CRYPTOZERO

Versão 0.0.1

DOCUMENTO DE ARQUITETURA

Responsável: Jesiel Viana da Silva

LOCAL BONNUM

HISTÓRICO DE REVISÕES

Data	Versão	Descrição	Autor
17/01/2019	0.1	Criação do documento	Jesiel Viana
30/01/2019	0.2	Revisão do documento	Jesiel Viana
01/02/2019	0.3	Revisão do documento	Rafael Marques
02/02/2019	0.4	Revisão do documento	Jackson Lima
29/03/2019	1.0	Atualização da arquitetura	Jesiel Viana

ÍNDICE

1. INTRODUÇÃO

Este documento fornece um projeto de arquitetura detalhada da plataforma de *marketplace* de criptomoedas Local Bonnum, concentrando-se nos requisitos funcionais significativos do ponto de vista da arquitetura e em três atributos-chave de qualidade: usabilidade, segurança e testabilidade. Esses atributos foram escolhidos com base em sua importância no projeto e na construção do aplicativo.

O objetivo deste documento é ajudar a equipe de desenvolvimento a determinar como o sistema será estruturado no mais alto nível. E também o gestor do projeto poderá usar este documento para validar que a equipe de desenvolvimento está atendendo aos requisitos acordados durante sua avaliação dos esforços da equipe.

2. VISÃO GERAL

O projeto Local Bonnum é uma plataforma web descentralizada que permite aos usuários anunciar e trocar criptomoedas por moeda fiduciária, permitindo com que pessoas que não tenham acesso a banco possam realizar transações financeiras (compra e venda de coisas) e guardar valores. Além disso, possibilita que empresas façam doações de criptomoedas para pessoas que não possuem conta em bancos.

A Figura 1 representa o diagrama de contexto do sistema, que mostra uma visão geral do sistema e como ele se encaixa no mundo em termos das pessoas que o utilizam e dos outros sistemas de software com os quais ele interage.

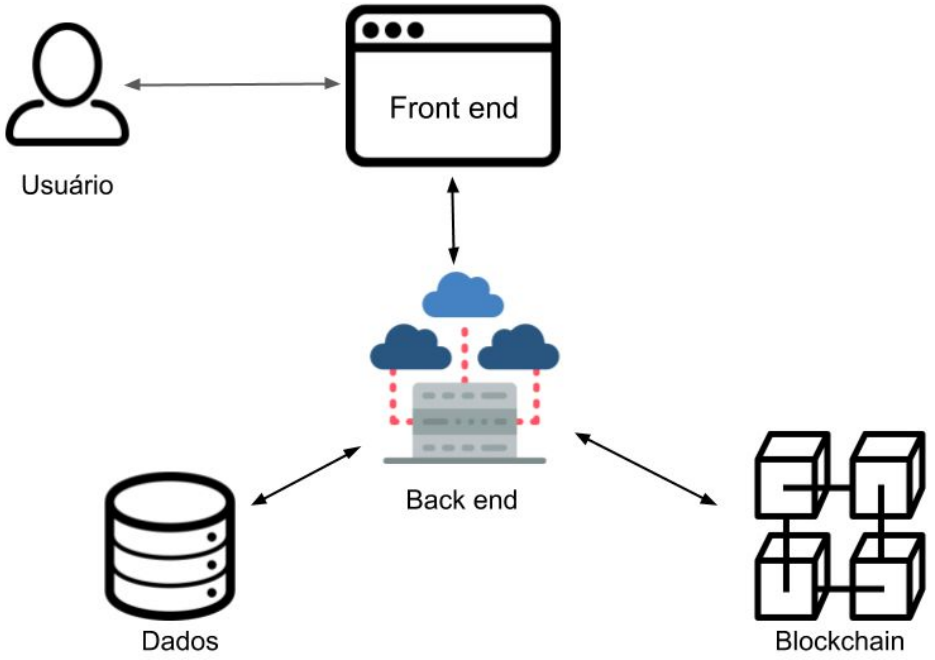

Figura 1 - Diagrama de visão geral da arquitetura

Abaixo será apresentado uma descrição de cada um dos elementos da Figura 1.

Usuário

Agente externo (pessoa), que acessa o sistema para realizar alguma operação.

Front-end

Interface gráfica da aplicação, que possibilita a interação com o usuário. No contexto da plataforma Local Bonnum, o front-end é disponibilizado via navegador Web. O front-end interage com o back-end através de solicitações HTTP seguindo o modelo RESTful.

Back-end

Módulo responsável pela lógica de negócios, manipulação dos dados e fornecimento de uma API para comunicação com o front-end.

Dados

Camada responsável pelo armazenamento e recuperação dos dados persistentes do sistema.

Blockchain

Blockchain é uma tecnologia para registro compartilhado de transações que permite, através de técnicas criptográficas, agilização de transações complexas.

