# KRYPTO

Gamified Virtual Cryptocurrency Platform

## Cahier des Charges / Requirements Document

**Prepared by:** Taha Jaiti
**Supervised by:** Mr. Khalil Abouabdelmajid
**Date:** November 2025

# Contents

# 1. Project Context

## 1.1. Introduction

**KRYPTO** is an interactive web platform that gamifies cryptocurrency creation and trading. Users start with a base currency, called **KRYP**, and can use it to create new cryptocurrencies with custom names, images, abbreviations, and initial supply. They can trade these virtual assets, complete challenges, and earn badges—all within a secure, blockchain-powered environment. Unlike basic trading simulators, KRYPTO introduces a **real blockchain layer built in Java**, ensuring authentic, immutable transaction records while remaining risk-free since all tokens are virtual.

## 1.2. Problem Statement

Most crypto simulators are centralized, storing transactions in traditional databases, which removes the core educational purpose of understanding decentralization. They also lack competitiveness and gamification, making users lose interest quickly. KRYPTO addresses these problems by merging blockchain authenticity with gaming motivation, creating an environment that's both **fun** and **educational**.

## 1.3. Proposed Solution

KRYPTO aims to offer a scalable and interactive ecosystem by implementing:

- A custom-built blockchain in Java ensuring transaction immutability.
- A distributed backend based on microservices using Spring Boot.
- RabbitMQ for asynchronous inter-service messaging.
- Spring Cloud and Eureka for dynamic service registration.
- Optional Keycloak integration for identity management.
- Redis caching for performance optimization.

- PostgreSQL for core transactional data, with optional NoSQL support for analytics.
- React + TypeScript frontend for a smooth, real-time trading experience.

## 1.4. Objectives

- Build a functional blockchain to simulate real crypto transactions.
- Develop scalable backend microservices for modularity.
- Integrate caching, asynchronous communication, and service discovery.
- Deliver an engaging frontend with gamified features.
- Ensure security, scalability, and low latency.

## 1.5. Target Audience

- Students and developers learning blockchain and distributed systems.
- Gamers and competitive users interested in crypto simulation.
- Educators or training centers demonstrating blockchain principles.

## 1.6. Project Scope

- Blockchain-based backend with wallet, trading, and coin management.
- Gamified system with challenges, badges, and leaderboards.
- Responsive web interface using React.
- Integration of modern DevOps tools (Docker, optional Kubernetes).

# 2. Functional and Technical Description

## 2.1. Functional Description

### 2.1.1. User Roles and Stories

**Player Stories:**

- As a player, I want to register and create my personal account securely, so I can store and access my wallet anytime. This includes email verification, password hashing, and optional two-factor authentication for enhanced security.
- As a player, I want to view my KRYP balance and all my created coins, so I can track my assets in real time. The wallet should display a multi-currency view, including net worth calculated in KRYP equivalents based on current market prices, with breakdowns for each held coin.
- As a player, I want to create a new cryptocurrency by specifying its name, symbol, image, and initial supply (e.g., 1,000,000 tokens for a new "Doge" coin), paying a KRYP fee for creation. The initial supply influences the coin's starting price relative to KRYP, with market volatility and events adjusting it over time.
- As a player, I want to buy and sell user-created coins through a simulated market, so I can compete with others and increase my portfolio value. This includes coin-to-coin and coin-to-KRYP conversions, with blockchain-backed validation to ensure transaction integrity, incorporating fees (e.g., 1% transaction fee in KRYP), simulated volatility (e.g., price fluctuations based on supply-demand), and rewards for successful trades.
- As a player, I want to execute trades using strategies like limit orders (buy/sell at a specific price) or market orders (immediate execution at current price), handling edge cases such as partial fills during high volatility or insufficient balance warnings.
- As a player, I want to participate in daily and weekly challenges, so I can earn extra KRYP or special badges. Challenges are tied to real actions, such as "Trade 10 different

coins in a day" for a "Diversifier" badge or "Hold a coin through a 20% volatility drop" for a "Diamond Hands" badge.

- As a player, I want to see live leaderboards and achievements, so I can compare my progress with others. Leaderboards rank by net worth in KRYP, total trades, or challenge completions, with badges like "Top Trader" awarded to the highest weekly performer.
- As a player, I want to verify that all my transactions appear on the blockchain, ensuring transparency. This includes browsing transaction history with details like timestamps, hashes, and involved parties.

**Admin Stories:**

- As an admin, I want to monitor all transactions on the blockchain, to ensure the system integrity and prevent abuse. This includes real-time dashboards showing anomaly detection for unusual patterns like rapid fee exploitation.
- As an admin, I want to manage user accounts and suspend any malicious ones, with audit logs for actions taken.
- As an admin, I want to adjust KRYP economy parameters (fees, rewards, volatility factors, etc.) to balance gameplay, simulating real-world market interventions.

