PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído

Projeto Integrado

Relatório Técnico

Plataforma de palpites em partidas da Copa do Mundo "Pitaco FC"

Paulo Henrique da Cruz

Araraquara
Dezembro, 2022.

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

Projeto Integrado – Arquitetura de Software Distribuído 2
1. Introdução 3
2. Cronograma do Trabalho 5
3. Especificação Arquitetural da solução 6
3.1 Restrições Arquiteturais 6
3.2 Requisitos Funcionais 7
3.3 Requisitos Não-funcionais 9
3.4 Mecanismos Arquiteturais 10
4. Modelagem Arquitetural 11
4.1 Diagrama de Contexto 11
4.2 Diagrama de Container 12
4.3 Diagrama de Componentes 14
5. Prova de Conceito (PoC) 16
5.1 Integrações entre Componentes 16
5.2 Código da Aplicação 22
6. Avaliação da Arquitetura (ATAM) 26
6.1. Análise das abordagens arquiteturais 26
6.2. Cenários 26
6.3. Evidências da Avaliação 27
6.4. Resultados Obtidos 36
7. Avaliação Crítica dos Resultados 37
8. Conclusão 40
Referências 42

1. Introdução

Em ano de copa do mundo, onde as maiores seleções de futebol se reúnem para disputar o título mundial, a paixão de fãs e torcedores pelo esporte e suas seleções fica mais evidenciada e aparente. Pensando nesse contexto, o presente projeto propõe a criação de uma plataforma web responsiva de integração de partidas que permita o acompanhamento em tempo real dos jogos, utilizando-se de gamificação para estimular a competição entre os jogadores por meio de palpites nos resultados, mensurando a assertividade dos palpites de cada jogador em um contexto global e também dentro de um determinado grupo.

A constante evolução tecnológica e acesso aos meios de comunicação por toda a sociedade, seja por meio de celulares/smartphones, tablets e televisão, auxilia que toda a sociedade esteja por dentro das novidades do mundo do entretenimento, tendo o conteúdo consumido por meio dessas telas/equipamentos, e buscando formas de diversão entre amigos e familiares, entre os mais variados segmentos (BATISTA, 2018).

Atualmente o modo como os jogos são "experimentados" tem sofrido constantes mudanças, sendo possível perceber a união do mundo real e o mundo virtual (JUNIOR, 2014), como por exemplo Pokémon Go, que faz com que seus jogadores criem seu avatares, e tenham de caminhar para localizar e capturar novos pokémons. O mesmo cenário tem sido aplicado em jogos esportivos, como o caso mais conhecido no Brasil, o Cartola FC, onde o desempenho do "cartoleiro" é mensurado de acordo com o desempenho de jogadores de futebol do mundo real.

Fantasy Sports Games, de forma geral, são jogos virtuais inspirados em modalidades esportivas do mundo real, sendo específicos de um campeonato e/ou liga. Permitem que seus usuários criem e gerenciem seus times/equipes, selecionando os membros de seus times, através de acesso em plataformas mobile e/ou web, permitindo uma competição entre seus usuários, através de partidas onde o desempenho da pessoa real rende pontos para o jogador virtual (ALMEIDA; ALMEIDA; LIMA, 2015).

Cartola FC é o caso mais famoso de fantasy sport game brasileiro. Criado no ano de 2005 o jogo tem crescido ano após ano, e de passatempo se tornou um grande e rentável negócio que envolve clubes de futebol, emissoras e patrocinadores, além dos usuários. Criado e mantido pela Globo, o jogo atrai anualmente milhares de usuários, que escalam seus times rodada a rodada, pontuando e sendo ranqueados de acordo com o seu desempenho. O jogo possui versão gratuita, com recursos limitados, mas que permite que os usuários possam aproveitar a brincadeira, e conta também com uma "versão pró", onde cada usuário passa uma taxa anual de R$49,90, onde o usuário tem alguns recursos extras e também concorre a prêmios em dinheiro e um prêmio especial de um carro no final do campeonato para o maior pontuador de forma global. O jogo é baseado no Campeonato Brasileiro da Seria A, disputado anualmente por 20 clubes em rodadas de turno e returno que ao todo somam 38 rodadas. Os usuários escalam seus times, selecionando jogadores de clubes reais e pontuando de acordo com o desempenho do jogador em cada partida, sendo que a pontuação pode ser positiva e negativa, baseada em regras pré-determinadas e de conhecimento de todos os usuários (NEVES, 2019).

Para atender as expectativas e o desenvolvimento proposto, apresenta-se os 3 objetivos macro do projeto:

- Ser uma plataforma de fácil acesso, permitindo a sua utilização nos diversos navegadores web e mobile;
- Ter uma interface amigável e funcionalidades bem definidas, afim de evitar dificuldades de entendimento dos usuários;
- Ser escalável e tolerante a falhas, permitindo a aplicação possa acompanhar o crescimento do número de usuários ao decorrer do tempo.

Afim de refinar os requisitos foram definidos os objetivos específicos abaixo:

- Ser uma plataforma web responsiva;
- Permitir cadastro de usuário e login por meio de redes sociais, primariamente Google e Facebook;
- Permitir a utilização de vários campeonatos além da Copa do Mundo;
- Permitir que os usuários realizem seus palpites com janelas de fechamento;

- Fornecer dados estatísticos sobre jogos e equipes;
- Permitir que os usuários acompanhem os jogos em tempo real;
- Atualizar placares e pontuação em tempo real;
- Definir mecanismos de ranqueamento de usuários;

Afim de atender esses objetivos, a plataforma Pitaco FC será desenvolvida com tecnologias atuais, utilizando o framework Angular para o desenvolvimento do front-end, a linguagem Java para construção do back-end, e o MySQL como sistema de gerenciamento de banco de dados.

2. Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

Datas		**Atividade / Tarefa**	**Produto / Resultado**
De	**Até**		
14/04/2022	16/ 04/2022	1. Cronograma do Trabalho	Construção desta tabela
14/04/2022	24/04/2022	2. Contextualização do trabalho	Construção da contextualização deste projeto
25/04/2022	26/04/2022	3. Definição dos requisitos Arquiteturais	Lista dos requisitos Arquiteturais identificados
26/04/2022	27/04/2022	4. Definição dos requisitos Funcionais	Lista dos requisitos funcionais identificados
27/04/2022	28/04/2022	5. Definição dos requisitos Não-funcionais	Lista dos requisitos Não-funcionais identificados
28/04/2022	29/04/2022	6. Definição dos Mecanismos Arquiteturais	Lista dos Mecanismos Arquiteturais identificados
02/05/2022	03/05/ 2022	7. Construção dos Diagramas de Contextos	Diagrama de contexto criado no Draw.io e documentado
05/05/2022	15/05/2022	8. Revisão da Etapa 1	Documento Etapa 1 revisado
16/05/2022	18/05/2022	9. Apresentação em PPT da Etapa 1	PPT
30/07/2022	30/07/2022	10. Construção do vídeo de apresentação da Etapa 1	Vídeo criado da Etapa 1
30/07/2022	30/07/2022	11. Submissão do vídeo da etapa 1 no Youtube	Vídeo disponibilizado publicamente no Youtube
14/04/2022	14/08/2022	12. Publicação no repositório Github Etapa 1	Arquivos produzidos no Github disponíveis abertamente
16/08/2022	18/08/2022	13. Construção dos Diagramas de Contêineres	Diagramas de contêineres

18/08/2022	20/08/2022	14. Construção dos Diagramas de Componentes	Diagramas de componentes
20/08/2022	01/10/2022	15. Código da aplicação	Diagramação de código da aplicação
14/10/2022	14/10/2022	15. Publicação no repositório Github Etapa 2	Arquivos produzidos no Github disponíveis abertamente
17/10/2022	18/10/2022	16. Análise das abordagens arquiteturais	Descrição das Análises Arquiteturais
23/10/2022	26/10/2022	17. Cenários para realização de testes	Descrição dos cenários de testes para validar requisitos
01/11/2022	07/11/2022	18. Evidências da avaliação	Documento de avaliação do sistema
09/11/2022	15/11/2022	19. Resultados obtidos	Documentação dos resultados da avaliação
16/11/2022	21/11/2022	20. Avaliação crítica dos resultados e conclusão	Documentação de avaliação crítica com a conclusão
02/12/2022	04/12/2022	21. Construção do vídeo de apresentação da Etapa 3	Vídeo da etapa 3 disponível
14/12/2022	14/12/2022	22. Publicação no repositório Github Etapa 3	Arquivos produzidos no Github disponíveis abertamente
01/12/2022	01/12/2022	23. Publicação da plataforma na internet	Disponibilização da plataforma aberta na internet

3. Especificação Arquitetural da solução

Esta seção tem o objetivo de apresentar a especificação básica da arquitetura da plataforma que será desenvolvida, incluindo diagramas, restrição arquitetural, requisitos funcionais e requisitos não funcionais definidos pelo autor, permitindo entendimento e visualização da macroarquitetura da solução.

3.1 Restrições Arquiteturais

Os Requisitos Arquiteturais são todos os requisitos, sejam eles Funcionais ou Não-Funcionais que têm impacto direto sobre a Arquitetura do Sistema. Dessa forma, o Arquiteto precisa analisar os requisitos do sistema identificando algumas propriedades e então “filtrando” os Requisitos Arquiteturais. A lista a seguir apresenta os requisitos arquiteturais que foram identificados para implementação inicial da plataforma Pitaco FC.

ID	Descrição
RA01	O front-end deve utilizar como tecnologia o framework Angular.
RA02	O back-end deve ser construído utilizando a linguagem Java junto ao framework Spring.
RA03	O sistema de gerenciamento de banco de dados deve ser o MySQL.

RA04	Deve-se utilizar o padrão arquitetural REST para realizar a integração entre o front-end e o back-end, realizando comunicação através do protocolo HTTP, com mensagens no formato Json.
RA05	O padrão JWT deve ser utilizado para trafegar informações do usuário logado, assim verificando a sua permissão para acessar os diversos serviços.
RA06	O sistema deve permitir o cadastro de usuários pela plataforma e também com o uso de login sociais como Google e Fcebook..
RA07	O sistema deve possuir interface responsiva, permitindo que usuários de diferentes dispositivos acessem a plataforma através de navegadores de internet.
RA08	O sistema deve ser resiliente/tolerante a falhas, utilizando o padrão request/replay em comunicações transacionais.
RA09	A atualização do placar e pontuação deve ocorrer de forma paralela aos serviços on-line, utilizando-se para isso web crawlers para atualização de dados.

3.2 Requisitos Funcionais

Os Requisitos Funcionais podem ser definidos como um mapeamento das necessidades, desejos e solicitações dos usuários que requerem um software, o seu correto levantamento são imprescindíveis, pois com eles os desenvolvedores/analistas conseguem realizar a construção do software de forma assertiva empregando seu esforço no que realmente vai atender as necessidades do seu cliente final. Dito isso, requisitos funcionais o que sistema deve fazer. A lista a seguir apresenta os requisitos funcionais identificados para o desenvolvimento inicial da plataforma.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	A plataforma deve permitir o cadastramento dos usuários na plataforma e também através de logins sociais, inicialmente Facebook e Google.	A	A
RF02	A plataforma deve permitir que usuários realizem a recuperação de senha e sua redefinição.	M	A
RF03	Os usuários devem identificar-se com as suas credenciais de acesso previamente cadastradas, ou por meio dos botões de redes sociais disponibilizados para login.	M	A
RF04	A plataforma deve permitir que o usuário possa se inscrever e cancelar a inscrição nos campeonatos disponíveis.	M	A
RF05	A plataforma deve permitir que o usuário visualize todas as partidas cadastradas de um determinado campeonato.	A	A

RF06	A plataforma deve permitir que os usuários visualizem a classificação geral das equipes de um determinado campeonato.	A	A
RF07	A plataforma deve permitir o usuário consultar o histórico de jogos de uma equipe dentro de um determinado campeonato.	M	A
RF08	A plataforma deve permitir que o usuário visualize todos os jogos do dia corrente na página inicial.	A	A
RF09	A plataforma deve permitir que o usuário realize os palpites nas partidas dos campeonatos em que esteja inscrito.	A	A
RF10	A plataforma deve permitir que os usuários possam consultar detalhes das apostas realizadas em uma partida.	A	A
RF11	A plataforma deve permitir que que todos os usuário possam visualizar os apostadores de cada placar.	M	A
RF12	A plataforma deve permitir que os usuários visualizem rankings gamificados.	A	M
RF13	A plataforma deve fornecer feedback para os palpiteiros, listando os pontuadores de cada rodada, no dia posterior a rodada.	B	B
RF14	A plataforma deve permitir que o usuário possa visualizar detalhes de outros usuários.	M	M
RF15	A plataforma deve permitir que o usuário indique campeonatos que gostaria que fossem incluídos no jogo.	B	B
RF16	A plataforma deve permitir que o usuário autenticado visualize seus dados e edite-os quando julgar necessário.	B	M
RF17	A plataforma deve permitir que o usuário completo seu cadastro.	B	M
RF18	A plataforma deve permitir a edição do avatar, disponibilizando avatares pré definidos para seleção.	M	B
RF19	A plataforma deve permitir que o usuário realize a edição de sua senha sempre que julgar necessário.	A	A
RF20	A plataforma deve realizar o envio de comunicações de marketing para seus usuários através de e-mails.	M	B
RF21	A plataforma deve permitir que os usuários criem grupos para um determinado campeonato, podendo ser público ou privado.	A	A
RF22	A plataforma deve permitir que os usuários visualizem grupos ativos e possam realizar a sua inscrição.	M	M
RF23	A plataforma deve permitir que os usuários visualizem detalhes do grupo que participa.	B	M
RF24	A plataforma deve permitir que os usuários convidem seus amigos para participar do jogo e se inscreverem na plataforma.	M	B
RF25	A plataforma deve permitir que usuários administradores de grupo realizem a edição dos dados do grupo.	M	B

