SYSTÈME DE LOTERIE DISTRIBUÉ

Architecture Microservices

Projet de rattrapage

M2 MIAGE ACSI - IDMC 2025

Abdelkarim Lekdioui

1. Introduction

Ce projet implémente un système de loterie backend basé sur une architecture microservices.

L'objectif était de créer une solution distribuée, scalable et résiliente capable de gérer des tirages de loterie, des paris et l'authentification des utilisateurs.

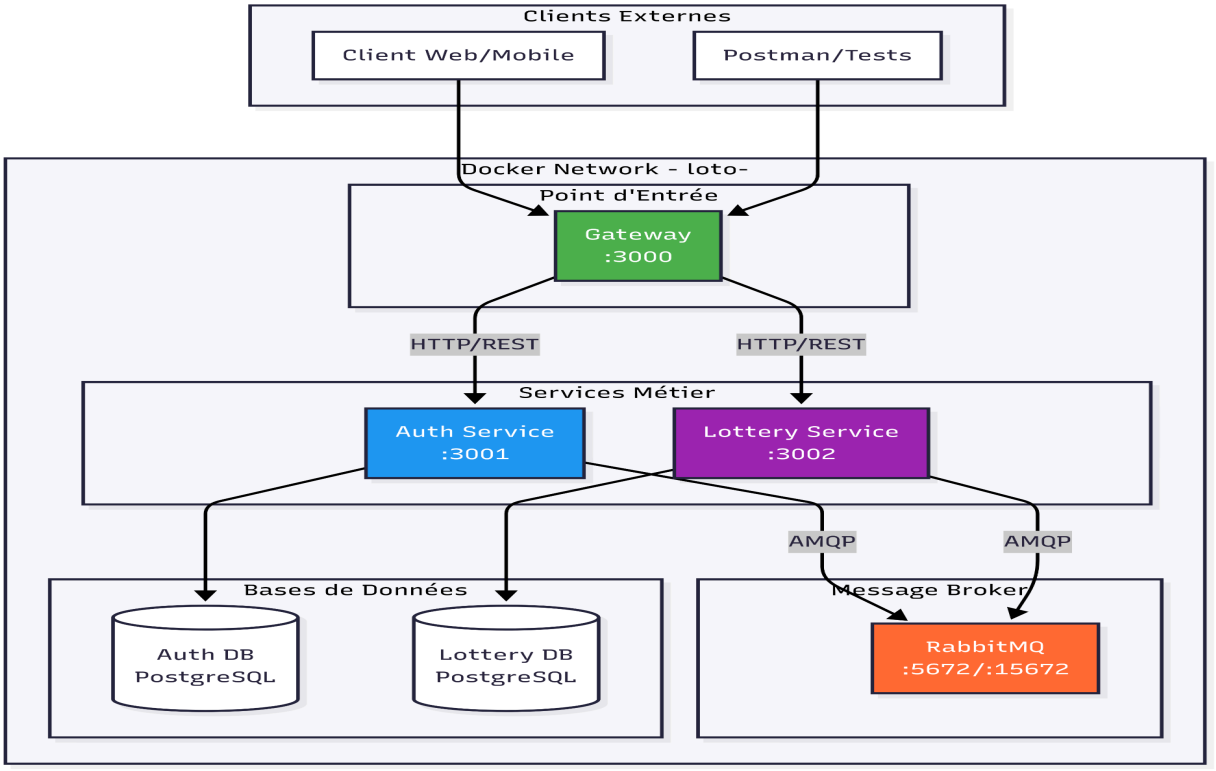
Technologies utilisées :

