FutQuiz - Registro de Decisões Arquiteturais

Indaiara Bonfim
Lucca Lobo
Pedro H. Costa
Rian Vasconcelos

OVERVIEW

Este documento apresenta as decisões estruturais de um sistema de quizzes denominado FutQuiz. Neste *software* os usuários poderão testar os seus conhecimentos acerca de times de futebol em quizzes gerais e personalizados.

Este documento é uma proposta inicial e pode sofrer alterações durante o desenvolvimento.

INTRODUÇÃO

O FutQuiz é um aplicativo de quizzes para amantes do futebol brasileiro. Os jogadores poderão competir entre si em diferentes dificuldades de quizzes, cada uma com um ranking próprio.

Para incentivar do espírito competidor entre os jogadores, o jogo terá um sistema de créditos que servirão como pagamento e recompensa em certos quizzes. A proporção entre custo de entrada e recompensa está listada no documento de requesitos, REQ15.

Um jogador comum terá acesso aos quizzes, quizzes gerais—Produzidos proceduramente pelo programa através de um banco de questões—e quizzes personalizados—Criados pelos administradores a partir da seleção e criação de questões.

CARACTERÍSTICAS ESTRUTURAIS

De acordo com os requesitos listados, o time de arquitetos listou quatro características essenciais para este sistema: (a) Segurança, (b) Elasticidade, (c) Modulariadade e (d) Resiliência.

É necessário que o jogo tenha sistemas seguros, especialmente o de pagamentos, sendo capaz de lidar com inúmeros clientes sem o vazamento de informações. Módulos de pagamento são comumente relegados a APIs externas, contudo, o contato inicial e o registro das informções são armazenados juntamente a outras informações sensíveis. O vazamento destes dados pode gerar prejuízos imensos.

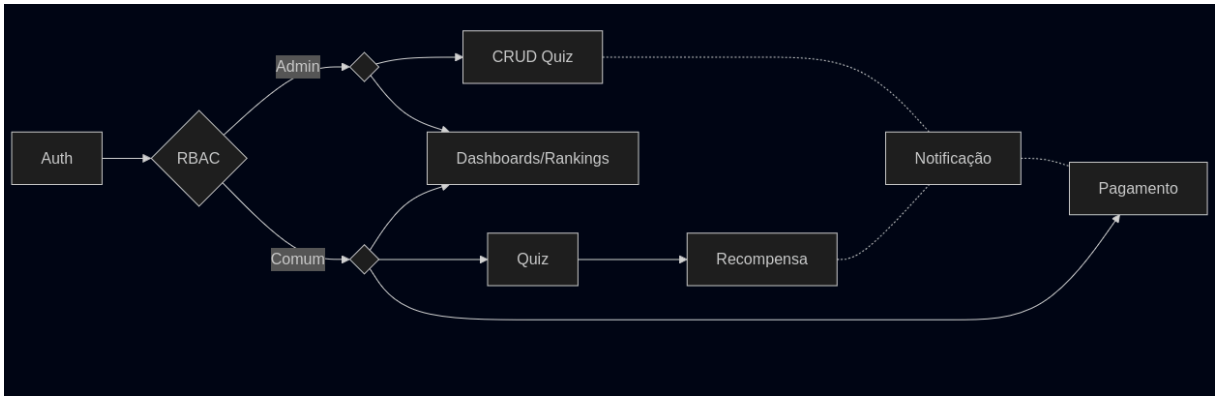
A Elasticidade é considerada como uma característica arquitetural importante devido a variabilidade de jogadores simultâneos no aplicativo. Diferentemente de outros produtos que possuem uso constante, o FutQuiz pode sofrer com picos em momentos inesperados, especialmente no lançamento de novos quizzes. Lidar com esses picos sem falhas é de extrema importância.

A Modularidade é considerada essencial, pois atualizar e realizar manutenções em sistemas desacoplados é mais simples. Desta maneira, substituir um sistema ou utilizar um *failover* torna-se mais fácil porque a comunicação entre os módulos pode ser mantida apesar da substituição.

