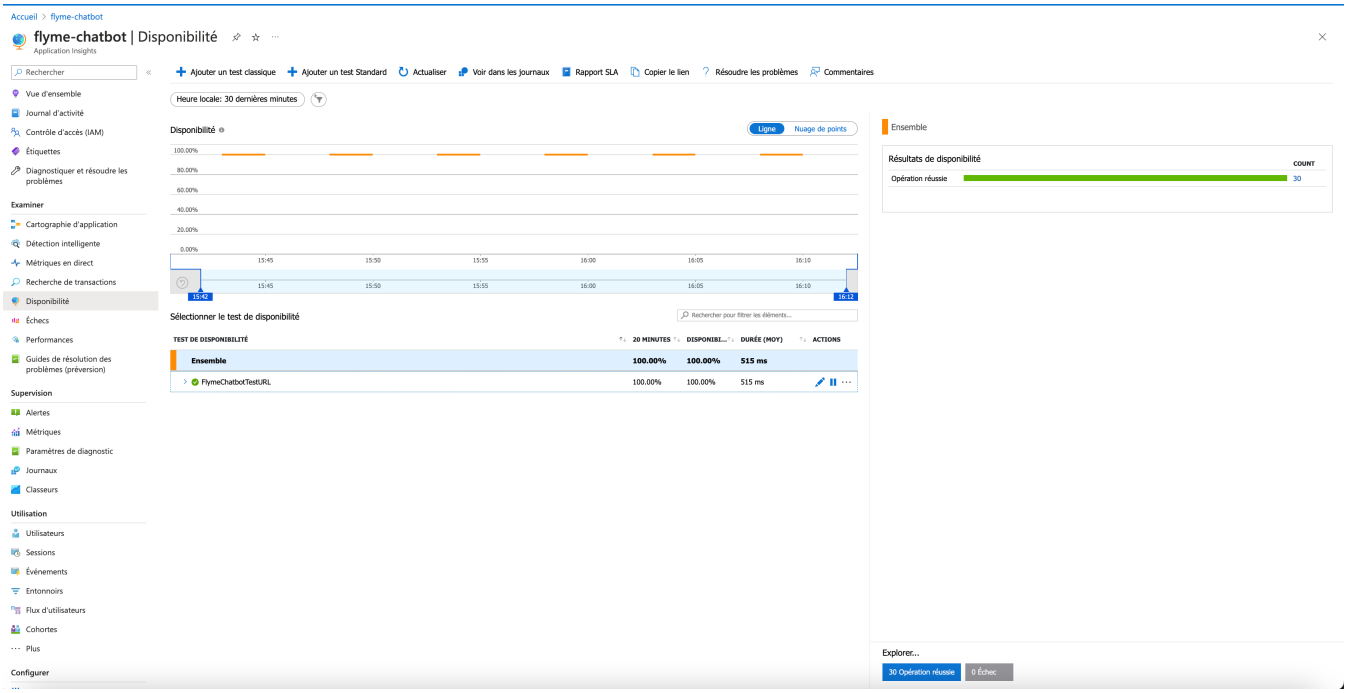
4.3 Surveillance de la disponibilité : fondée sur la réponse de l'application au ping.

Nous avons mis en place un système de surveillance pour suivre la disponibilité de l'application. Ce système utilise un test de disponibilité qui consiste à envoyer un signal appelé **ping** à l'application et à surveiller la réponse.