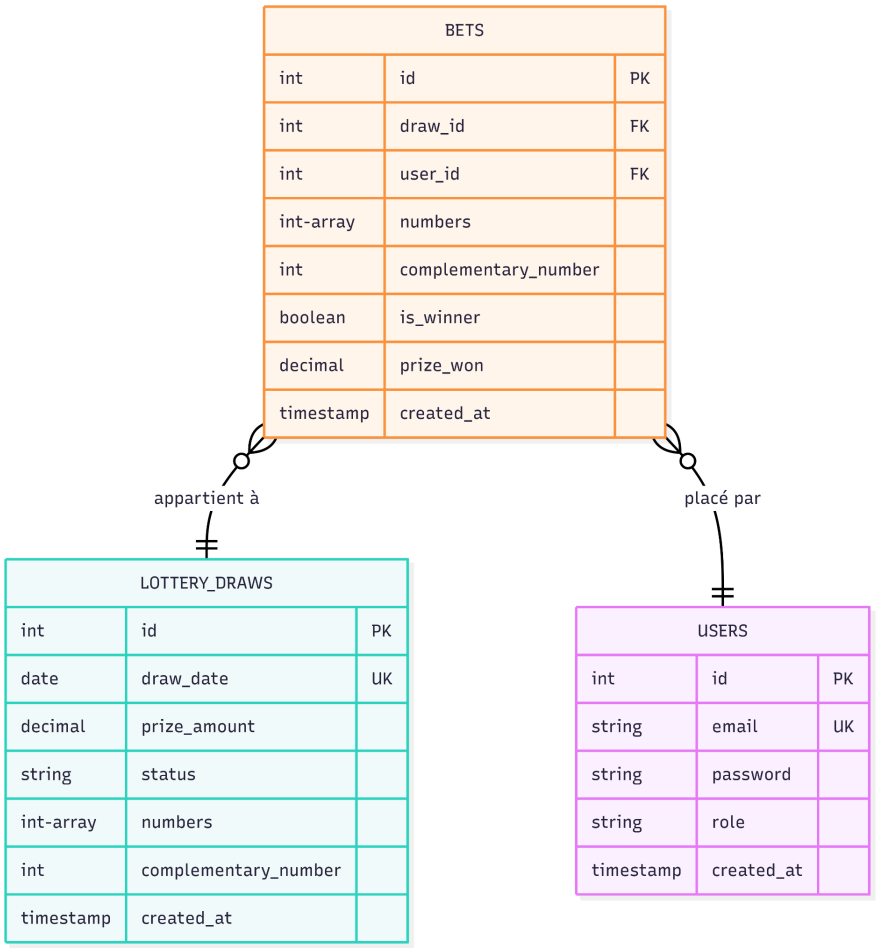
- Node.js / Express.js
- PostgreSQL
- RabbitMQ
- Docker / Docker Compose
- JWT pour l'authentification

2. Choix d'Architecture

2.1 Architecture Microservices

- Séparation des responsabilités : chaque service a un périmètre fonctionnel défini
- Services autonomes avec leurs propres bases de données
- Communication asynchrone via RabbitMQ
- Couplage faible entre les services



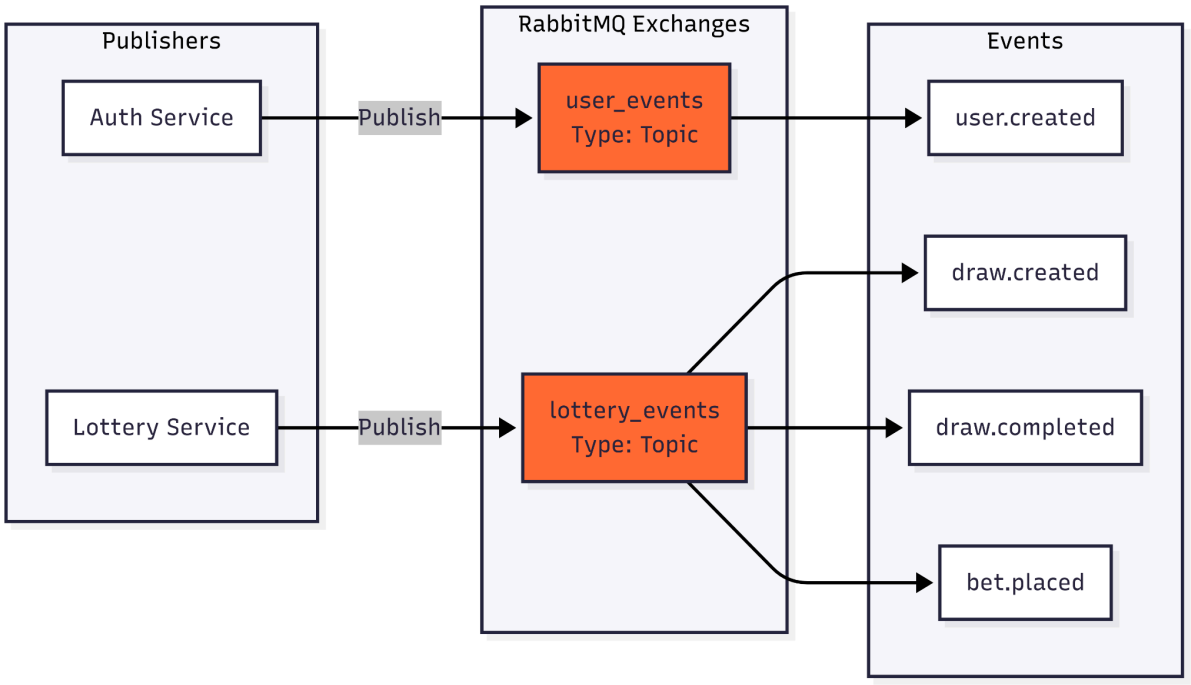


2.2 Pattern Gateway

- Point d'entrée unique pour les clients externes
- Centralisation de l'authentification JWT
- Routage intelligent vers les services internes
- Protection du réseau interne Docker