RF26	A plataforma deve permitir que usuários administradores de grupos privados aceitem ou rejeitem solicitações de novos membros.	A	M
RF27	As partidas devem ser atualizadas automaticamente, possibilitando o acompanhamento do placar em tempo real.	A	A

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

Os Requisitos Não-Funcionais estão associados às restrições de funcionalidades que ditam como o sistema deve fazer. A lista a seguir apresenta os requisitos não funcionais identificados para o desenvolvimento inicial da plataforma.

ID	Descrição	Prioridade B/M/A
RNF01	A plataforma deve operar e estar disponível para acesso todos os dias e permitir ser acessada por navegadores web e mobile.	A
RNF02	A plataforma deve otimizar o uso da rede, realizando comunicação (front/back) apenas quando necessário.	M
RNF03	A plataforma deve permitir o acompanhamento dos resultados em tempo real, atualizando placar e tempo das partidas de forma automatizada sempre que estiver em período com partidas.	A
RNF04	O front-end deve ser desenvolvido utilizando o modelo MVC (Model-View-Controller).	M
RNF05	A plataforma deve ser segura, restringindo a exibição de dados apenas para usuários autenticados e com sessão válida.	A
RNF6	As notificações por e-mail devem ser enviadas por mecanismos que não impactem os acessos on-line, podendo ser enviadas por meio de Jobs/Scheduled ou filas.	A
RNF7	A atualização de dados do jogos deve acontecer de forma automática, e paralela para não afetar os acessos on-line	A
RNF8	As consultas de ranking, mais onerosas, devem utilizar cache e serem atualizados sempre que os placares forem alterados.	A
RNF9	O acesso ao banco de dados deve ser realizados apenas por repositories, não permitindo o acesso direto de usuários a base de dados.	A
RNF10	A plataforma deve permitir o monitoramento dos microserviços.	M

*B=Baixa, M=Média, A=Alta.

3.4 Mecanismos Arquiteturais

Os mecanismos arquiteturais representam conceitos técnicos fundamentais que serão padronizados por toda a solução. Eles são refinados durante o projeto em três estados, representados pelas três categorias de Mecanismos Arquiteturais:

- Mecanismo de Análise, que dá ao mecanismo um nome, uma descrição resumida e alguns atributos básicos derivados dos requisitos do projeto.
- Mecanismo de Design, que são mais concretos e assumem alguns detalhes do ambiente de implementação.
- Mecanismo de Implementação, que especifica a exata implementação de cada mecanismo.

Análise	***Design***	**Implementação**
Persistência	ORM	Hibernate com utilização do Spring Data JPA
Persistência	SGBD	MySQL
Front end	SPA – Single Page Application	Angular/TypeScript
Back end	Microserviço	Java 11
Back end	Gateway	Spring Cloud Gateway
Back end	Cloud Config	Spring Cloud Config Server
Autorização e Autenticação	JWT - Json Web Token	Utilização de Json Web Token com Spring Security
Integração	Rest API	Protocolo HTTP com trafego de objeto JSON
Integração	Crawler	Web crawler com JSOUP (Globo Esporte)
Integração	Crawler	Web crawler com JSOUP (Google)
Integração	Login Social	Facebook e Google
Log	Rastreio de log	Log4J
Teste de Software	Testes unitários	JUnit
Dependência	Gerenciador de dependência	Maven
Usabilidade	Notificação por E-mail	Google Gmail
Deploy	War	Deploy do war em servidor web
Versionamento	Versionamento de código	Github

4. *Modelagem Arquitetural*

Esta seção apresenta a modelagem arquitetural da aplicação proposta, permitindo de forma simples o entendimento do modelo de negócio utilizado e visando à implementação da Prova de Conceito (PoC) da plataforma Pitaco FC na seção 5.

O modelo C4 (Contexto, Containeres, Componentes e Código), utilizado para modelagem arquitetural, possui 4 níveis de diagramas hierárquicos, e que nos possibilita realizar a descrição da arquitetura de um software em diferentes níveis. O diagrama de contexto permite visualizar o software e sua interação com pessoas e integração com outros sistemas. O diagrama de container amplia a visibilidade do software, exemplificando as integrações entre as camadas, banco de dados e outros serviços. O diagrama de componentes amplia a visualização do container afim de mostrar de forma individual abstrações e agrupamentos de código. Por fim, o código, um diagrama onde mostra classes e seus relacionamentos com diversas faces do sistema.

