Projet Interopérabilité & Innovation

M2 MIAGE

Groupe 9

Equipe : CHADHOULI Nizar - AMILHAU Cédric - AMAREH Ali Houssein - ET-TAZY Mohammed - ERGUN Sinan

---

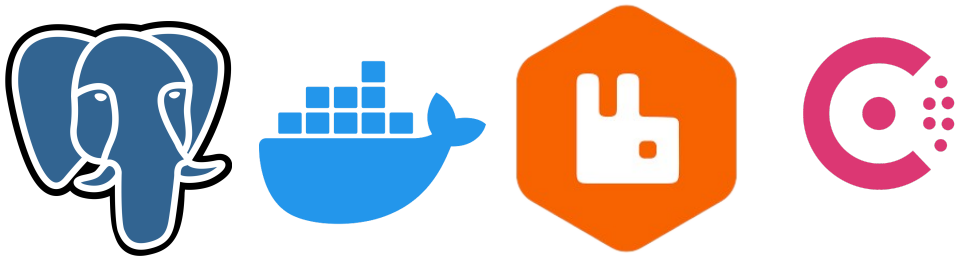


Table des matières :

Projet Bet-Event

L'objectif du projet de l'application de paris sportifs était de concevoir une architecture de microservices pour une plateforme de paris en ligne, visant à améliorer la gestion des paris sur divers événements sportifs et à assurer une sécurité renforcée ainsi qu'une évolutivité optimale.

Ci-dessous le lien de la vidéo de présentation de notre application : https://filesender.renater.fr/?s=download&token=93f20e59-80ff-4bf4-bc14-101d8adbd303

Lien du git : https://pdicost.univ-orleans.fr/git/users/o2197374/repos/groupe9-microservices/browse

Microservices

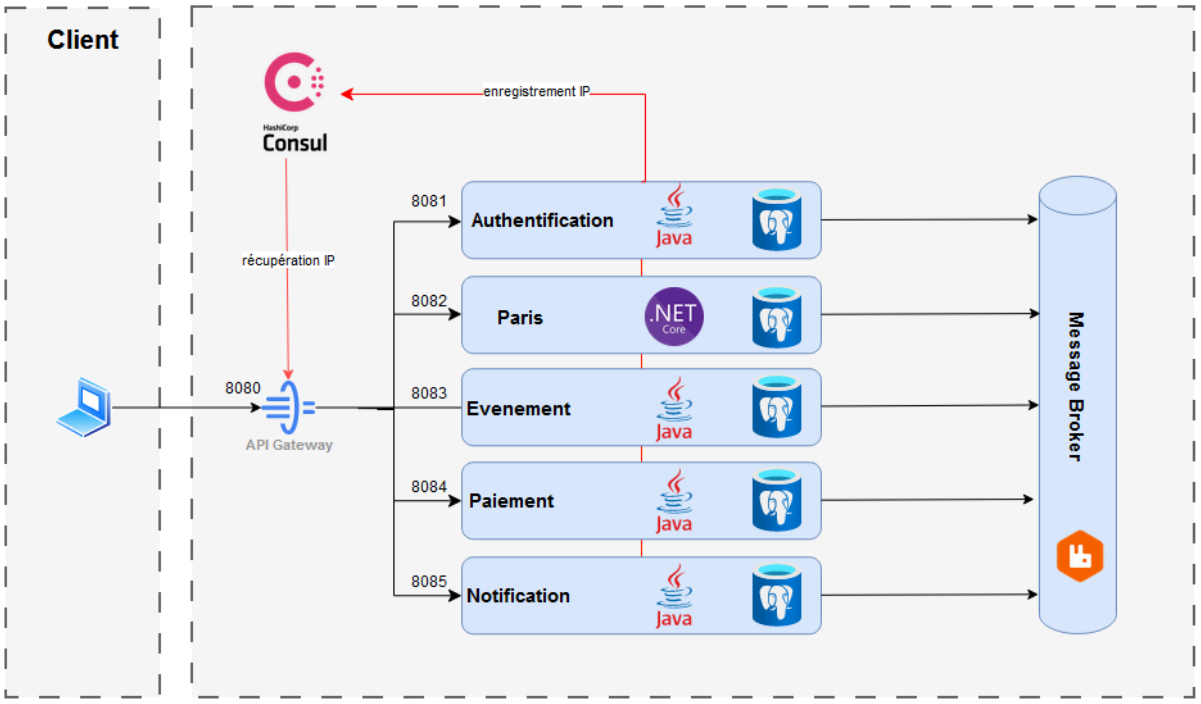
Il y a cinq microservices :

