Rapport Cloud Computing : Web App – Tennis players

Vue d'ensemble de l'objectif et des buts du projet

Objectif

L'objectif principal de ce projet était de développer un outil à l'aide de plusieurs services de Azure. J'ai choisi de développer une application web en utilisant Flask pour explorer les statistiques des joueurs de tennis. L'application permet aux utilisateurs de rechercher des joueurs et d'afficher leurs statistiques pertinentes, telles que le classement, le nombre total de points, le nombre de tournois joués, et leur pays. Ce projet vise à démontrer comment des applications simples peuvent être enrichies avec des technologies cloud modernes pour garantir scalabilité, fiabilité et performance.

Buts

Créer une application fonctionnelle et conviviale pour rechercher et afficher des statistiques sur les joueurs de tennis.

Intégrer plusieurs services Azure afin d'assurer un déploiement robuste et une gestion optimisée des données et fichiers statiques.

Mettre en œuvre une architecture cloud basée sur Azure qui inclut :

- **Azure App Service** pour héberger l'application.
- **Azure Storage Account** pour gérer les fichiers statiques comme les images et feuilles de style.
- **Azure Database for MySQL Flexible Server** pour stocker et gérer les données des joueurs de tennis.

Permettre un déploiement continu et automatisé via l'intégration avec GitHub.

Intégration entre les services Azure pour démontrer un cas d'utilisation

Le projet utilise trois services Azure intégrés pour réaliser une application complète et fonctionnelle :

1. **Azure App Service :**
   - Héberge l'application Flask et garantit sa disponibilité pour les utilisateurs finaux.
   - Connecté à GitHub pour des déploiements automatiques, permettant des mises à jour fluides du code.
2. **Azure Database for MySQL Flexible Server :**
   - Gère les données des joueurs de tennis.

- Fournit un backend fiable et performant pour les requêtes dynamiques sur les données, comme le classement et les statistiques.

3. **Azure Storage Account :**
    - Héberge des fichiers statiques, notamment les images de fond et les fichiers CSS.
    - Offre une URL publique pour chaque fichier, permettant une intégration facile avec le frontend de l'application.

Cas d'utilisation démontré

L'utilisateur accède à l'application via Azure App Service. Lorsqu'il recherche un joueur de tennis :

1. L'application envoie une requête à la base de données MySQL sur Azure pour récupérer les statistiques du joueur.
2. Les fichiers statiques (image de fond, styles CSS) sont chargés depuis le Storage Account pour améliorer l'expérience utilisateur.
3. Les résultats sont affichés dynamiquement dans le navigateur.

Explication des services Azure sélectionnés et leur choix

1. Azure App Service

- **Rôle :** Fournir un environnement d'hébergement pour l'application Flask.
- **Pourquoi ce choix :**
    - Facilité d'utilisation : Configuration et intégration simples avec GitHub pour des déploiements automatisés.
    - Prise en charge de Python et des frameworks modernes comme Flask.
    - Scalabilité : Permet une montée en charge automatique si le trafic augmente.

2. Azure Database for MySQL Flexible Server

- **Rôle :** Stocker et gérer les données des joueurs de tennis.
- **Pourquoi ce choix :**
    - Compatibilité : MySQL est une base de données relationnelle bien adaptée aux besoins d'une application Flask.
    - Flexibilité : Azure Flexible Server offre des options avancées comme les fenêtres de maintenance.
    - Sécurité : Intégration facile avec Azure pour des connexions sécurisées.

3. Azure Storage Account

- **Rôle :** Héberger et servir les fichiers statiques nécessaires à l'application.

- **Pourquoi ce choix :**
    - Performance : Fournit un accès rapide aux fichiers hébergés.
    - Facilité d'intégration : Chaque fichier a une URL publique accessible depuis le frontend.
    - Coût-efficacité : Une solution simple et économique pour gérer des fichiers.