### 2.1.2. Core Features

- **Authentication:** Secure registration and login via JWT.
- **Wallet Management:** Each user has a wallet holding KRYP and user-generated tokens, with net worth calculations in KRYP equivalents and support for multi-currency displays.
- **Coin Creation:** Players can design and issue their own cryptocurrencies using KRYP, specifying name, symbol, image, and initial supply. The initial supply sets the starting market cap relative to KRYP, with prices evolving based on trading volume, simulated volatility, and market events.
- **Trading Engine:** Market operations simulate supply-demand driven price fluctuations, supporting limit/market orders, coin-to-coin/KRYP conversions, fees, and edge cases like slippage during volatile periods.
- **Blockchain Ledger:** Stores all transactions immutably with block verification.
- **Gamification:** Challenges, badges (e.g., "Top Trader", "Diamond Hands"), leaderboards, and achievements drive engagement, tied directly to trading and holding actions.

## 2.2. Technical Description

### 2.2.1. System Architecture

The platform is structured into microservices communicating asynchronously via RabbitMQ. Each service focuses on a specific domain (wallets, trading, blockchain, challenges, etc.), registered dynamically through Eureka and orchestrated with Spring Cloud.

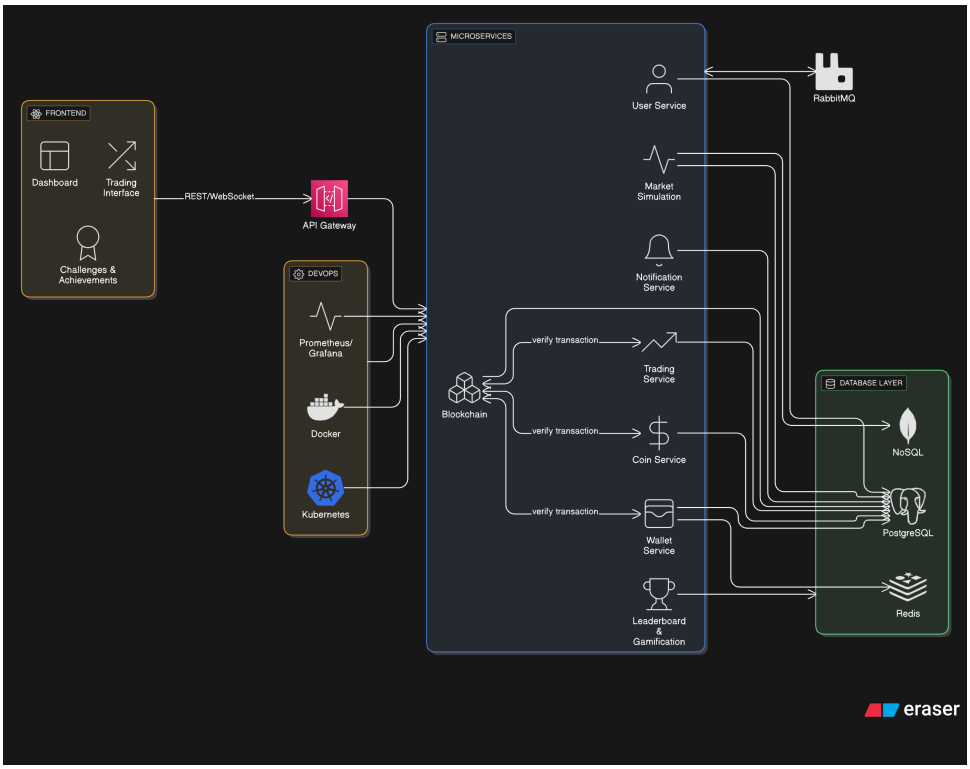


*Figure 1: Microservices Architecture*

### 2.2.2. Blockchain Design

The blockchain service handles:

- Block creation, validation, and chain linking.
- Proof-of-Work or simplified consensus mechanism.
- Transaction verification and hashing for immutability, including validation for coin creations with initial supply and trading conversions.