4.1 *Diagrama de Contexto*

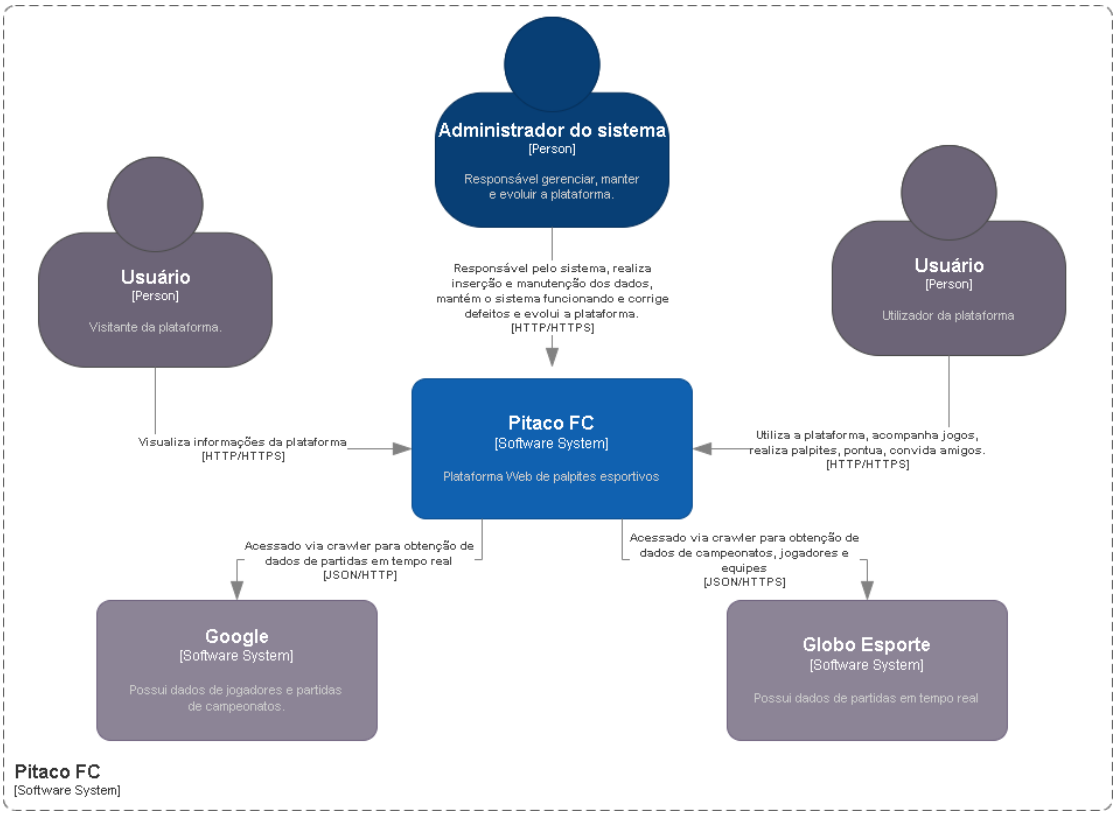

Figura 1 - Visão Geral da Plataforma Pitaco FC.

A figura 1 mostra a visão geral da solução, de forma a exemplificar o fluxo de integração entre o sistema web (microserviços e crawler) e os sistemas externos, além de personificar os principais tipos de usuários envolvidos no sistema de palpites. É importante ressaltar que a integração com as páginas do Google e Globo Esporte são utilizadas como fonte de dados em tempo real, onde o Google será consultado em períodos de jogos em andamento para obtenção do placar, jogadores que marcaram gols e tempo das partidas, atualizando dessa forma os dados em base de dados que será utilizado na consulta pela plataforma web, e Globo Esporte para obtenção dos dados atualizados referente a artilharia, partidas e seleções do campeonato mundial.

4.2 Diagrama de Container

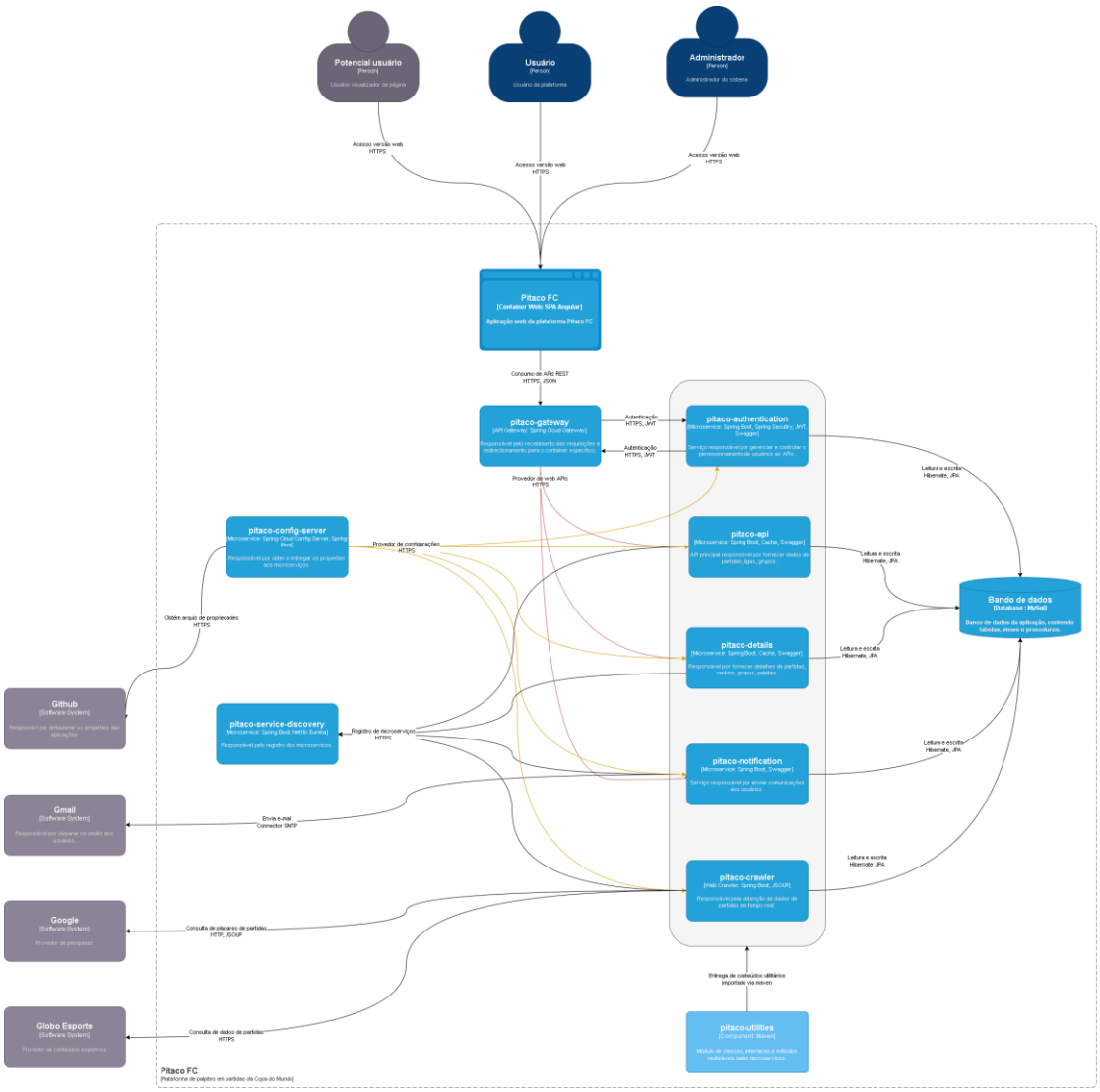

Figura 2 – Diagrama de contêineres da Plataforma Pitaco FC.

A Figura 2 apresenta o detalhamento dos contêineres que compõem a plataforma Pitaco FC, ilustrando a distribuição e comunicação entre os diversos serviços distribuídos e suas integrações internas e externas.