Description détaillée de l'architecture

L'architecture du projet est basée sur une approche en trois couches :

1. **Frontend (Utilisateur - Navigateur Web)**
    - L'utilisateur accède à l'application via un navigateur. La page d'accueil est servie par Azure App Service, avec des fichiers statiques (CSS, images) récupérés depuis le Storage Account.
    - L'utilisateur entre le nom d'un joueur dans la barre de recherche, ce qui déclenche une requête backend.
2. **Backend (Azure App Service + Flask)**
    - Azure App Service héberge l'application Flask.
    - Flask interagit avec la base de données MySQL pour récupérer les statistiques du joueur demandé.
3. **Données et Fichiers (Azure Database & Storage Account)**
    - La base de données MySQL contient les informations des joueurs. Flask exécute des requêtes pour extraire les données pertinentes.
    - Les fichiers statiques, comme les images et feuilles de style, sont hébergés sur Azure Storage Account.

Diagramme de l'architecture (description textuelle) :

- L'utilisateur interagit avec l'application hébergée sur **Azure App Service**.
- Les requêtes de données passent par **Azure Database for MySQL**.
- Les fichiers statiques sont chargés depuis **Azure Storage Account**.

Défis rencontrés et solutions apportées

Défis

1. **Erreur Nginx lors du déploiement sur App Service :**
    - Je n'avais pas la bonne application web créée au départ. Je n'avais pas sélectionné Python lors de la création de la web app.

Résumé des enseignements

Ce qui a été appris :

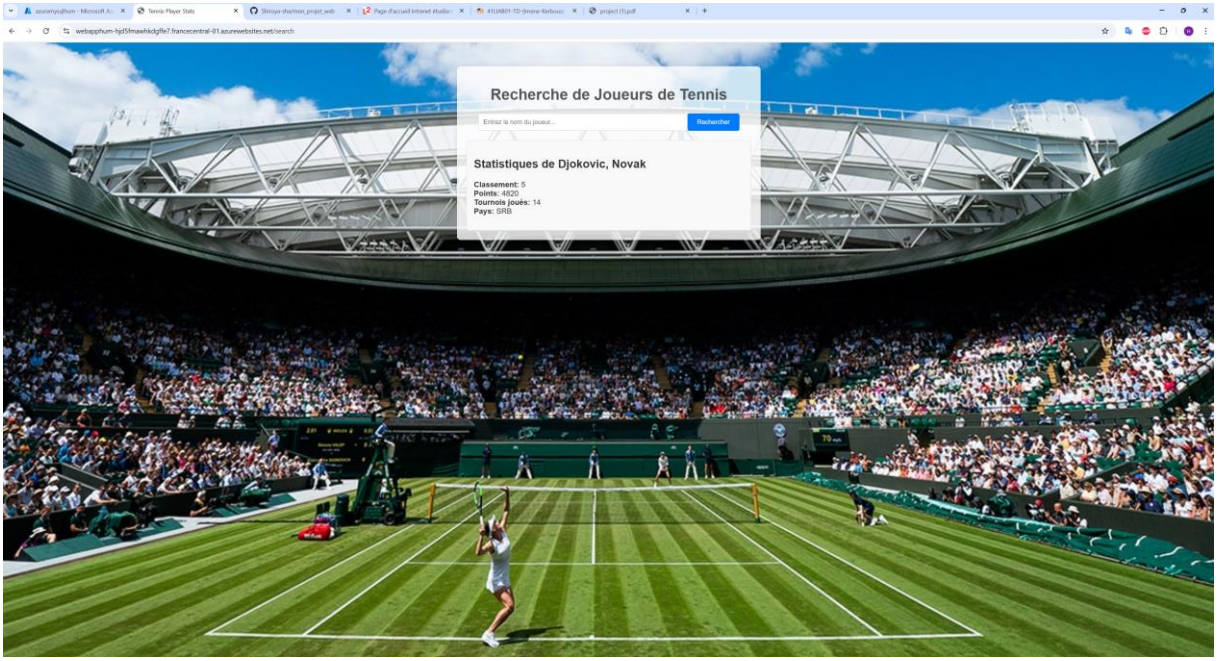
Gestion d’une architecture cloud :

Utilisation des services Azure en synergie pour héberger, stocker et gérer les données d’une application.

Résolution de problèmes techniques :

Debugging des erreurs liées aux dépendances et configurations, et amélioration de l'application en tenant compte des meilleures pratiques.

On peut voir le bon fonctionnement de l’application ci-dessous :



L’utilisateur peut chercher le nom du joueur et avoir ses statistiques grâce au dataset derrière l’application.