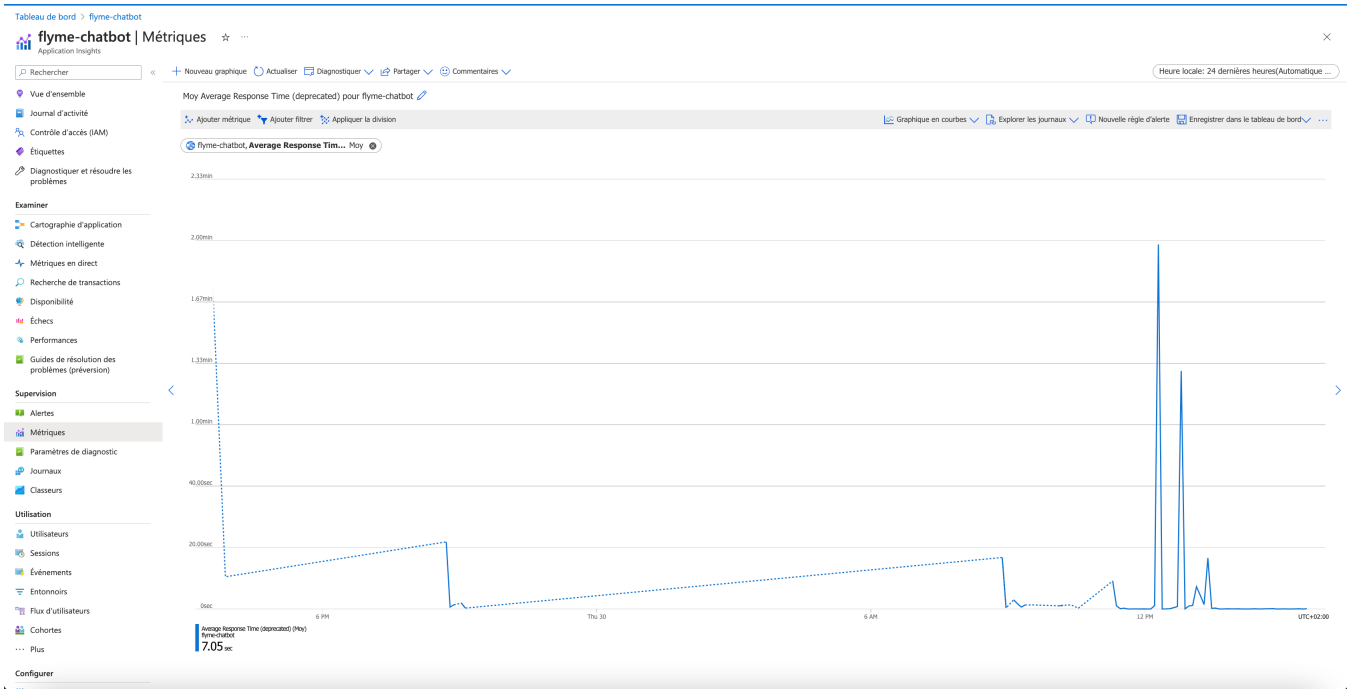
Si l'application répond correctement, cela signifie qu'elle est disponible. Si elle ne répond pas, cela indique qu'il peut y avoir un problème d'indisponibilité.

En utilisant cette méthode, nous pouvons suivre en temps réel la disponibilité de l'application et détecter rapidement tout problème qui pourrait affecter la disponibilité de l'application pour les utilisateurs finaux.

Cette surveillance nous permet de prendre des mesures rapidement pour résoudre les problèmes et minimiser les temps d'arrêt.



4.4 Le temps de réponse : la durée moyenne pour répondre aux requêtes.



4.5 Critère de dégradation de la performance

Le critère de dégradation de performance est mis en place pour détecter si le chatbot fait trois erreurs ou plus sur une période d'une heure. L'objectif est d'être alerté si le chatbot commence à avoir des difficultés à comprendre les demandes des utilisateurs et à traiter correctement les transactions.

la mise en place de ce critère de dégradation est réalisée en utilisant la méthode d'Application Insights suivante :

— **track_trace()** : pour enregistrer des événements liés à l'interaction de l'utilisateur avec le chatbot

```
if step_context.result:
        self.telemetry_client.track_trace("TransactionConfirmed : YES", booking_info, "INFO")
        self.telemetry_client.track_trace("ChatHistory", self.chat_history, "INFO")
        return await step_context.end_dialog(booking_details)
else:
        self.telemetry_client.track_trace("TransactionDismissed : NO", booking_info, "ERROR")
        self.telemetry_client.track_trace("ChatHistory", self.chat_history, "ERROR")
        return await step_context.end_dialog()
```