2.3 Message Broker (RabbitMQ)

- Communication événementielle asynchrone
- Résilience : les services peuvent fonctionner même si d'autres sont temporairement indisponibles
- Exchanges de type 'topic' pour la flexibilité



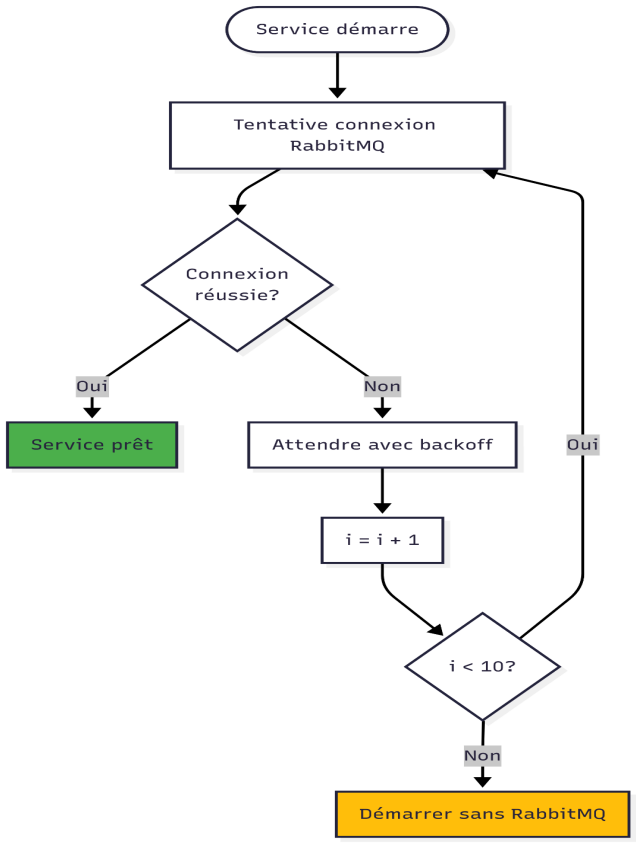
3. Difficultés et Solutions

Difficulté 1 : Synchronisation au démarrage

- Problème : Les services tentaient de se connecter à RabbitMQ avant qu'il soit prêt
- Solution : Implémentation d'une logique de retry avec backoff exponentiel
- Résultat : Démarrage fiable à 100%

Difficulté 2 : Gestion des dépendances

- Problème : Ordre de démarrage des services critique
- Solution : Health checks et gestion des dépendances dans docker-compose
- Résultat : Orchestration robuste des services



4. Fonctionnalités Implémentées

Rôle Administrateur :

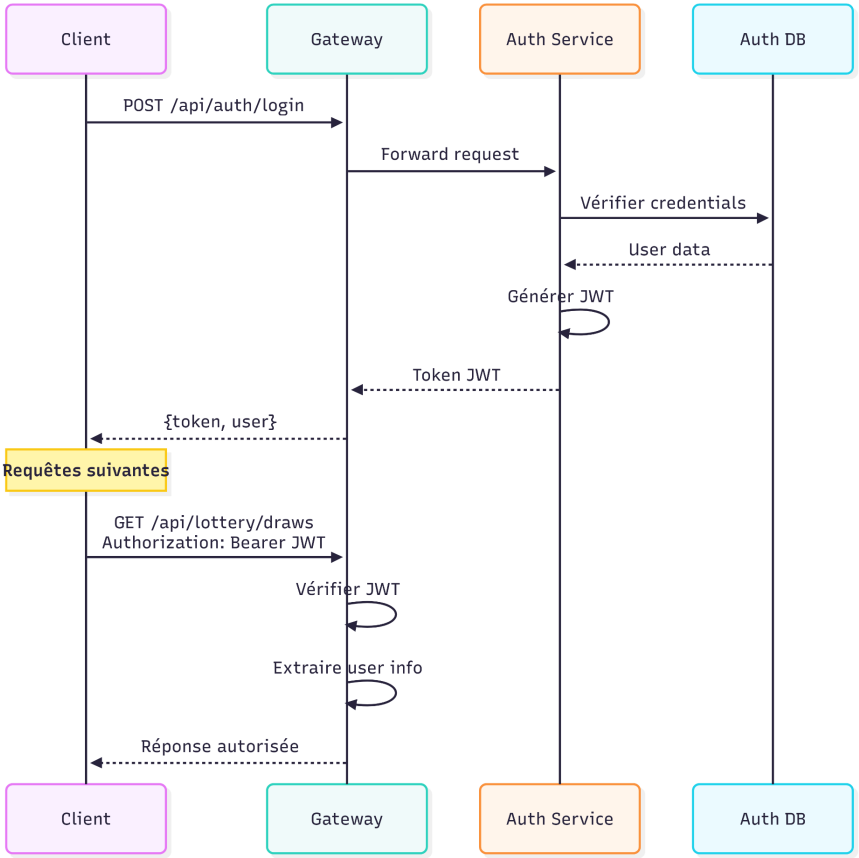
- Création et planification des tirages
- Gestion des statuts (ouvert/fermé/terminé)
- Génération aléatoire des résultats
- Détection automatique des gagnants