A plataforma possuí 3 tipos diferentes de usuários, o usuário que acessa as páginas estáticas do site e não possuí registro, o usuário registrado e ativo que participa dos palpites e acompanha as partidas em tempo real e também o administrador do sistema que executa manutenções e evoluções na plataforma. A plataforma pode ser acessada pelos usuários por meio de navegador web em diferentes dispositivos (celular, tablete, desktop) por ser responsiva e adaptativa, desenvolvida com a utilização do framework Angular.

A comunicação do front-end com o back-end é feita por um ponto central, o pitaco-gateway que intercepta as requisições e as encaminha para os microserviços responsáveis de cada funcionalidade por meio de comunicação HTTPS e trafego de JSON. O processo de autenticação e autorização é realiado em um módulo senpre acessível pelo gateway afim de verificar a validade da sessão do usuário em acesso.

O pitaco-service-config é responsável por obter os arquivos de configuração (.properties) de um repositório no github e entrega-los a cada um dos microserviços da arquitetura, o que permite que cada microserviço tenha seu arquivo especifico e também um arquivo de configurações compartilhadas. A utilização do service-discovery permite que os microserviços façam o seu registro e permite ao administrador monitorar os serviços up/down.

O módulo de compartilhado via dependências permite a reutilização de código, classes, métodos e interfaces, maximizando a eficiência de manutenção em um ponto central e único com reflexo para todos os seus utilizadores.

A realização da atualização dos dados de partidas é feita por um crawler, que é responsável por capturar dados em tempo real de páginas do Google utilizando raspagem web para capturar as informações de tempo, placar e nomes dos marcadores dos gols, e obter dados de partidas e equipes para inserção em base de dados do Globo esporte via requisições HTTPS consumindo JSONs. O crawler é um microserviço inacessível por usuários que atua na obtenção das informações e atualização em base de dados (MySql), e os microserviços realizam as consultas na base de dados para exibição para os usuários na aplicação web. O banco de dado possuí rotinas (procedures) para atualização de pontuação de acordo com o avanço das partidas e mudanças nos placares de forma a refletir em tempo real os rankings para os usuários.

O envio de comunicações é feito por um módulo único, responsável por disparar as solicitações de envio de e-mail para o Gmail, por meio de conector SMTP.

4.3 Diagrama de Componentes

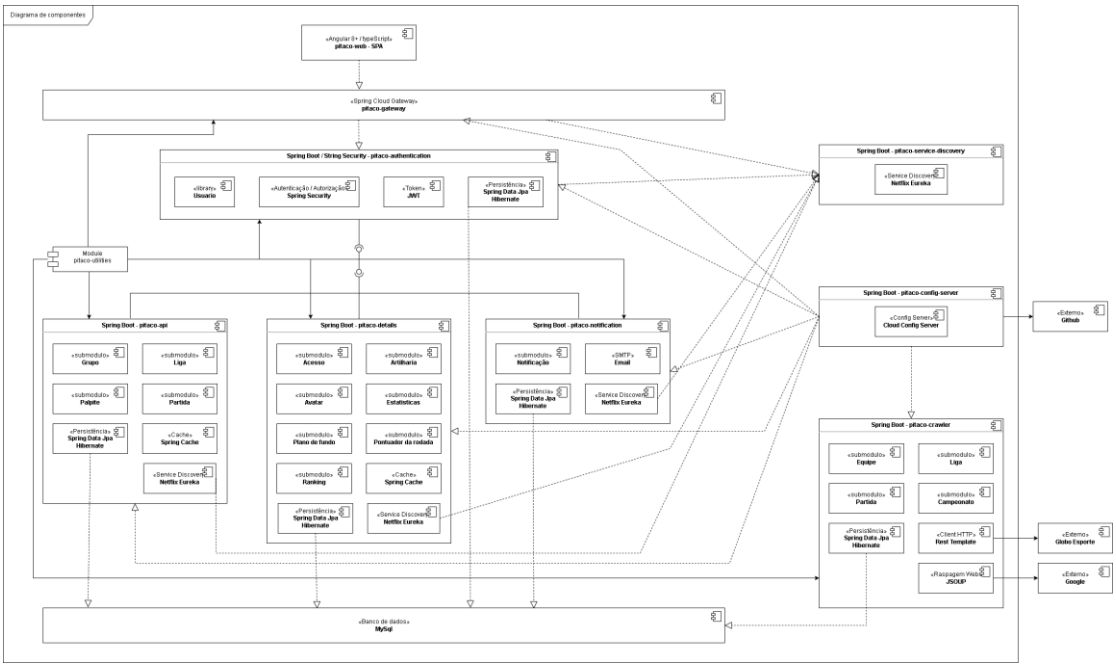

Figura 3 – Diagrama de componentes Pitaco FC.

A figura 3 mostra o diagrama de componentes do projeto Pitaco FC, permitindo através dele, visualizar os pacotes e tecnologias da plataforma. Abaixo um detalhamento dos componentes da plataforma:

Cliente web SPA: Aplicação desenvolvida utilizando o framework Angular, responsivo e adaptativo para diferentes navegadores e plataformas (desktop e mobile). Permite a autenticação e cadastro de usuários por meio de logins sociais como o Google e Facebook. Sua comunicação com o back end se dá por meio de http(s) com o pitaco-gateway sendo necessário estar autenticado para acessar os serviços e funcionalidades da plataforma.

Pitaco-gateway: Desenvolvido utilizando framework Spring Cloud Gateway, esta aplicação é responsável por receber as requisições do front-end e redirecionar para o serviço responsável pela funcionalidade solicitada.

Pitaco-authentication: Aplicação responsável pelo cadastro, autenticação e autorização de usuários, utiliza o Spring Security para controle de acesso, com trafego de Json Web Token validos por um período determinado de tempo. Toda comunicação com o back-end necessita que o usuário esteja autenticado e tenha autorização de acesso para a funcionalidade.

Pitaco-api: Responsável pelas regras de negócio das APIs de grupos, ligas, palpites e partidas, listando, inserindo e atualizando as informações dessas entidades.

Pitaco-details: responsável por fornecer detalhes das informações armazenadas e agrupadas como rankings, artilharia, pontuador da rodada, controle de acesso, plano de fundo, utilizando-se de cache para maximizar o tempo de resposta para os clientes assim como também o processamento.

Pitaco-notification: Serviço responsável por disparar comunicações para os usuários via e-mail.

Pitaco-service-discovery: Responsável atuar como um servidor de registro dos demais serviços, utiliza-se da biblioteca Netflix Eureka, o que permite uma fácil visualização dos serviços e sua situação (up/down), fornecendo um painel de acesso rápido para acompanhamento.

Pitaco-service-config: Responsável por entregar para os demais serviços as configurações (.properties) necessárias para execução, atua literalmente como um servidor de configurações, obtendo os arquivos do github e entregando-os para que solicitar.