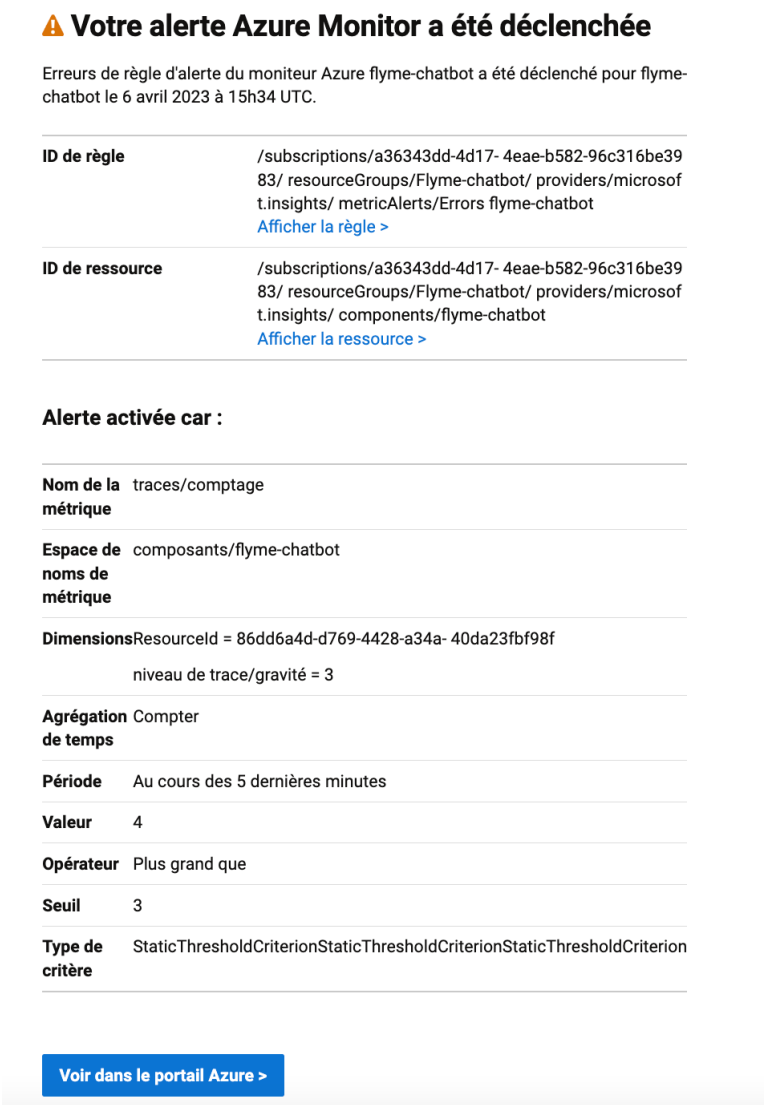
— Si l'utilisateur a **validé la transaction**, une information est envoyée à Application Insights indiquant que la transaction a été validée.

— Si l'utilisateur n'a **pas validé la transaction**, une information de type **ERROR** est envoyée à Application Insights indiquant que la transaction n'a pas été validée. Les échanges entre l'utilisateur et le chatbot sont également enregistrés.

Dans Azure, une règle est mise en place pour calculer le nombre de fois où le chatbot a fait une erreur (c'est-à-dire que l'utilisateur n'a pas validé la transaction). Si le nombre d'erreurs est supérieur ou égal à trois sur une période de cinq minutes, un message d'alerte est envoyé par email décrivant l'erreur.

Cette règle permet d'observer les interactions des utilisateurs avec le chatbot afin de détecter rapidement d'éventuels problèmes de performance et de prendre les mesures nécessaires pour y remédier.

Ci-dessous un exemple de réception d'un mail d'alerte



⚠ Votre alerte Azure Monitor a été déclenchée

Erreurs de règle d'alerte du moniteur Azure flyme-chatbot a été déclenché pour flyme-chatbot le 6 avril 2023 à 15h34 UTC.