3. DESCRIÇÃO DO PROBLEMA

Bilhões de pessoas são desbancarizados[1], com isso, não têm acesso direto à bancos, não conseguem efetuar transações financeiras. Há também situações em ambiente de guerra, refugiados e de hiperinflação onde pessoas perdem o direito de acesso aos bancos e o poder de compra com a moeda fiduciária.

4. RESPONSÁVEIS E INTERESSADOS

Equipe de desenvolvimento

- Franciosney Souza
- Jackson Lima dos Santos
- Jesiel Viana da Silva
- João Batista Fonseca Cavalcanti
- Rafael de Brito Marques
- Renan Oliveira Nunes
- Wilson Felipe dos Santos Lima

[1] https://exame.abril.com.br/negocios/dino/desbancarizacao-traz-oportunidades-ao-varejo-no-brasil/

Cliente (Bonnum)

- Alessandro Lima (PO) - CDO Bonnum
- Edmilson Rodrigues - CEO Bonnum
- Eric Cavalcanti - CTO Bonnum

5. REQUISITOS FUNCIONAIS

Os requisitos funcionais significativos do ponto de vista da arquitetura são:

- Módulo de Usuário (RF01 - RF05) que envolve o cadastro de usuário no padrão *Know Your Customer (*KYC[2]) e autenticação em dois fatores.
- RF06 - Depósito de Criptomoedas: o usuário deposita criptomoedas no endereço gerado pela plataforma;
- RF07 - Transferência de Criptomoedas: o usuário transfere para terceiros criptomoeda da sua carteira na plataforma;
- RF08 - Realizar Escrow: - A plataforma retém o saldo em criptomoeda da negociação enquanto esta não for concluída.

Para mais informações sobre o escopo positivo e negativo do projeto e uma descrição mais detalhada de todos os requisitos funcionais consulte o Documento de Visão.

6. REQUISITOS NÃO FUNCIONAIS

Requisitos não funcionais que impactam na arquitetura do projeto:

Segurança

- RNF03 - Autenticação de dois fatores;
- RNF04 - Usuário será bloqueado após 03 tentativas de login sem sucesso;
- RNF05 - Apenas os usuários autenticados na plataforma e com as devidas autorizações podem realizar transações;
- RNF06 - Uso de criptografia para transação de criptomoedas;
- RNF07 - O sistema deverá implementar a política Know Your Customer (KYC) para cadastros de novos usuários;

[2] Verificação de identidade do cliente através de documentos e/ou outros meios. https://www.thomsonreuters.com.br/pt/financeiras/blog/repense-suas-estrategias-de-kyc.html

7. COMPONENTES DA ARQUITETURA

A visão detalhada da arquitetura descreve a estrutura em camadas do sistema. Ela mostra como o sistema é estruturado como um conjunto de unidades de código funcional ou módulos.

A arquitetura representada na Figura 2, mostra as camadas que compõem o sistema e a hierarquia dessas camadas. A camada de apresentação é encapsulada pelo front-end, enquanto as camadas inferiores são encapsuladas pelo back-end.