- Microservice d'Authentification et annuaire utilisateurs
- Microservice des Événements sportifs
- Microservice des Paris Sportifs
- Microservice des Paiements
- Microservice des Notifications

Travail realisé

- Authentification par token : V
- Chaque micro-service a sa propre base de données en Postgresql : V
- Le micro-service pari a été développé en .Net : V
- Les 4 autres micro-services ont été développées en Spring Boot : V
- Client en Angular : V
- API Rest : V
- Consul utilisé et fonctionnel : V
- Gateway utilisé et fonctionnel : V
- RabbitMQ utilisé et fonctionnel : V
- Application dockerisé : V

Architecture de l’application



Ci-dessus le diagramme de l’architecture de l’application avec le service pari développé en .NET et en Java pour les autres services. Chaque service a sa propre base de données PostgresSQL et communique des informations aux autres services via le bus d’événements RabbitMQ. Au lancement de chaque service ces derniers s’enregistrent auprès du Service Discovery Consul. Par la suite, la gateway récupère les informations enregistrées pour effectuer le routing vers les services attaqués par notre client Angular.

Guide utilisateur

Installation

Toutes les images étant conteneurisées et publiées sur Dockerhub il suffit juste d'exécuter la commande suivante à la racine du projet pour lancer les conteneurs :

`docker compose up -d`

PS: le service pari en .NET peut malheureusement mettre jusqu'à 5 min pour bien démarrer et ainsi empêcher l'application de bien fonctionner. Veuillez bien attendre que tous les services démarrent avant d'utiliser l'application. Merci

Lancement

Pour accéder à l'application : http://localhost:6020

Connexion :

- Admin : Login = admin@test.com | Mot de passe = admin
- User : Login = u1@test.com | Mot de passe = user1

I - Authentification : Nizar

Description :

Le service d'authentification permet de gérer les utilisateurs et leurs rôles. Il permet de s'authentifier et de récupérer un token JWT.

Fonctionnalités - Tableau des URIs :

Méthode	Uri	Réponse	Description
POST	/authentification/ inscriptions	201: Created 400: Bad Request	Inscription d'un utilisateur
POST	/authentification/ login	201: Created 400: Bad Request	Connexion d'un utilisateur
GET	/authentification/ inscriptions? page=0&size=3	200: OK 401: Unauthorized 403: Forbidden	Consultation des utilisateurs inscrits par un admin
GET	/authentification/ inscriptions	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Récupérer les infos d'un utilisateur
PUT	/authentification/ inscriptions	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Mise à jour d'un utilisateur

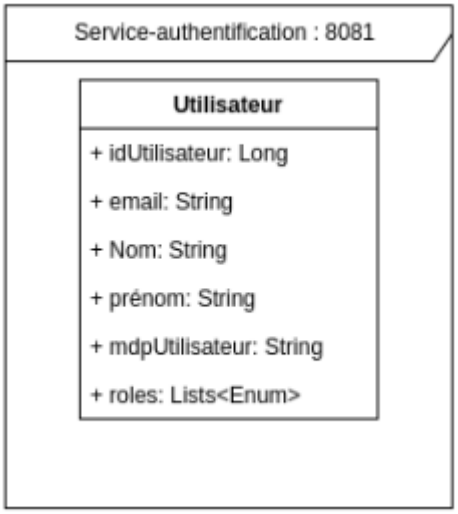
Certaines URLS n'ont pas été implémentées côté front mais sont le sont côté back via le fichier "gateway.http" du "service-authentification" :