Por fim, a Resiliência serve para garantir que as partipações dos usuários em quizzes não seja facilmente perdida. Mesmo que haja uma perda de contato momentânea durante um quiz, o progresso do usuário não pode ser perdido completamente. Especialmente se houve um custo monetário para participar do quiz. Este tipo de comportamento (quedas e perda de conexão) pode levar a expriências frustrantes e perda de interesse pelos jogadores.

WORKFLOW

Abaixo segue uma proposta de funcionamento e possíveis ações de um usuário, admin e comum, podem tomar durante o uso do aplicativo. Neste *work-flow* se encontram apenas as ações consideradas essenciais para a identificação dos sistemas.



Com este *workflow* é possível identificar os seguintes sistemas:

1. Auth, para login e criação de conta;
2. Quiz, para criação e uso;
3. Notificação;
4. Pagamento, compra e recebimento de créditos;
5. Dashboards/Rankings, para visualizar métricas;

SISTEMAS

O FutQuiz terá uma arquitetura baseada em cliente-servidor onde o aplicativo final será o cliente que se comunica com o servidor, que por sua vez possui vários sistemas e uma arquitetura própria. Abaixo estão descritas as arquiteturas de cada sistema e o seu papel principal.

É possível considerar o sistema geral como um de microserviços, pois ele será dividido entre componentes que realizam uma comunicação entre si.

AUTH

A lógica principal do sistema de autenticação será relegado a um serviço externo como Firebase Auth. Isto será feito para garatir que os usuários possam atrelar suas contas a outros serviços e assim tornar mais fácil a sua conexão. Para além disto, serviçoes especializados em autenticação são capazes de oferecer Multi Factor Authentication e outros sistema de segurança.

Com o usuário conectado, o sistema de autenticação irá controlar o acesso via RBAC, atrelando “papéis” aos usuários e limitando o controle da aplicação através deles.

QUIZ

O Sistema de quiz tem dois papeis: (a) Criar, editar (CRUD) quizes, questões e times, e (b) Gerenciar a participação dos jogadores (Contabilizar tempo, validar respostas e evitar trapaças).

A prioridade do sistema são os quizzes. Portanto, para evitar trapaças, um sistema confiável deve averiguar as respostas e tempo gasto em cada questão. Preferencialmente tal sistema deve ser capaz de lidar com múltiplos jogadores simultaneamente.

A segunda parte do sistema, criação e edição, pode ser realizada de maneira assíncrona com edição local e atualização na nuvem quando pronto.

Desta maneira, propomos:

- Uma arquitetura Event-Driven Cliente-Servidor para averiguação de respostas em tempo real.
- Com uma interface (API) para edição e criação Local e envio para a nuvem.

CRIAÇÃO

Os administradores poderão:

- Cadastrar Times;
- Cadastrar perguntas;
  - Perguntas cadastradas necessitam de: Time; Dificuldade; Lista de Respostas Possíveis; Resposta Correta; e Questão

Assim, os quizzes são gerados de maneira procedural mediante requisição do usuário. O usuário pode escolher o nível das perguntas, a quantidade de questões, com valores pré-definidos, e os times.

Será possível também criar quizzes com uma *pool* de questões reduzida para quizzes únicos com recompensa. Nestes quizzes será possível definir a quantidade de créditos necessária para participar.

PARTICIPAÇÃO

Ao participar de um quiz, o Cliente irá avisar ao Servidor do início com as informações necessárias (Usuário e Quiz, incluindo dificuldade e quantidade de questões se configurável). O fim do quiz se dá quando (a) o Cliente sinaliza o fim—por escolha própria, fim de tempo ou por chegar ao fim—ou (b) o Quiz é interrompido por um administrador.

Ao fim do quiz o usuário verá a quantidade de acertos, validado pelo Servidor. O Servidor será responsável por contabilizar o tempo gasto em cada questão.