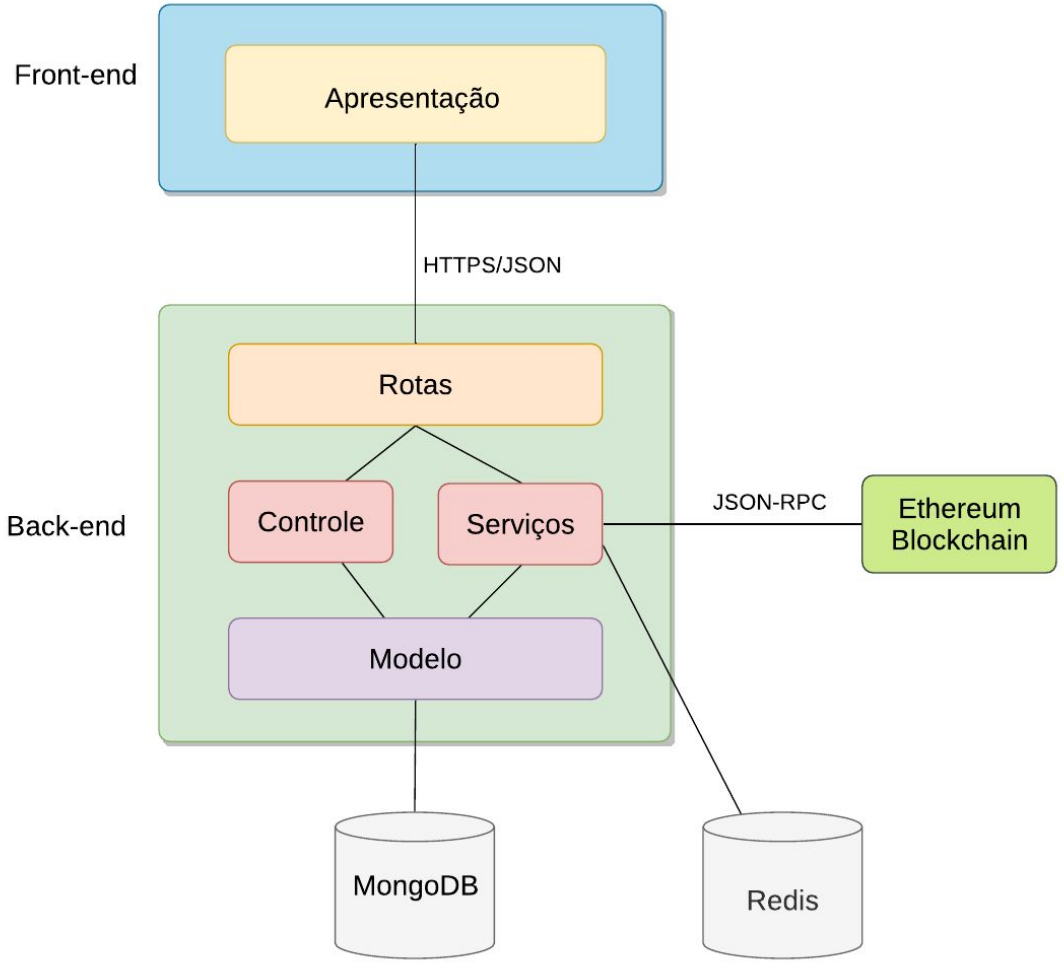

Figura 2 - Representação dos componentes e interações

Cada camada é denotada como retângulo. Os retângulos menores dentro representam um conjunto de elementos contidos em cada camada, porém não representam uma lista completa de entidades. Os grandes retângulos demonstram a separação física das camadas entre front-end e back-end. Abaixo será apresentado uma descrição de cada uma das camadas do sistema.

1. Front-end

O front-end será desenvolvido utilizando as tecnologias Vue.js com Bootstrap.

- **Camada de Apresentação**

  A camada de apresentação será desenvolvida utilizando Vue.js com Bootstrap. Ela será executada no navegador Web do usuário de forma responsiva podendo inclusive ser acessada via dispositivos móveis como tablets e smartphones. O principal objetivo dessa camada é fornecer uma interface gráfica com tudo que o usuário precisa para concluir suas tarefas no sistema.

2. **Back-end**

   O back-end do Local Bonnum será implementado em um servidor Node.js com Express e Mongoose. O servidor fornece uma API para comunicação através do protocolo HTTPS seguindo o modelo RESTful.

- **Camada de Rotas**

  A camada de rotas é ponto de acesso para o front-end, ela é responsável por modularizar diferentes recursos da aplicação. Ela será implementada de acordo com o modelo REST. Cada rota atuará como ponto de conexão, servindo como uma interface para um recurso que pode ser acessado pela camada de apresentação.

  A modularização das rotas dentro do sistema também facilitará a identificação de possíveis defeitos, permitindo fácil manutenção. Cada rota pode ser testada individualmente, o que permite uma boa testabilidade em todo o sistema.

- **Camada de Controle**

  A função da camada de controle é implementar a lógica de negócios principal. Ela também servirá como conexão entre a camada de rotas e o modelo de domínio, mantendo assim a separação de interesses.

  Certos recursos, como gerenciamento de carteira virtual e transferência de criptomoedas, exigem acesso a APIs externas. A camada de controle será responsável por interagir com essas APIs externas.

- **Camada de Modelo de Domínio**

  O modelo de domínio contém todas as representações de objetos de dados do sistema no sistema, conhecidas como entidades. Isso também inclui métodos associados para qualquer objeto que contenha sua própria funcionalidade.

  As entidades do modelo de domínio também são responsáveis pela coordenação de toda a comunicação entre os objetos na camada de domínio e suas respectivas coleções no banco de dados.

- **Camada de Dados**

Todas as informações persistentes e qualquer dado oriundo de integração de API externa (por exemplo, Blockchain) formam a camada de origem de dados. Isso inclui o banco de dados MongoDB que junto com a Blockchain conterão todos os dados do sistema.

- **Comunicação entre Front end e Back end**

  Front end e Back end se comunicam entre si usando um padrão de troca de mensagens request-response. A camada de apresentação (Front end) envia uma solicitação à camada de roteamento (Back end) e esta envia uma resposta como retorno. Front end e Back end usam JSON como linguagem comum para comunicação seguindo o modelo RESTful.

- **Blockchain**

  Neste projeto será utilizado a blockchain do Ethereum, através de Smart Contracts. A integração entre a camada de controle e o Ethereum será realizada via web3 e as transações serão feitas usando o ethereumjs-tx, que são bibliotecas usadas na integração com NodeJs.