ID de règle	/subscriptions/a36343dd-4d17- 4eae-b582-96c316be39 83/ resourceGroups/Flyme-chatbot/ providers/microsof t.insights/ metricAlerts/Errors flyme-chatbot Afficher la règle >
ID de ressource	/subscriptions/a36343dd-4d17- 4eae-b582-96c316be39 83/ resourceGroups/Flyme-chatbot/ providers/microsof t.insights/ components/flyme-chatbot Afficher la ressource >

Alerte activée car :

Nom de la métrique	traces/comptage
Espace de noms de métrique	composants/flyme-chatbot
Dimensions	ResourceId = 86dd6a4d-d769-4428-a34a- 40da23fbf98f niveau de trace/gravité = 3
Agrégation de temps	Compter
Période	Au cours des 5 dernières minutes
Valeur	4
Opérateur	Plus grand que
Seuil	3
Type de critère	StaticThresholdCriterionStaticThresholdCriterionStaticThresholdCriterion

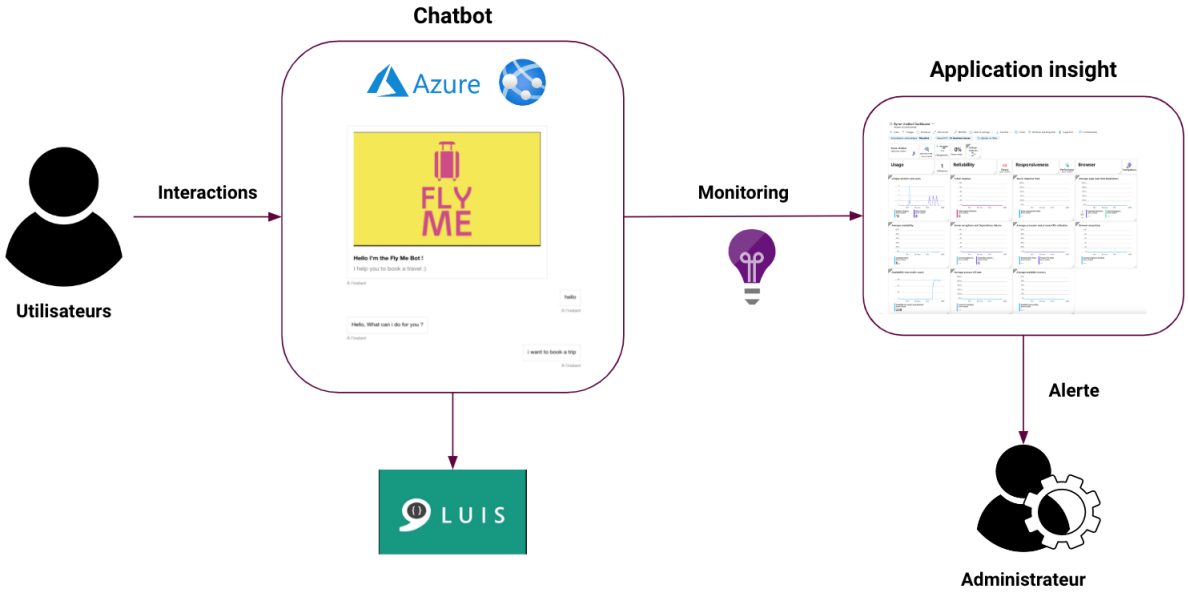
Voir dans le portail Azure >

5 Schéma du mécanisme d'évaluation du modèle en production

Le schéma montre le chatbot en production qui reçoit les messages entrants de l'utilisateur. Les messages sont traités par un modèle de traitement du langage naturel LUIS qui est responsable de déterminer l'intention de l'utilisateur et les entités associées.

Le chatbot envoie ensuite une requête à Azure Insight pour collecter des données sur la performance du chatbot pendant l'interaction.