Preferencialmente o sistema terá uma comunicação baseada em Eventos, como o Kafka e RabbitMQ. Desta forma, é possível manter uma história dos quizes feitos por um determinado usuário, com erros e acertos, além de permitir

que o Servidor consuma estas informações à medida que possível, permitindo elasticidade.

Contudo, se a comunicação através de uma API Web for mais fácil, então deve-se garantir que a carga seja distribuída adequadamente durante momentos de pico.

NOTIFICAÇÃO

O sistema de notificações vai depender fortemente de um serviço externo como Firebase Cloud Messaging (FCM) para a entre das notificações. Sendo assim, o sistema ficará responsável por gerar as notificações.

O envio das notifições será feito mediante (a) o surgimento de um novo quiz, (b) confirmação de pagamento e (c) quando houver uma nova recompensa de quiz.

O sistema responsável por se comunicar com o FCM ou outro provedor poderá ser chamado através de eventos, reutilizando o sistema de quizes e desacoplando os diferentes serviços. Esta arquitetura Event-driven vai garantir que nenhuma notificação seja perdida mesmo que o módulo de Notificação esteja fora do ar, devido à natureza de sistemas como Kafka e RabbitMQ.

PAGAMENTO

Similarmente ao sistema de Autenticação e Notificação, o sistema de pagamento depende da utilização de uma API externa para o seu uso, como Stripe, Mercado Pago e Google Pay. A implementação da lógica de pagamento vai ser influenciada pela escolha de API.

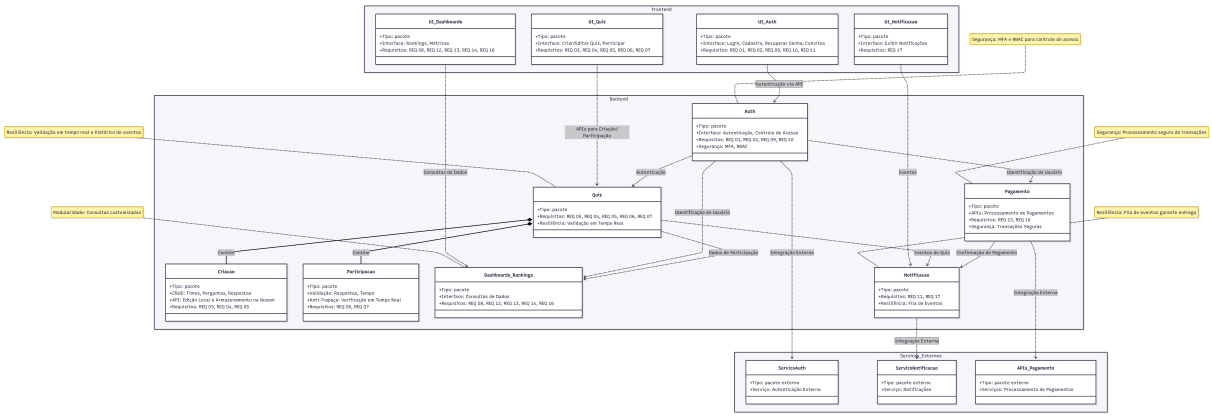
DASHBOARDS/RANKINGS

Os dashboards/rankings serão os sistemas mais próximos da camada de persistência, o banco de dados. Eles possuírão uma interface, preferencialmente definida em GraphQL, para a criação de dashboards de métricas variadas. Desta forma, os Rankings serão *queries* pré-definidas para cada quiz e rank geral.

