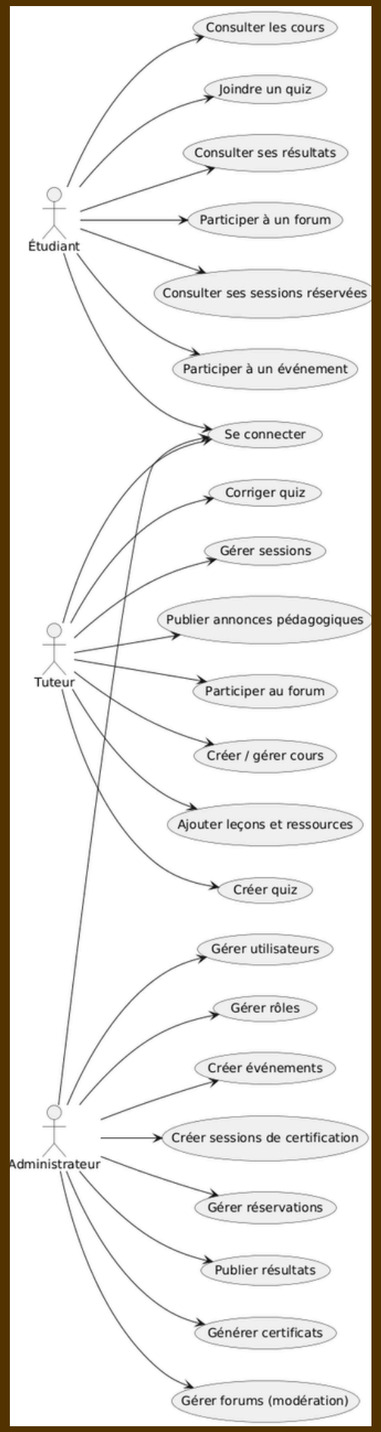
PLATEFORME DE GESTION D’UNE ÉCOLE DE LANGUE ANGLAISE

Objectif du projet:

Le projet consiste à concevoir et développer une plateforme web de gestion d’une école de langue anglaise, destinée aux étudiants.
La plateforme permet la gestion des cours, événements, forums, quiz, ainsi que des sessions et réservations, en s’appuyant sur une architecture microservices afin d’assurer la scalabilité, la maintenabilité et la séparation des responsabilités.


Consulter les cours
Joindre un quiz
Consulter ses résultats
Participer à un forum
Consulter ses sessions réservées
Participer à un événement
Étudiant
Se connecter
Corriger quiz
Gérer sessions
Publier annonces pédagogiques
Tuteur
Participer au forum
Créer / gérer cours
Ajouter leçons et ressources
Créer quiz
Gérer utilisateurs
Gérer rôles
Créer événements
Créer sessions de certification
Administrateur
Gérer réservations
Publier résultats
Générer certificats
Gérer forums (modération)

4.Description des microservices:

4. Description des microservices (complet)

Nom du microservice	Responsabilité	Fonctionnalités principales	Technologie utilisée
User Service	Gestion des utilisateurs et sécurité	• Inscription / connexion • Gestion des profils • Gestion des rôles	• Spring Boot • Spring Security / JWT • MySQL / PostgreSQL
Course Service	Gestion des cours	• Création et gestion des cours • Modules et leçons • Affectation aux utilisateurs	• Spring Boot • Spring Security / JWT • MySQL / PostgreSQL
Event Service	Gestion des événements	• Planification des événements • Gestion des participants • Calendrier	• Spring Boot • REST API • MySQL / PostgreSQL
Forum Service	Gestion des discussions	• Sujets et messages • Commentaires • Modération	• Spring Boot • REST API • Base relationnelle
Quiz Service	Évaluation des utilisateurs	• Création de quiz • Soumission des réponses • Calcul des scores	• Spring Boot • REST API • Message Broker (optionnel)
Session & Reservation Service	Gestion des sessions et réservations	• Création des sessions • Réservations / annulations • Gestion des disponibilites	• Spring Boot • REST API • Base relationnelle

5. Communication & flux:

5 Communication & Flux

Types de communication

Communication synchrone (REST)

- Protocole : HTTP / HTTPS
- Format : JSON
- Utilisée pour les appels directs entre le **Frontend** et les **microservices** via **l'API Gateway**

Exemple :

- Session & Reservation Service → **User Service** (vérification de l'utilisateur et de son rôle)

Communication asynchrone (optionnelle)

- Via un **Message Broker** : Kafka ou RabitMQ
- **Utilisation** : Pour le traitement des èvènements et des [illegible] bloquantes

Exemple :

- Quiz Service → **Notifications / Certifications** (publication des résultats, génération de certificats)

Flux principal des requêtes

1. **L'utilisateur envoie une requête** depuis le **client frontend** (Angular)
2. **L'API Gateway** intercepte la requête et applique :
   - **Authentification** (JWT)
   - **Autorisation par rôles** (*Étudiant, Tuteur,* Administrateur)
   - **Routage**
   - **Rate Limiting**
3. **Le microservice** concerné traite la requête
4. La **réponse** est retournée au **client frontend** via l'API Gateway

Infrastructure & déploiement:

🐳 Conteneurisation

- Chaque microservice est déployé dans un conteneur Docker
- Déploiement indépendant
- Isolation des pannes

🗄️ Bases de données

Une base de données par microservice :

- **UserDB**
- **CourseDB**
- **EventDB**
- **ForumDB**
- **QuizDB**
- **SessionDB**

🌐 Communication avec le Frontend :

- Le frontend communique uniquement avec l’API Gateway
- Les microservices ne sont jamais exposés directement

Conclusion:

Notre architecture microservices découpe l'application en services autonomes, communicant via REST et un broker asynchrone. L’API Gateway sécurise l’accès avec JWT. Cette structure garantit flexibilité, évolutivité et maintenance simplifiée.

THANK YOU