Si des problèmes sont identifiés, une alerte est envoyée à l'administrateur.



6 Dérive du modèle *(Drift)*

Le drift est un phénomène qui se produit lorsqu'un modèle de chatbot devient moins précis ou pertinent dans ses réponses au fil du temps, en raison des changements dans le comportement et le langage des utilisateurs ou des modifications de l'environnement.

le drift peut survenir pour plusieurs raisons, telles que :

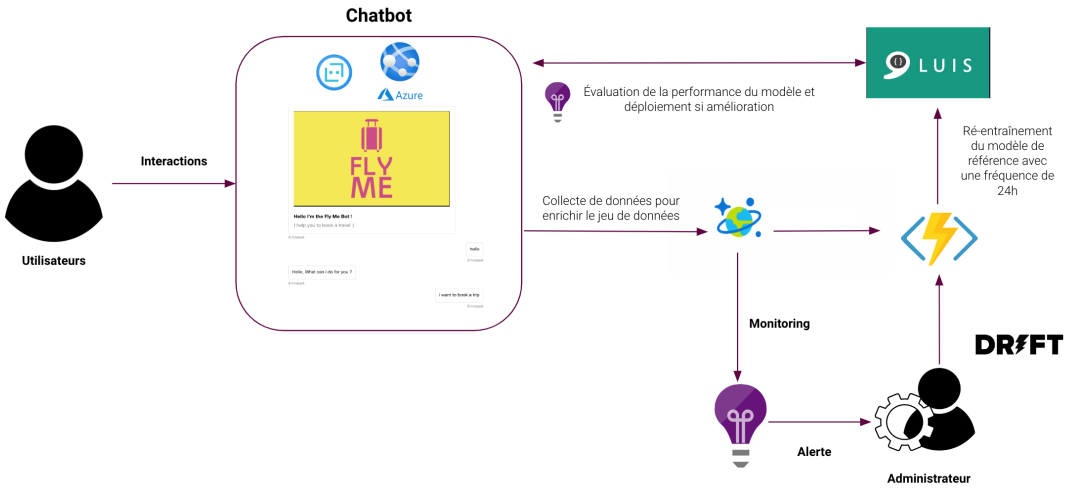
- **Les changements dans le comportement et le langage des utilisateurs**.
- **Les fluctuations dans les types de demandes et les attentes des utilisateurs**.
- **Les erreurs de compréhension ou les incohérences dans les réponses du chatbot**.
- **Les erreurs dans la collecte et le formatage des données d'entraînement**.

La détection du drift est une étape importante pour maintenir la précision et la pertinence du modèle de chatbot. Pour détecter le drift, des seuils sont définis pour surveiller les performances du chatbot. Ces seuils peuvent être basés sur des métriques telles que le taux de précision, le taux de rappel et le taux de faux positifs.

Si le chatbot commence à donner des réponses inexactes ou inappropriées au-delà des seuils définis, par exemple :

- Si **le taux d'erreurs dépasse 5 %** pendant une période de 24 heures, une alerte est déclenchée pour signaler un possible drift du modèle.
- Si **la précision des réponses du chatbot diminue de plus de 10 %** en une semaine,
- Si **le pourcentage de demandes traitées avec succès diminue de plus de 8 %** en une journée,
- Si **le pourcentage de demandes non comprises par le chatbot augmente de plus de 15 %** en une semaine,
- Si **le temps moyen de réponse du chatbot dépasse de plus de 50 %** le temps moyen de réponse habituel pendant une journée,

Une alerte est envoyée à l'administrateur du chatbot pour qu'il puisse prendre des mesures afin de corriger le drift, et mettre à jour le modèle.



7 Modalités de mise à jour du modèle

Pour assurer la mise à jour du modèle en production suite à la détection du drift, nous proposons la méthodologie suivante :

Tout d'abord, mise en place d'un système d'enregistrement des données échangées entre l'utilisateur et le chatbot en production, stocké dans une base de données. En cas d'un nombre d'erreurs important du chatbot, le modèle sous-jacent est mis à jour en suivant les étapes suivantes :

— **Récupération des données** : les échanges préalablement sauvegardés sont récupérés pour être utilisés dans le processus de mise à jour.

