IT KOLLEDŽ
TALLINNA TEHNIKAÜLIKOOL

SPORTSBOOK API

Scope of work in Distributed Systems project

Student:	Gert Vesterberg
Student Code:	175871IDDR
Supervisor	Andres Käver

Tartu 2019

Table of Contents

1 Introduction

The goal of this project is to create a sports-betting product for the Kenyan market. According to the latest studies, digital betting is one of the second biggest internet activities amongst young Kenyans. There are currently two big players in the sports betting sector in Kenya - SportPesa and betPawa, which both have constant and stable growth in terms of player base and the total amount of transactions. [2]

Biggest barriers to market penetration are licensing requirements and payment processing contracts. The license for operating online-betting venture is 40 000 EUR, which is not much, compared to other regulated markets, but in order to obtain this license, local connections are required due to the widespread corruption in the country.

In order to process the payments (deposits and withdrawals), a contract with the most popular payment solution, mPesa is required. MPesa is a mobile-based payment solution, operated by Safaricom. Due to the poor nature of the country, credit cards are not widely used in Kenya.

Typical transaction size in Kenyan betting sector is 100 KSh (less than 1 EUR). [2] Therefore, when Bitcoin Lightning network (microtransactions) gain more momentum, funding the wallet with Bitcoin is also a viable solution.

The scope of this project is to create a minimum viable product (MV) for sports betting product. Although the author doesn't have any previous .NET development experience, he will still utilize his previous knowledge from online-casino and payments processing development.

2 Scope of work

2.1 Overview

The scope of the MVC is to develop 3 different microservices:

Wallet API - which allows the player to transfer money into his online sportsbook account and make withdrawals from it. Also, funding of the bets is deducted from his balance. Disclaimer: Wallet service does not include any integration with a payment processor (mPesa for example) as this requires a contract.

Sportsbook API - allows players to place a bet for certain match. It also contains player registration, which in theory could be separated to a different API, but due to time limitation of this subject, is part of Sportsbook API. Sportsbook API has also an integration with the odds provider API, which regularly updates the odds and pulls the new matches.

Frontend UI - A separate UI application, written in React, which has following functionalities:

- Signup
- Make a deposit
- Make a withdrawal
- Place a bet
- View results
- Administrator dashboard for player management

In sports betting, there are different ways to place a bet, which are called markets. For example, a player can make a bet based on the game result in the first quarter or bet on specific game result. In this MVC, Head to head is the only available market, where player must predict, which of the two opposing teams will win the match. Live betting (placing bets during the match) will not be available because this needs a contract with the live odds-provider.

For requesting the odds, https://the-odds-api.com/ API will be used, this allows to make 150 API requests per day, which is enough for the MVP.

All applications are stateless and will be run in Kuberenetes cluster. API services can be scaled up and down, depending on the system load. For MVP, DigitalOcean 3-node k8s cluster has been provisioned.

2.2 Additional services and middleware

Additional components, run on the same cluster include:

PostgreSQL database (will not be run in replication-mode in MVP) - used for storing the players, bets and wallet data.

Redis - will be used as a distributed cache (for example, keeping the state of the balance for each player).

Possibly a **RabbitMQ** / **Kafka** or similar message broker for communication between the sportsbook API service and wallet service (when the player places a bet, then the amount will be deducted from his/her balance).

This needs some more research as the author is not sure how easily AMQP protocol can be implemented in .NET Core.

Another approach is to use API gateway, which will combine 2 API calls into one single API request.

Load balancer - most probably NGINX or DigitalOcean native load balancer bound to k8s cluster. - to be decided.

2.3 Architecture

The architecture of this project will try to follow the best practices of Clean Architecture [3] and REST and distributed microservices. The author hopes to get at least 100 TPS performance out of this application.

Authentication and authorization will be based on JWT - the token's signature will be validated on each API / wallet request using the secret key. JWT will also contain the user context (ID) and role, which will be used to create an identity context during the request lifecycle and authorization of each of the API actions.