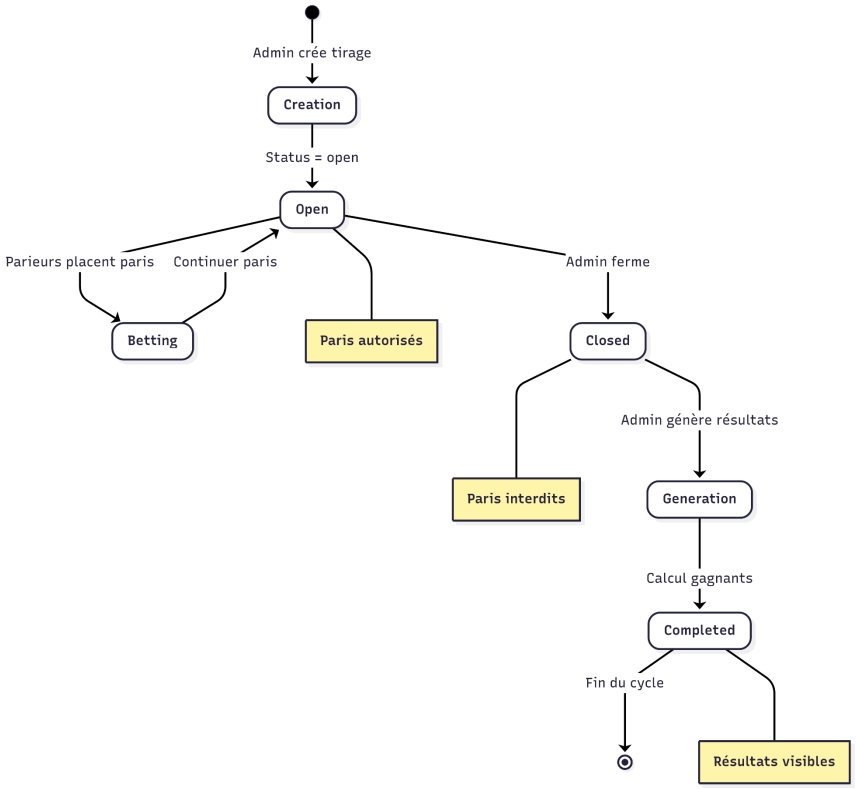
Rôle Parieur :

- Inscription et connexion
- Placement de paris (1 par tirage maximum)
- Consultation de l'historique des paris
- Visualisation des gains

Sécurité :

- Authentification JWT
- Validation des rôles
- Protection des routes privées


Client
Gateway
Auth Service
Auth DB
POST /api/auth/login
Forward request
Vérifier credentials
User data
Générer JWT
Token JWT
{token, user}
Requêtes suivantes
GET /api/lottery/draws
Authorization: Bearer JWT
Vérifier JWT
Extraire user info
Réponse autorisée
Client
Gateway
Auth Service
Auth DB


Admin crée tirage
Creation
Status = open
Open
Parieurs placent paris
Continuer paris
Admin ferme
Betting
Paris autorisés
Closed
Admin génère résultats
Paris interdits
Generation
Calcul gagnants
Completed
Fin du cycle
Résultats visibles

