TP : Système de Réservation de Billets

Description détaillée :

L'objectif de ce TP est de concevoir un système de réservation de billets intégrant deux microservices distincts, chacun gérant une partie spécifique du processus. Le premier microservice se concentrera sur la gestion des vols, stockant les informations dans une base de données NoSQL pour une flexibilité accrue. Le second microservice s'occupera des réservations, utilisant une base de données relationnelle pour maintenir la cohérence des données. En outre, vous implémenterez des mécanismes de communication synchrone et asynchrone pour permettre la coordination entre les deux microservices.

Fonctionnalités attendues :

1. **Microservice de Gestion des Vols (Base de données NoSQL):**
    a. **Création de Vols** : Permettre l'ajout de nouvelles données de vol, incluant le numéro de vol, l'origine, la destination, la date et l'heure, le nombre de sièges disponibles, les informations sur l'avion, etc.
    b. **Consultation de Vols** : Autoriser la récupération de la liste complète des vols ainsi que la recherche de vols spécifiques par numéro de vol, origine ou destination.
    c. **Mise à jour de Vols** : Autoriser la modification des informations d'un vol existant.
    d. **Suppression de Vols** : Permettre la suppression d'un vol existant.
    e. **Mise à jour des Sièges** : Mettre à jour les informations sur les sièges disponibles, incluant le nom et le statut du siège.
    f. **Consultation de l'état d'un siège** : Permettre la récupération du statut d'un siège d'un vol connu par son numéro de vol
2. **Microservice de Gestion des Réservations (Base de données relationnelle) :**
    a. **Création de Réservations** : Permettre aux utilisateurs de réserver un billet en spécifiant l'identifiant de l'utilisateur, numéro de vol et numéro de siège.
    b. **Consultation de Réservations** : Autoriser la récupération de la liste complète des réservations ainsi que la recherche de réservations spécifiques par numéro de vol ou nom de l'utilisateur.
    c. **Annulation de réservation** : Permettre l'annulation d'une réservation existante. Nb: ce qui implique que la réservation a un statut. Toute annulation de réservation conduit à la libération du siège du vol réserver.
    d. **Création d'un utilisateur**: Permettre à un utilisateur de s'inscrire avec ces informations : nom, prénom, email, date de naissance. (CRUD) - L'email doit être unique pour chaque utilisateur.
    e. **Consultation d'un utilisateur**: Permettre la lecture des informations d'un utilisateur par son identifiant.

3. **Communication Synchrone (Appel API en Http):**
    a. Lorsqu'un utilisateur effectue une réservation, le microservice de réservations doit vérifier la disponibilité des sièges en interrogeant le microservice de vols de manière synchrone par http. et de vérifier l'existence de l'utilisateur.
4. **Communication Asynchrone (Utilisation RabbitMq ou Azure Service Bus) :**
    a. Lorsqu'une réservation est effectuée avec succès, un événement asynchrone doit être émis pour mettre à jour les places disponibles dans le microservice de vols et un émail est envoyé à l'utilisateur ayant fait la réservation du vol, cet email contient les informations de l'utilisateur, de sa réservation et du vol.

Exemples :

- Création de Vol :

```
Endpoint : POST /flights

Body :
{
  "id": {},
  "flightNumber": "AV123",
"origin": "Paris",
   "destination": "New York",
  "date": "2023-10-27T21:09:08.481Z",
  "sieges": [
      {"name": "A1", "status": "Disponible"},
       {"name": "A2", "status": "Disponible"}
  ],
  "informationAeroplane": {
    "modele": "Boeing 737",
    "compagny": "AirFrance"
  }
}
```

- Consultation de Vols :

```
   Endpoint : GET /flights
```

- Création de Réservation :

```
 Endpoint : POST /reservations

Body :
{
   "UtilisateurId": 1,
   "NumeroVol": "AV123",
   "NumeroSiege": "A1",
```

```
   "Statut": 0,
 "changement": "2023-10-27T21:09:08.481Z",
}
```

- Consultation de Réservations :

```
   Endpoint : GET /reservations?numero_vol=AV123
```

Étapes à suivre :

1. **Conception de l'architecture** : Définissez les endpoints, les méthodes HTTP et la gestion des erreurs pour chaque microservice.
2. **Mise en place des Microservices** : Créez deux microservices distincts en utilisant des technologies appropriées (.Net , Node Js avec Express, etc).
3. **Base de Données** :
    a. Pour le microservice de vols, utilisez une base de données NoSQL comme LiteDB ou MongoDB pour stocker les informations de vol.
    b. Pour le microservice de réservations, utilisez une base de données relationnelle comme SQLite, MySQL ou PostgreSQL pour maintenir la cohérence des données.
4. **Implémentation des Endpoints** :Programmez les endpoints pour effectuer les opérations CRUD sur les vols et les réservations.
5. **Communication Synchrone** : Assurez-vous que le microservice de réservations interagit de manière synchrone par un appel HTTP avec le microservice de vols lors de la création de réservations.
6. **Communication Asynchrone** : Implémentez un mécanisme d'émission d'événements asynchrones pour mettre à jour les places disponibles après une réservation réussie en utilisant RabbitMq ou Azure Service Bus ou d’autre système de gestion d’événements.
7. **Documentation** : Créez une documentation complète expliquant comment utiliser les API, en détaillant les endpoints et en fournissant des exemples de requêtes. Le plus important est comment on installe votre environnement et comment on peut l’utiliser pour le tester facilement.

Critères d'Évaluation :

- Conception de l'architecture (15%)
- Mise en place des microservices (15%)
- Gestion de la base de données (15%)
- Implémentation des endpoints (20%)
- Communication synchrone (15%)
- Communication asynchrone (15%)

- Documentation (10%)

Ce TP vise à approfondir votre compréhension en matière d'architecture de microservices en mettant en œuvre une communication synchrone et asynchrone entre deux API gérant des bases de données de types différents. Ce TP a des ouvertures vous permettant d’élargir les structures des données, d’apporter des idées et de jouer avec les fonctionnalités.