Wallet service does not store the players balance anywhere, the balance is calculated on-demand (during the deposit and withdrawal request) and stored to distributed REDIS cache. This should allow scaling services horizontally without having any inconsistencies of states between different cluster nodes.

Sportsbook API will also poll the odds regularly from the odds provider, using Hangfire scheduler [1]. To make sure the odds are not overwritten by several instances of the sportsbook API, database level locking will be used.

3 Diagrams

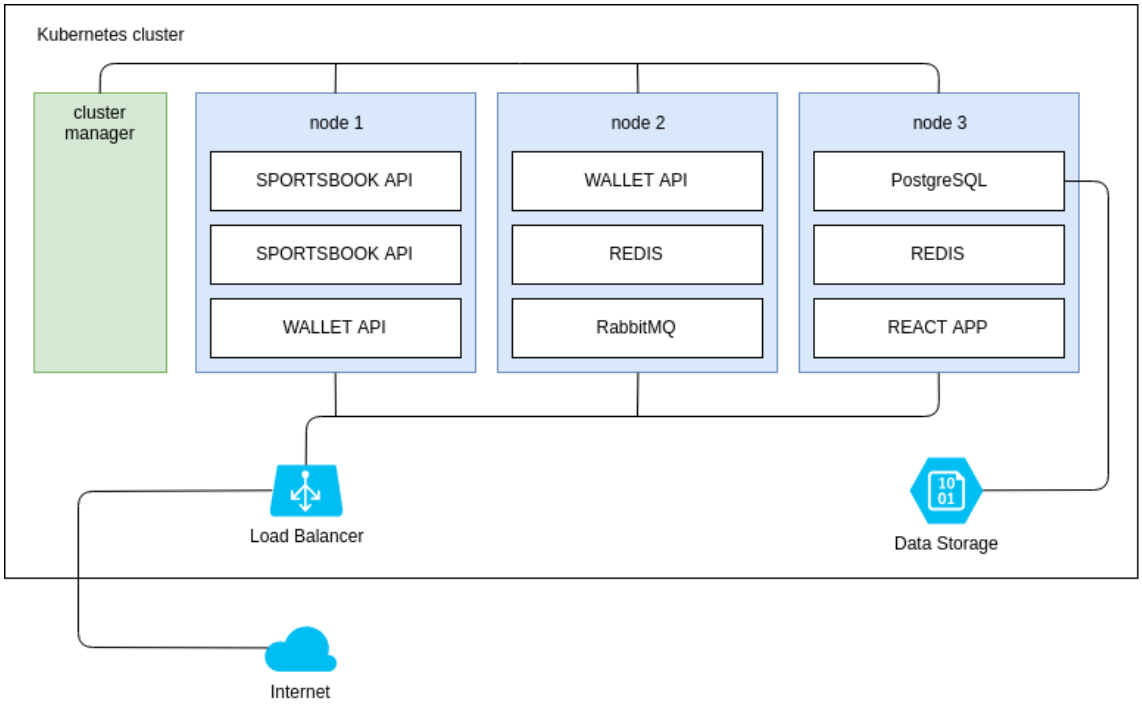