— **Labellisation des données** : les échanges sont labellisés pour identifier les intentions et les entités correspondantes.

— **Ré-entrainement du modèle** : le modèle LUIS est ré-entrainé avec les données labellisées à l'aide d'un script dédié.

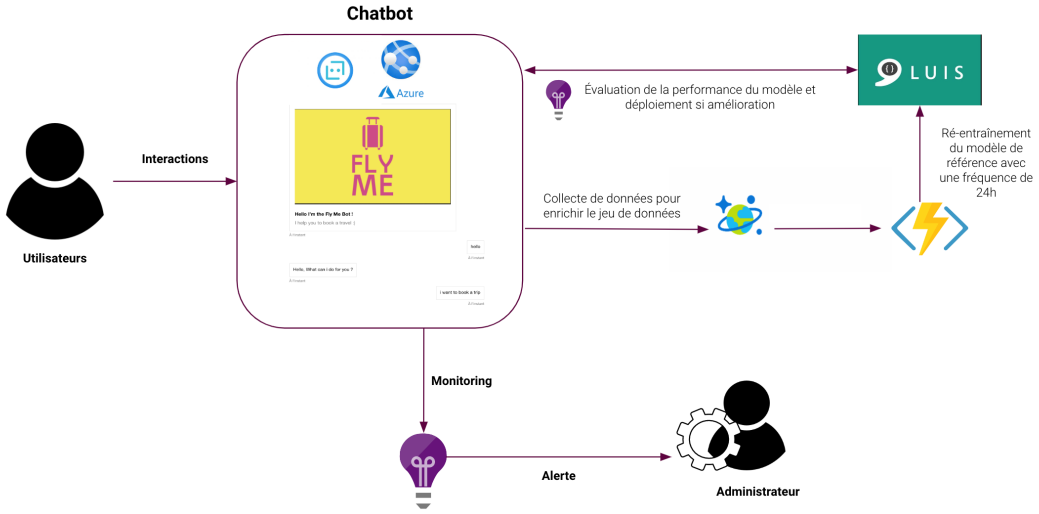
— **Déploiement du nouveau modèle** : le nouveau modèle est déployé pour être utilisé en production.

— **Observation du nouveau modèle** : si le nouveau modèle fait moins d'erreurs que le modèle de référence, il est gardé et devient le nouveau modèle de référence. Sinon, si le nouveau modèle fait plus d'erreurs, on revient au modèle précédent.

Nous recommandons de mettre en place cette méthodologie de manière continue et automatisée, afin de permettre une amélioration continue du modèle grâce à de nouvelles données et de ne remplacer le modèle que s'il est supérieur au modèle de référence.

Cela garantit une performance optimale du modèle en production et une expérience utilisateur de qualité.

En somme, la détection du drift avec des seuils définis, les alertes avec Azure Insight, et la mise à jour du modèle via Azure Function et LUIS permettent de maintenir la précision et la pertinence du modèle de chatbot à long terme.



8 Conclusion

Le projet avait pour objectif de concevoir un chatbot pour aider les utilisateurs à sélectionner une offre de voyage en utilisant des services Azure tels que le service cognitif **LUIS**, le service **Web App** et le **Bot Framework Emulator**.

Cette note souligne **l'importance de surveiller attentivement la performance et l'efficacité** du modèle de chatbot sur une longue période.

Les critères d'évaluation et le schéma du mécanisme d'évaluation proposés dans cette note permettent d'assurer **la qualité et la pertinence du modèle**.

De plus, la détection du drift et les modalités de mise à jour du modèle garantissent que celui-ci reste précis et pertinent en fonction des changements de l'environnement.

En adoptant cette approche, nous pouvons facilement **détecter les éventuels problèmes** et mettre en place des **mesures correctives** pour garantir une expérience utilisateur optimale.