- Consultation des utilisateurs inscrits par un admin
- Récupérer les infos d'un utilisateur
- Mise à jour d'un utilisateur

Fonctionnalités - Tableau des events RabbitMQ :

Producer	Consumer	Event	Objet	Description
service-authentification	service-notification service-paiement	utilisateur.nouveau	UtilisateurDTO(long idUtilisateur, String email, String nom, String prenom, String mdpUtilisateur)	Inscription d'un utilisateur. Le service notif génère une notification et le service paiement initialise un compte parieur

Diagramme de classe :



II - Événements sportifs : Ali

Description :

Le service d'événement sportif gère tout ce qui est lié aux événements sportifs sur lesquels les utilisateurs vont parier.

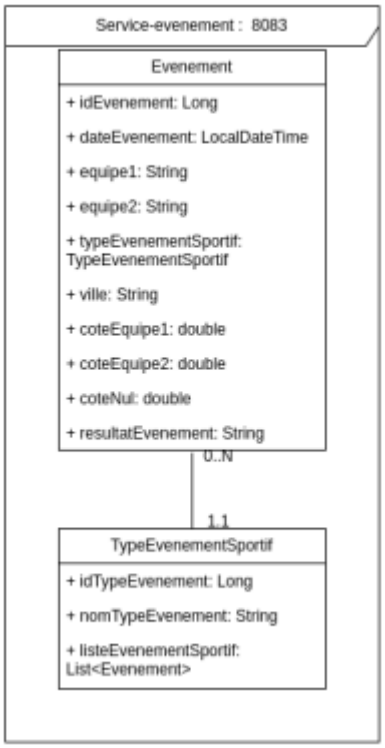
Fonctionnalités - Tableau des URIs :

Méthode	Urls	Réponse	Description
POST	/evenement	201 : Created 401: Unauthorized 403: Forbidden	Création d'un nouvel événement sportif (seulement admin)
GET	/evenement/{id}	200: OK 404 : Not found	Récupérer un événement sportif
DELETE	/evenement/{id}	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Supprimer un événement sportif (seulement admin)
POST	/evenement/id	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Modifier un événement sportif (seulement admin)
POST	/evenement/{id}/ resultat	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Modifier le résultat d'un evenement sportif (seulement admin)
GET	/evenement/even ments?page=0&si ze=3	200: OK 404: Not Found	Consultation des événements sportifs par pagination

Fonctionnalités - Tableau des events RabbitMQ :

Producer	Consumer	Event	Objet	Description
service-evenement	service-pari	evenement.nouveau	EvenementCreationDTO (Long idEvenement, LocalDateTime date)	Envoie les infos d'un événement au service pari,
service-evenement	service-pari	evenement.resultat	EvenementResultatDTO (Long idEvenement, String typeResultat, Double coteResultat)	Envoie le résultat d'un événement sportif
service-evenement	service-pari	evenement.supprime	EvenementSupprimerDTO(Long idEvenement)	Envoie l'id de l'événement sportif supprimé

Diagramme de classe :



III - Paris sportifs : Sinan

Description :

Le service paris est responsable de la gestion, du contrôle et de la validité des paris. Ce service s'appuie sur 2 tables pour effectuer cela. La table Paris s'effectue de gérer les paris et la table PariOuverts s'occupe du contrôle du contrôle et la validité d'un paris. Dans le contexte RabbitMQ ce service reçoit et envoie des évènements avec le service événement, notification et paiement.

Fonctionnalités - Tableau des URLs :

Méthode	Urls	Réponse	Description
GET	/paris	200: OK 40 : Unauthorized 403: Forbidden	Récupérer tous les paris (seulement admin)
GET	/paris/utilisateurs	200: OK 401: Unauthorized 403: Forbidden	Récupérer tous les paris d'un utilisateur
GET	/paris/{id}	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Récupérer un pari
GET	/paris/evenements	200: OK	Récupérer le nombre de paris pour les événements
DELETE	/paris/{id}	200: OK 401: Unauthorized 403: Forbidden 404: Not Found	Suppression d'un pari
POST	/paris	201: Created 401: Unauthorized 400: Bad Request 403: Forbidden	Création d'un pari