Diagram 1- Network architecture

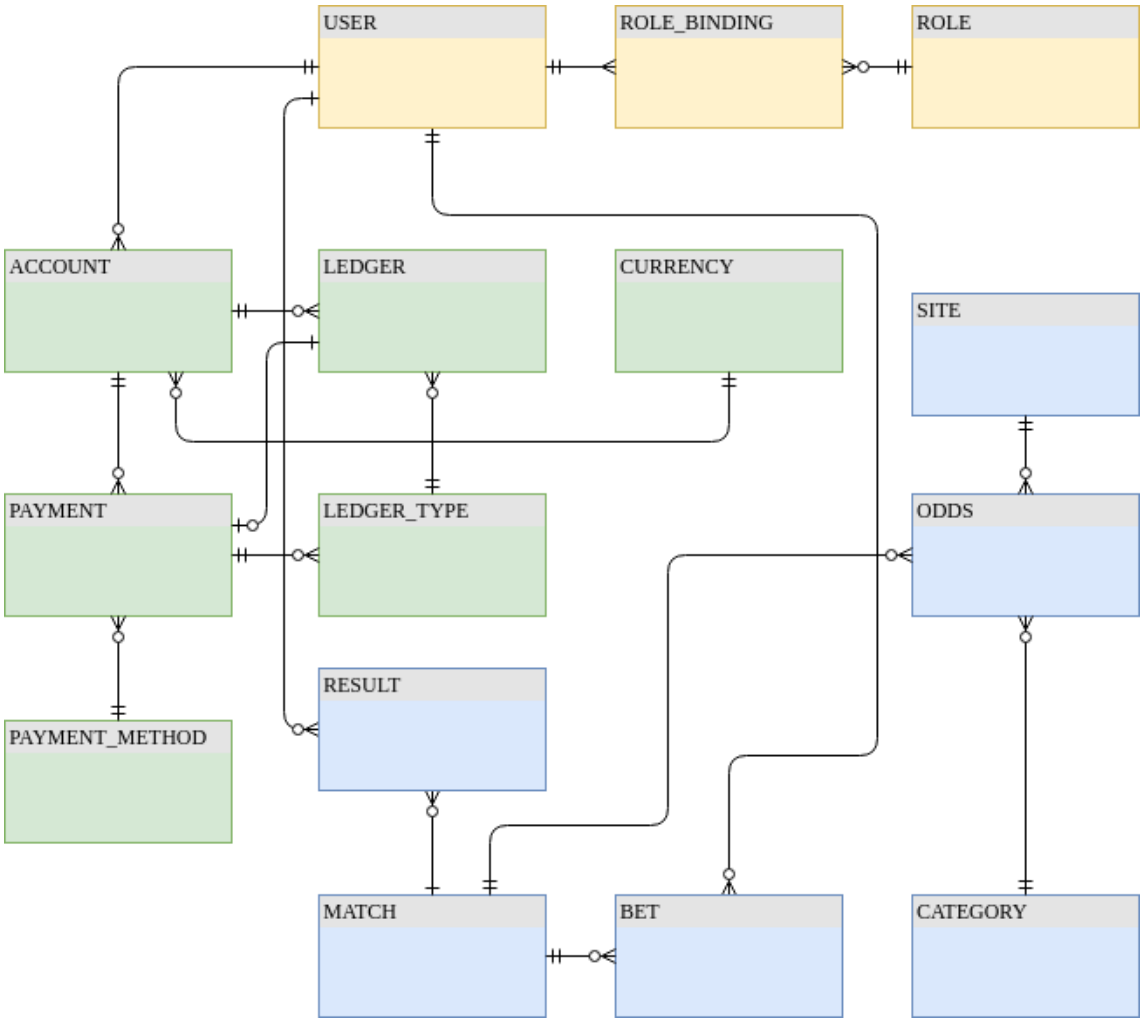

Diagram 2 – ERD

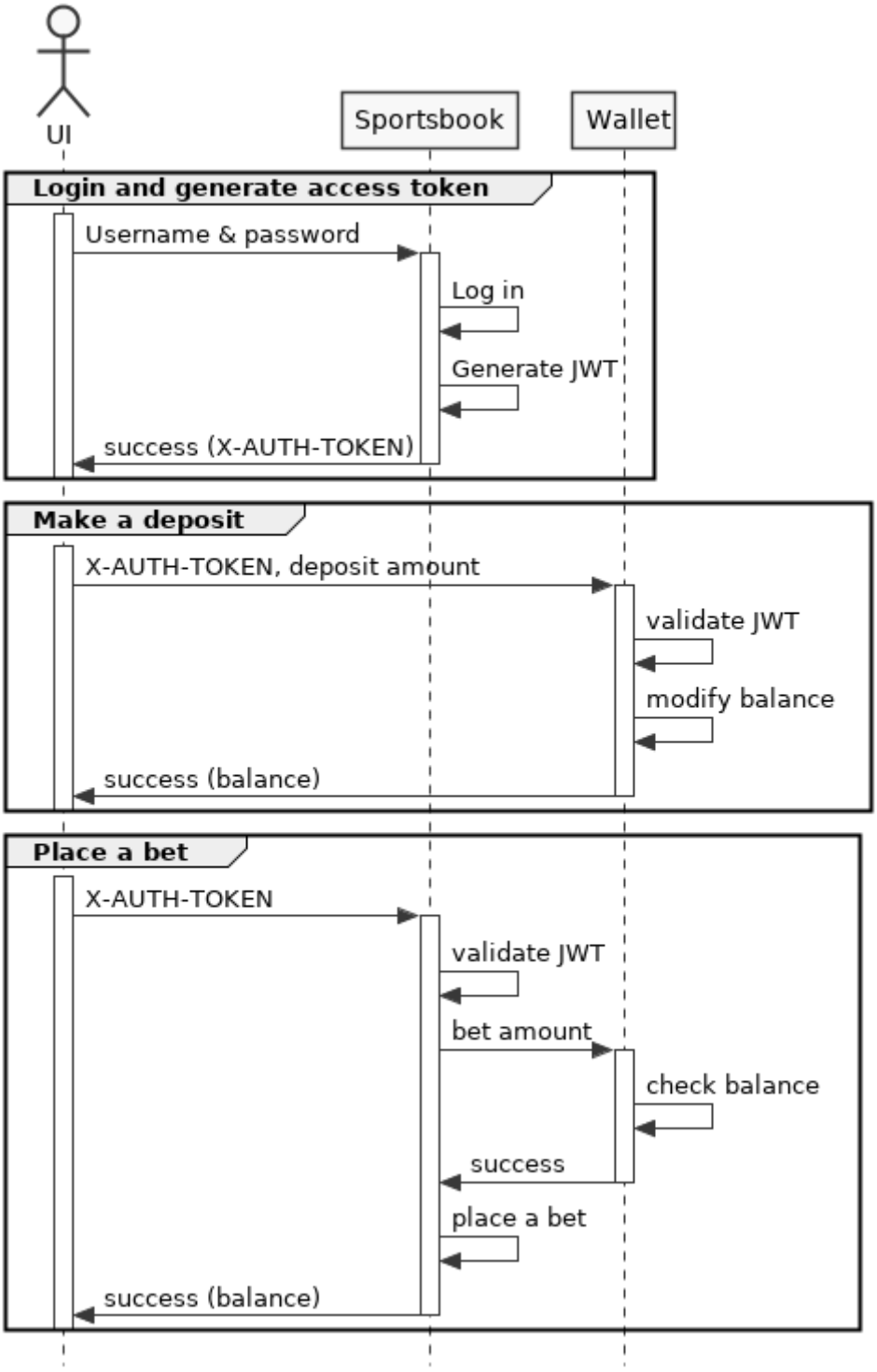

Diagram 3 - Main flows

4 REST endpoints

Payments service (/api)

- Account - user payment accounts (default: sportsbook)
- Currency - supported currencies
- Ledger - player monetary transaction records
- LedgerTypes - transaction record types (payment, bet)
- Payment - player payments (deposit, withdrawal)
- PaymentMethod - supported payment methods
- Product - supported products
- User - users

http://127.0.0.1:5080/swagger/index.html

Sportsbook service (/api)

- Category - Sports game categories management
- Match - Match management
- Odds - Odds for matches
- Result - Match results
- Site - Sites where odds are collected
- Team - Teams on matches

http://127.0.0.1:5080/swagger/index.html

6 References

[1] Hangfire, „Hangfire,“ [Webpage]. Available: https://www.hangfire.io/.

[2] ReachAfrica, „2018 Year in Review,“ [Webpage]. Available: https://www.findreach.com/uploads/2018_year_in_review.pdf/.

[3] Robert C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, 2017 [Publication]. Available: http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

7 Diagrams