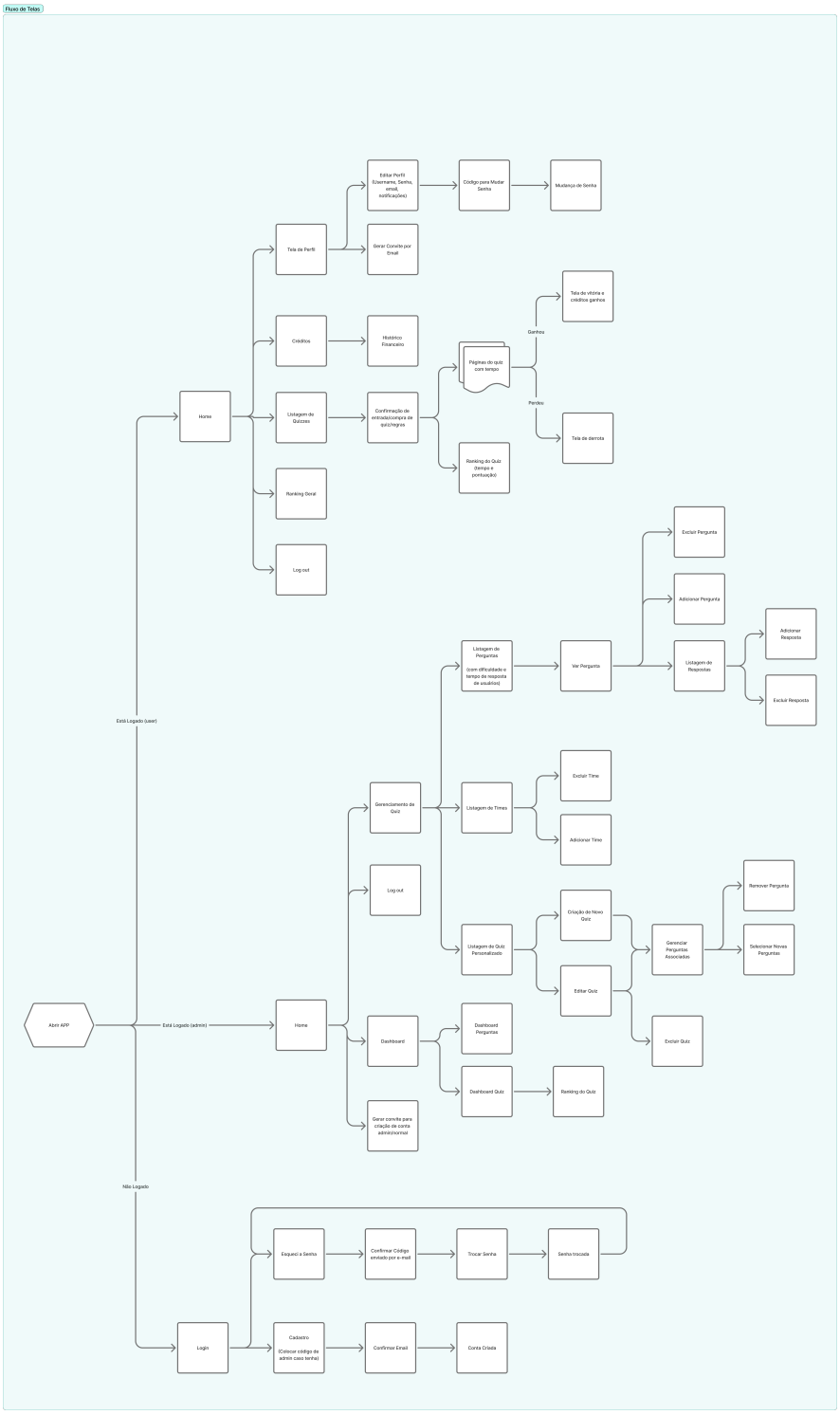
Este módulo é o mais simples, com uma implentação gráfica do cliente nos aplicativos e uma implementação do servidor para a geração das informações.

ANEXOS

DIAGRAMA DE MÓDULOS



FLUXO DE TELAS



Fluxo de Telas no Figma:

https://www.figma.com/design/wcp4HmMWO6trJUX7UyoMpj/MATA62-Final?node-id=0-1&p=f&t=vYXWe8XK0siUrkBk-0

DIAGRAMA DE COMPONENTES