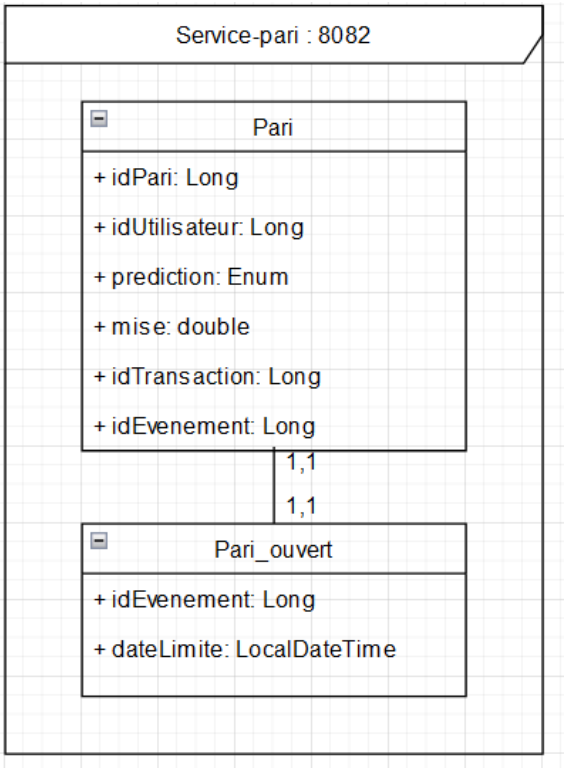
On notera qu'il n'y a pas de mise à jour d'un pari car cela ne fait aucun sens. Un utilisateur, même dans un cas réel, ne peut pas mettre à jour son pari et doit le supprimer en premier lieu pour en recréer un par la suite.
De plus, des vérifications supplémentaires ont été ajoutées sur chaque endpoint avec par exemple les erreurs 403: forbidden qui sont retournées si un utlisateur essaie d'accéder à une ressource qu'il ne possède pas (ex : accès ou suppresion d'un paris qui ne lui appartient pas). Des erreurs sont aussi retournées au niveau du POST si un utilisateur parie sur un événement déjà terminé ou avec des données incorrectes (ex : envoie d'un "string" au lieu d'un "double" pour le montant).

Fonctionnalités - Tableau des events RabbitMQ :

Producer	Consumer	Event	Objet	Description
service-pari	service-paiement service-notification	pari.annule	PaiementAnnuleDTO(double montant,long idPari, long idUtilisateur)	Remboursement d'un pari
service-pari	service-paiement service-notification	pari.gagne	PaiementDTO(double gain, long idPari, long idUtilisateur)	Envoie du montant gagné par l'utilisateur
service-evenement	service-pari	evenement.nouveau	EvenementCreationDTO (Long idEvenement, LocalDateTime date)	Création d'un PariOuvert à la création d'un événement
service-evenement	service-pari	evenement.resultat	EvenementResultatDTO (Long idEvenement, String typeResultat, Double coteResultat)	Réception du résultat pour un événement
service-evenement	service-pari	evenement.supprime	EvenementSupprimerDTO(Long idEvenement)	Supprimer des paris à la suppression d'un événement

Les événements sont assez simples dans la majorité des cas puisqu'ils consistent à une simple communication entre 2 services.
Cependant j'aimerais en détailler un qui résulte d'une suite d'events RabbitMQ qui se déclenchent.
La mise à jour, donc déclaration du vainqueur d'un événement sportif (evenement.resultat) enclenche dans le service pari un événement qui va vérifier si le pari en question est gagnant ou non puis des events vont être envoyés en fonction de cela afin de créditer le compteur vainqueur ou non. Le service notification est aussi notifié de cela.
La possibilité de parier sur un événement est dictée par l'envoie de l'événement "evenement.resultat". On peut parier sur un événement tant que l'admin ne renseigne pas de résultat pour cet événement.