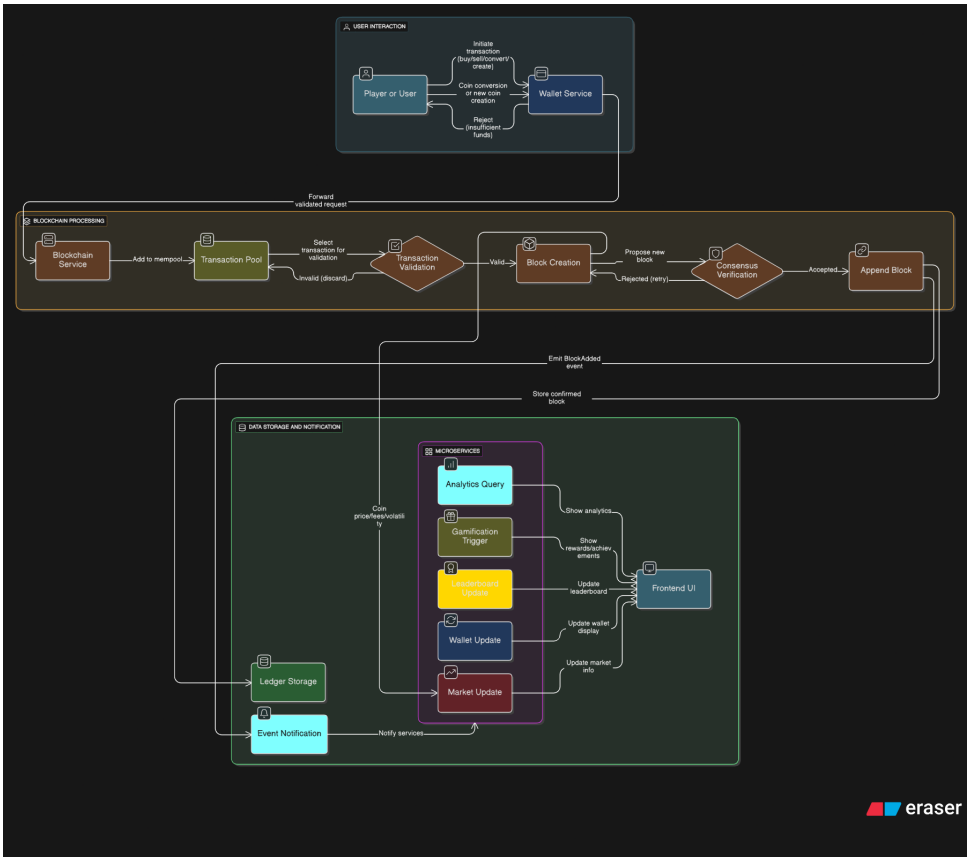


*Figure 2: Blockchain Flow Diagram*

### 2.2.3. KRYP Flow and Economy

- Every player starts with a base amount of KRYP.
- Creating a custom token requires paying a fee in KRYP and specifying initial supply, which determines the starting price relative to KRYP (e.g., market cap = initial supply × base KRYP value).
- Market values fluctuate based on trading volume, simulated volatility (e.g., random events like "market crash" reducing prices by 10-30%), and supply-demand dynamics.
- Challenges reward KRYP or limited edition badges, with edge cases like reward caps to prevent exploitation.
- KRYP acts as the base currency, similar to USD in real markets, for pricing all user-created coins.

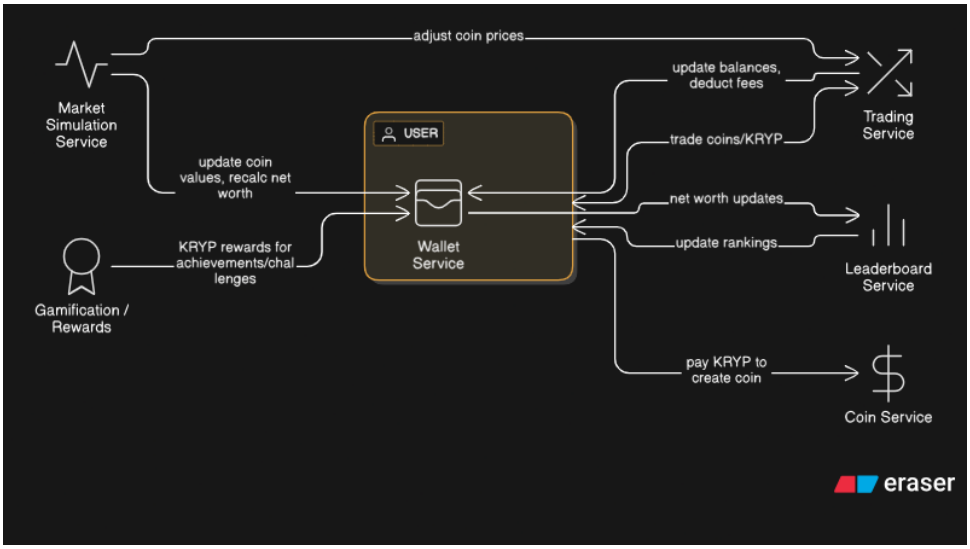


*Figure 3: KRYP Token Flow Diagram*

### 2.2.4. Technology Stack

- **Backend:** Java 21, Spring Boot, Spring Security, Spring Cloud, Eureka, RabbitMQ, JWT.
- **Frontend:** React, TypeScript, TailwindCSS, Vite, Zustand, React Query.
- **Database:** PostgreSQL (primary), optional MongoDB.
- **Caching:** Redis for high-speed reads.
- **DevOps:** Docker, optional Kubernetes.

### 2.2.5. Security and Performance

- JWT or Keycloak-based access control.
- Blockchain-based transaction verification for transparency.
- Redis and RabbitMQ minimize latency and ensure smooth scaling.

## 2.3. Expected Deliverables

- Fully functional blockchain-integrated KRYPTO platform.
- Modular microservices backend with complete documentation.
- Responsive frontend with live dashboards and real-time data.
- Complete technical documentation and deployment guide.

## 2.4. Project Timeline

- **Month 1:** Architecture planning, blockchain prototype.
- **Month 2:** Wallet and coin management services.
- **Month 3:** Trading simulation, caching, gamification.
- **Month 4:** Frontend, integration, testing, documentation.

# Conclusion

**KRYPTO** merges blockchain principles, gamification, and distributed system design into a single educational platform. It aims to showcase mastery of advanced backend engineering concepts—while providing users with a fun and realistic crypto experience built entirely on virtual assets.