Pitaco-crawler: Serviço responsável por inserir e atualizar informações de partidas, campeonatos/ligas, artilharia, etc, nas bases de dados que são consultadas pelos outros serviços. Possui duas funcionalidades essenciais, obtendo registros de campeonatos, partidas e jogadores da API do Globo Esporte, e atualizando as partidas

(tempo, resultado, gols, artilheiros, etc) obtendo informações em tempo real de páginas web como o Google por meio de raspagem web.

5. Prova de Conceito (PoC)

Nessa sessão, será aprofundado o detalhamento do projeto desenvolvido a prova conceitual, validando a arquitetura proposta e funcionalidades levantadas nos requisitos funcionais e não funcionais com o objetivo de disponibilizar o sistema desenvolvido na web. Para realização de testes acesse o endereço abaixo com usuários e senha informados:

- **Endereço**: https://pitacofc.com.br/
- **Usuário**: pucminas@pitacofc.com.br
- **Senha**: 123qwe

Código da aplicação repositório git:

- https://github.com/phcruz/puc-minas

5.1 Integrações entre Componentes

Para demonstrar a aplicação desenvolvida, foi elencada 3 funcionalidades, as quais serão apresentadas e evidenciadas as imagens da aplicação que atendem aos requisitos funcionais elencados.

RFO8 – Permitir que o usuário visualize os jogos do dia corrente na página inicial.

Ao realizar o acesso a aplicação o usuário pode escolher entre se cadastrar e entrar com seu e-mail cadastrado ou realizar seu login utilizando redes sociais.

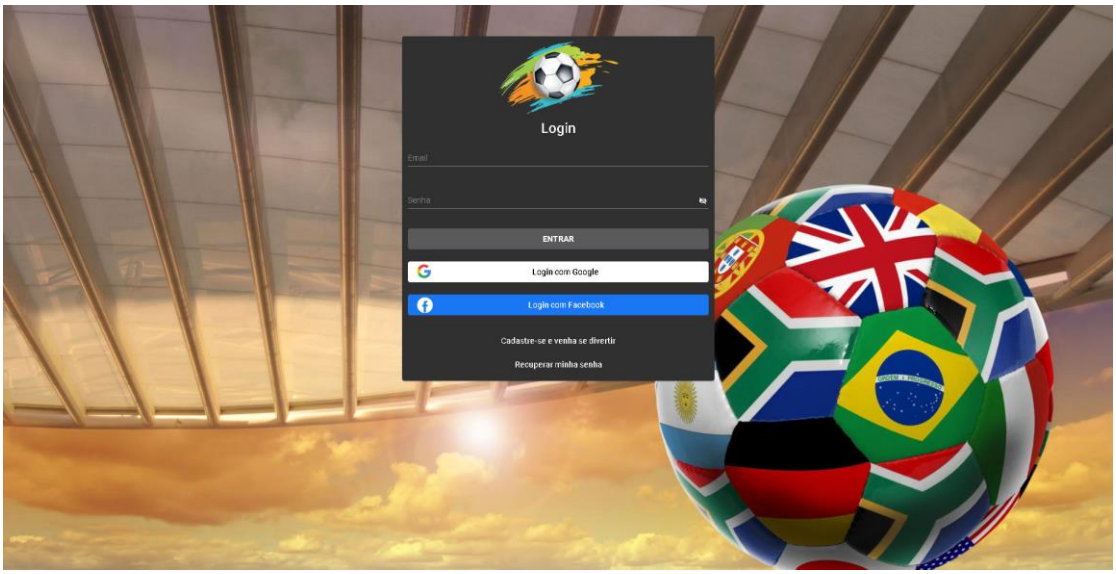

Figura 4 – Home da aplicação.

Ao acessar a aplicação o usuário cairá Home, onde é exibido todos os jogos que serão realizados na data atual.

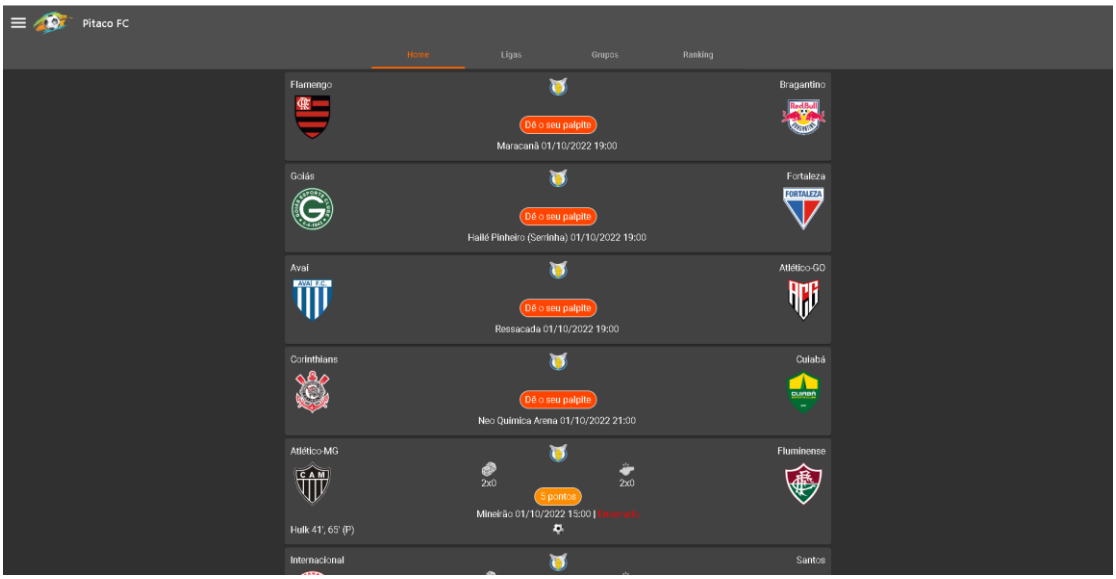

Figura 5 – Home da aplicação.

A estrutura da tela é adaptável para diferentes tamanhos de dispositivos, permitindo a sua utilização em smartphones, tablets e desktops. Cada card da aplicação tem um comportamento diferente de acordo com o palpite, horário antes, durante e após o jogo. A figura 5 mostra os 4 primeiros itens sem palpite e exibição antes do início do jogo e inclusive antes do período de fechamento das apostas (30 minutos antes do início da partida).