Diagramme de classe :



IV - Paiements : Mohammed

Description :

Le service de paiement est responsable de gérer toutes les transactions financières liées aux paris, il permet aussi de vérifier la validité des paiements et de maintenir un suivi précis des soldes des utilisateurs.

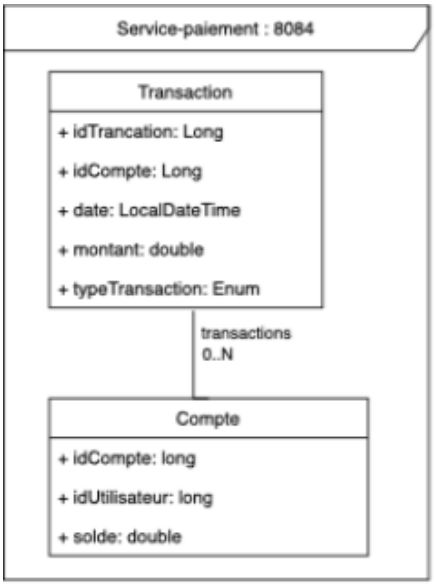
Fonctionnalités - Tableau des URIs :

Méthodes	URI	Description	Réponse
GET	/paiement/comptes/utilisateurs/{idUtilisateur}	Retourne toute les informations d'un compte à partir d'un idUtilisateur	OK: 200 401: Unauthorized 403: Forbidden 404: Not Found
POST	/paiement/reglements	Permet de payer un Pari	Created: 201 401: Unauthorized 403: Forbidden
POST	/paiement/versements	Gére les transactions entre le compte parieur et sa carte bancaire	Created: 200 401: Unauthorized 403: Forbidden

Fonctionnalités - Tableau des events RabbitMQ :

Producer	Consumer	Event	Objet	Description
service-pari	service-paiement	pari.gagne	PaiementDTO(double gain, long idPari, long idUtilisateur)	Créditer le compte d'un utilisateur d'un utilisateur un pari gagnant
service-pari	service-paiement	pari.annule	PaiementAnnuleDTO(Double montant, long idPari, long idUtilisateur)	Débiter/ Rembourser le montant misé suite à un pari annulé
service-paiement	service-notification	paiement.nouveau	TransactionDTO(long idUtilisateur, LocalDateTime date, double montant, TypeTransaction typeTransaction)	Informe le service notification à chaque fois qu'une transaction a été effectué
service-authentification	Service-paiement	utilisateur.nouveau	UtilisateurDTO(long idUtilisateur, String email, String nom,String prenom, String mdpUtilisateur)	Inscription d'un utilisateur.Le service notif génère une notification et le service paiement initialise un compte parieur

Diagramme de classe :



V - Notifications : Cedric

Description :

Le service de Notification récupère les événements envoyés par les autres services via RabbitMq pour créer des notifications afin d'informer l'utilisateur. Les notifications sont stockées dans une BDD PostgreSQL.

Fonctionnalités - Tableau des URIs :

VERBE	URL	Code de retour	Description
POST	/notifications	201 Created 401 Unauthorised 403 Forbidden 400 Bad Request	Création d'une Notification.
GET	/notifications/{idNotification}	200 OK 401 Unauthorised 403 Forbidden 404 Not Found	Récupération des informations spécifiques d'une Notification
GET	/notifications/utilisateur/{idUtilisateur}	200 OK 401 Unauthorised 403 Forbidden 404 Not Found	Récupère la liste des Notification d'un utilisateur

Fonctionnalités - Tableau des events RabbitMQ :

Producer	Consumer	Event	Objet	Description
service-authentification	service-notification service-paiement	utilisateur.nouveau	UtilisateurDTO(long idUtilisateur, String email, String nom, String prenom, String mdpUtilisateur)	Inscription d'un utilisateur. Le service notif génère une notification et le service paiement initialise un compte parieur
service-paiement	service-notification	paiement.nouveau	TransactionDTO(long idUtilisateur, LocalDateTime date, double montant, TypeTransaction typeTransaction)	Informe le service notification à chaque fois qu'une transaction a été effectué

Diagramme de classe :