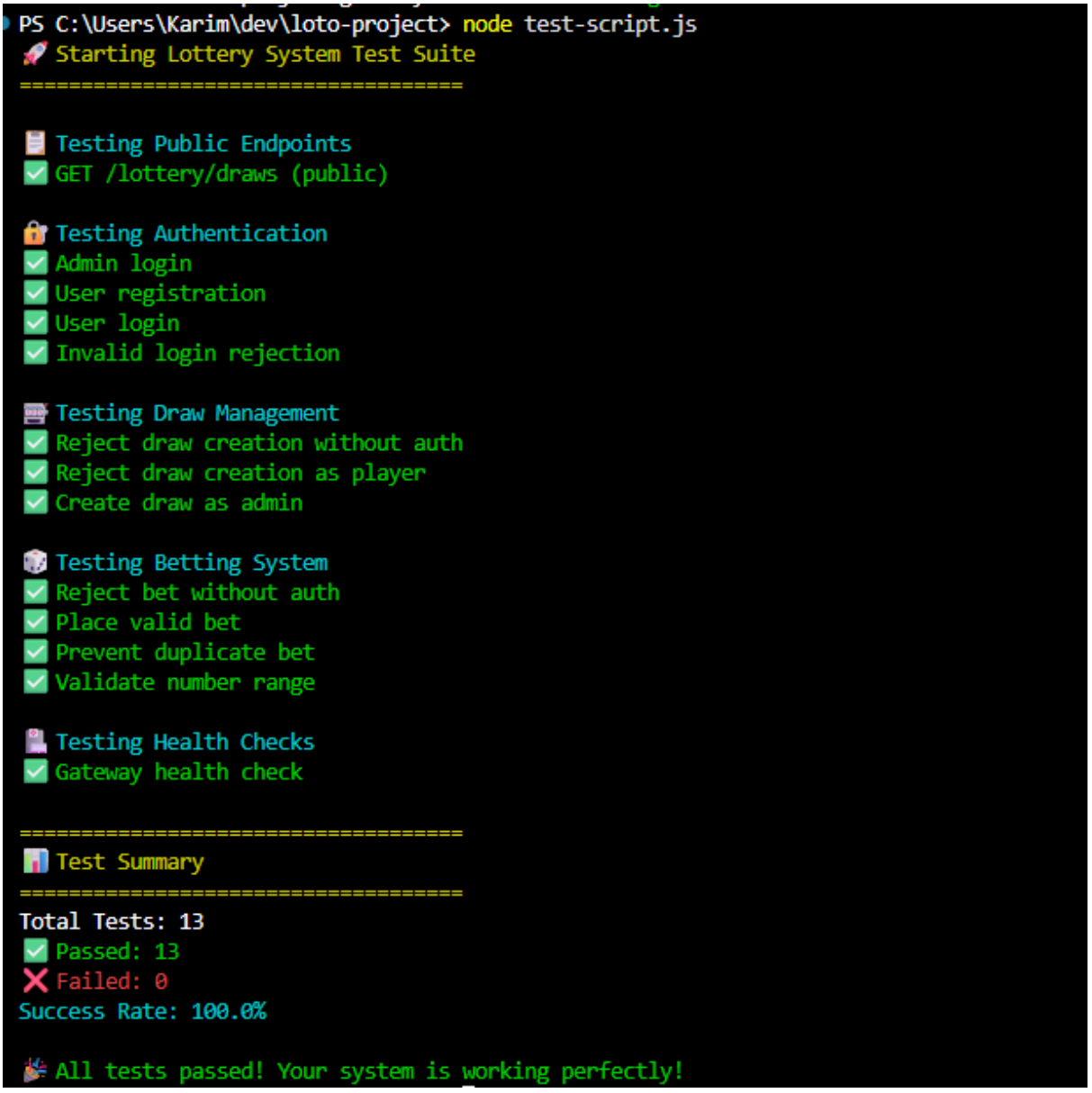
5. Tests et Validation

Tests automatisés :

- 13 scénarios de test
- Couverture complète des endpoints
- Tests de validation des données
- Tests de sécurité (authentification/autorisation)
- Résultat : 100% de réussite

Tests de charge :

- Services scalables horizontalement
- RabbitMQ gère la montée en charge
- Bases de données optimisées avec index



6. Conclusion

Le système développé répond à toutes les exigences du cahier des charges :

- Architecture distribuée et scalable
- Résilience aux pannes partielles
- Sécurité via JWT
- Communication asynchrone
- Code de qualité professionnelle

Le taux de réussite de 100% aux tests démontre la robustesse de la solution.