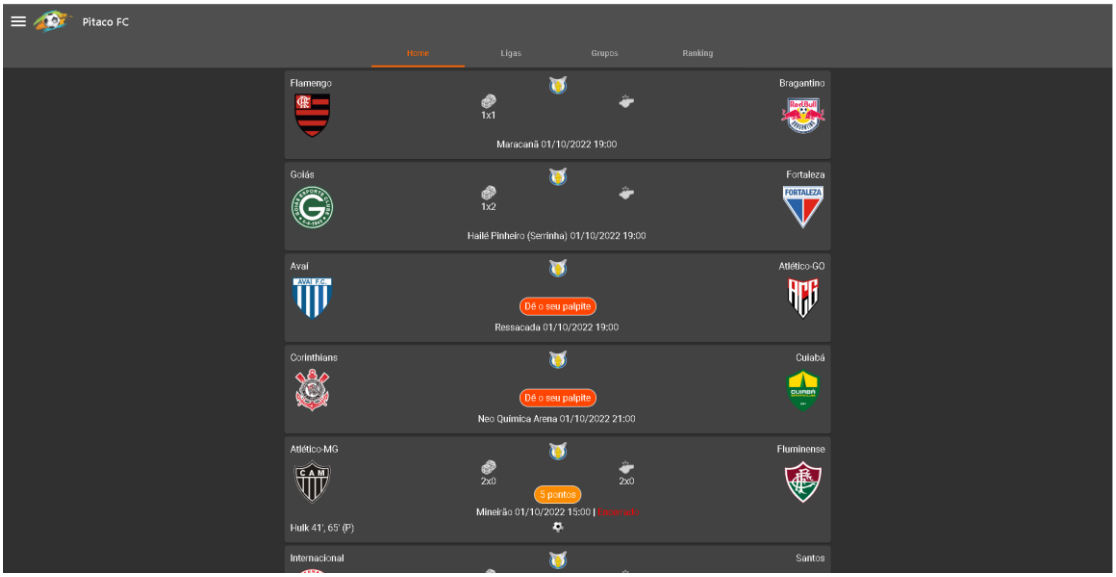

Figura 6 – Home da aplicação com palpite antes do fechamento de palpites.

A figura 6, exemplifica nos dois primeiros cards, a exibição de itens com palpites realizados antes do fechamento de palpites, ainda sem a pontuação iniciada.

Ao inicializar a partida, a configuração dos cards muda novamente, e passa a exibir as informações do placar real da partida, o tempo atual, além dos jogadores que marcam gols como pode ser visualizado na figura 7 logo abaixo.

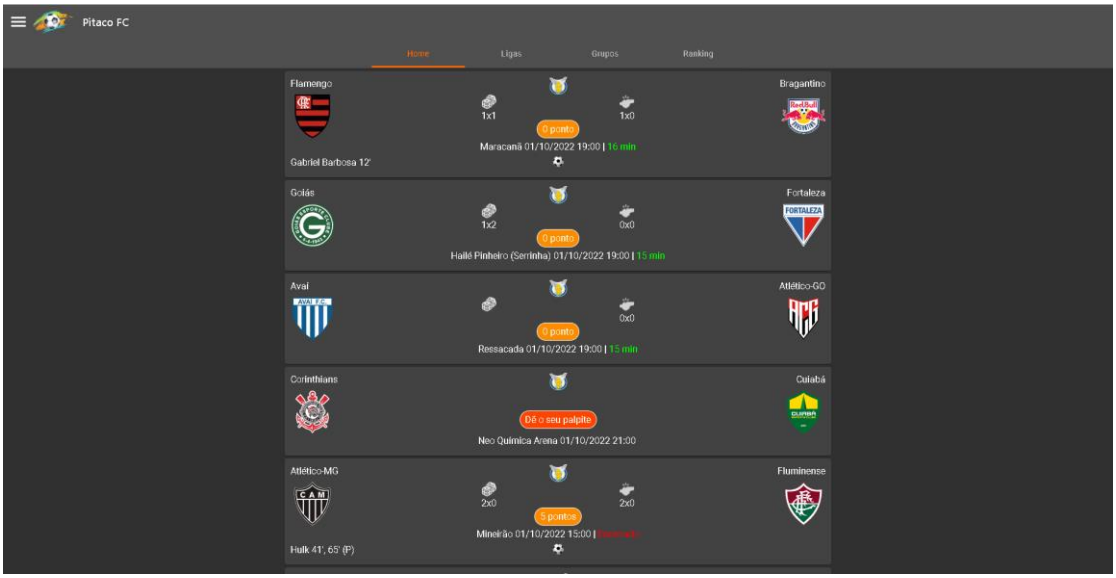

Figura 7 – Home da aplicação com palpite e partidas em andamento.

A pontuação também é atualizada dinamicamente a cada alteração de placar, ou seja, dessa forma a cada gol o usuário pode ganhar ou perder pontos de acordo com o seu palpite nas partidas. Importante ressaltar que após a partida ser iniciada, acontece o start de um timer no front que atualiza a cada 30 segundos a tela, sem a necessidade de

interação do usuário, atualizando dinamicamente toda a estrutura de informações presentes na tela conforme mostra a tela 8.

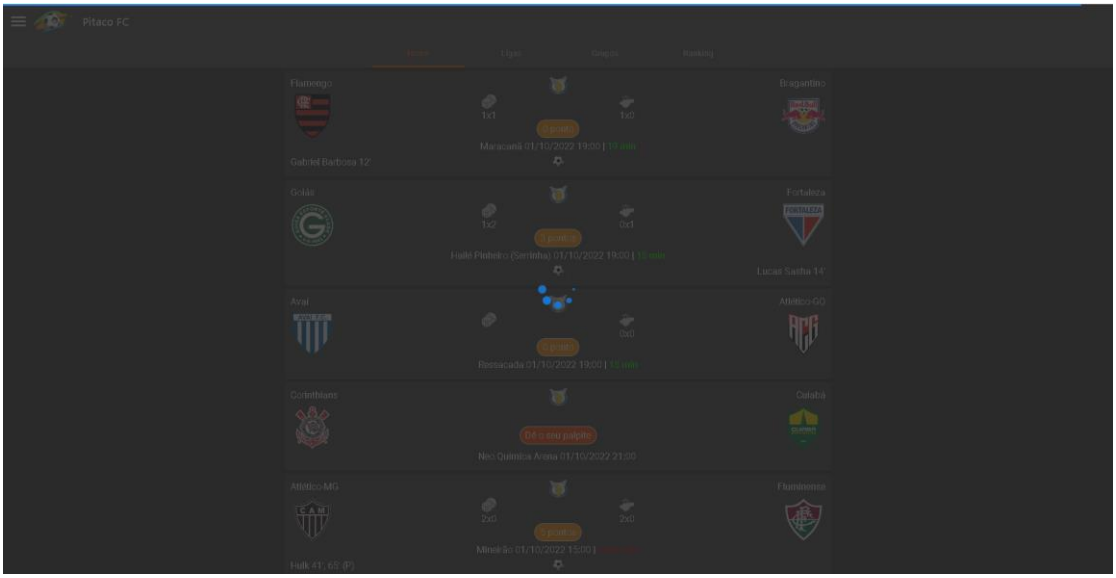

Figura 8 – Home da aplicação com palpite e partidas em andamento.

O fluxo de atualização das informações acontece todo em back-end, com a atualização do pitaco-crawler, que executa em períodos de jogos, obtendo as informações das equipes e partida e com essas informações recuperadas da base, varre o Google em busca de dados atualizados da partida, tornando dinâmico e automatizado o processo de atualização das informações, permitindo assim que os usuários consultem informações atualizadas em nossa base de dados, refletindo o tempo da partida, o placar, os jogadores que realizaram os gols, além de atualizar a pontuação a cada alteração de placar encontrado.

RF09 – Permitir que o usuário palpite nas partidas.

A funcionalidade contempla inserção e atualização de palpites, não sendo permitido a remoção dos palpites. Possui ainda validações de horário, restringindo os palpites nas partidas a 30 minutos antes do seu início, retornando um erro tratado em caso ocorra alguma tentativa pós esse período. Para realizar um palpite na partida basta que o usuário clique no card da respectiva partida, que será redirecionado para a tela de realização de palpite, como mostra a figura 9.

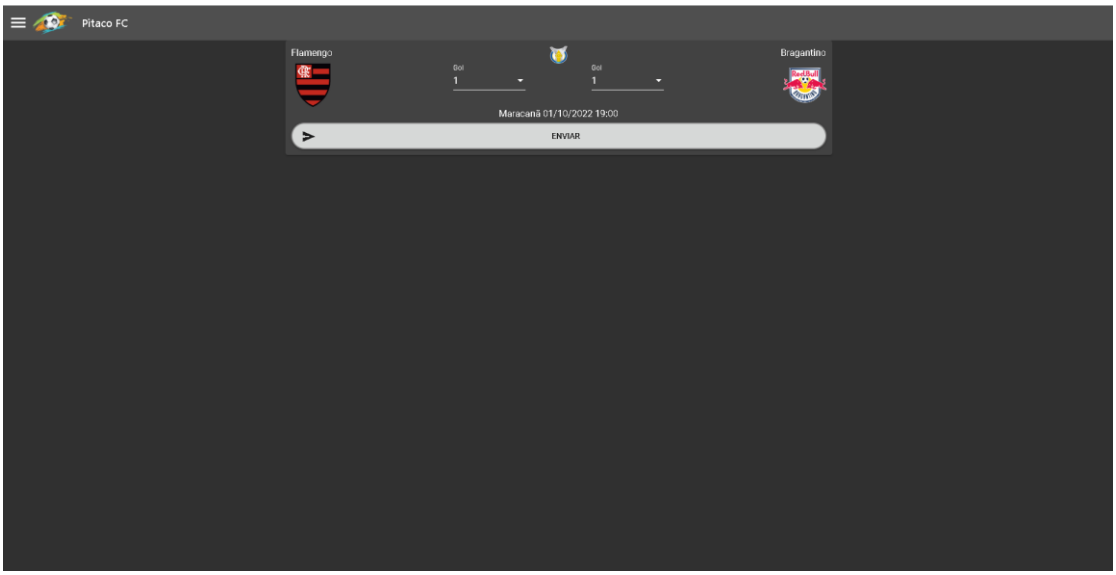

Figura 9 – Tela de palpites em partida.

Essa tela também possui comportamentos para diferentes cenários, a imagem acima permite a inserção do palpite pelo cliente, caso o usuário clique no card depois do fechamento dos palpites os campos são desabilitados, exibindo o palpite, caso exista, como exibe a figura 10.

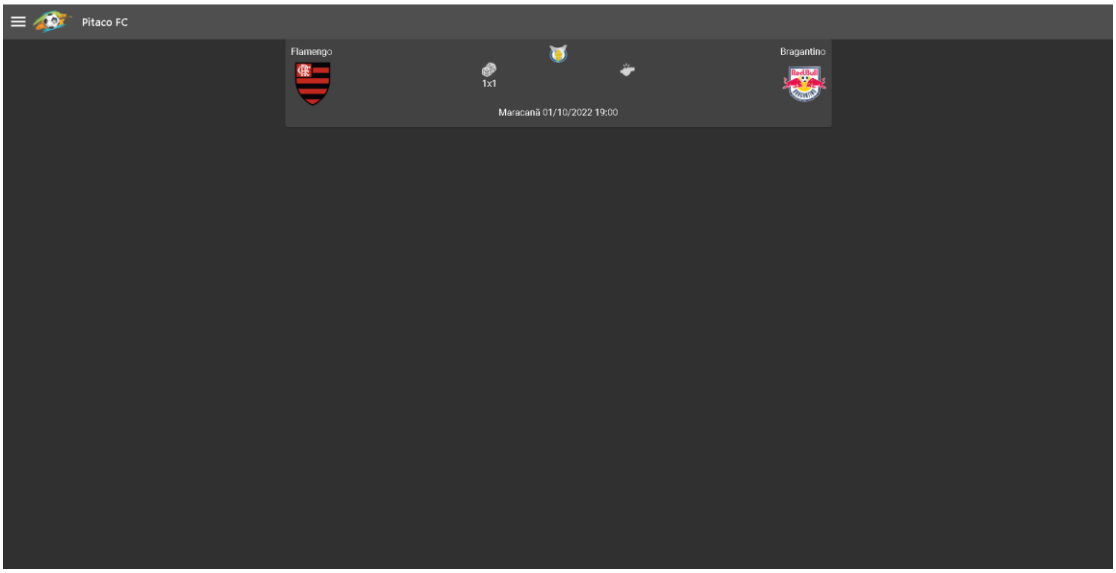

Figura 10 – Tela de palpites em partida pós fechamento de palpites.

Um terceiro comportamento da tela de palpites, torna visível a quantidade de palpites agrupados por placares em um gráfico de barras, além de um gráfico de pizza representando a quantidade de palpites na vitória de cada equipe e também do empate entre eles, conforma mostra a imagem 11.

Figura 11 – Tela de palpites em partida pós início das partidas.

RF12 – Visualizar rankings gamificados.

Afim de estimular a competição entre os usuários a aplicação exibe rankings diferenciados por campeonato, exibindo os detalhes de cada jogador, como avatar, total de pontos, quantidade de partidas com 5 pontos (quando o usuário acerta o placar exato), 3 pontos (quando o usuário acerta a equipe vencedora com a diferença do placar), 1 ponto (quando o usuário acerta a equipe vencedora) e 0 pontos (quando o usuário aposta no time perdedor). A figura 12 mostra a tela de ranking logo abaixo.

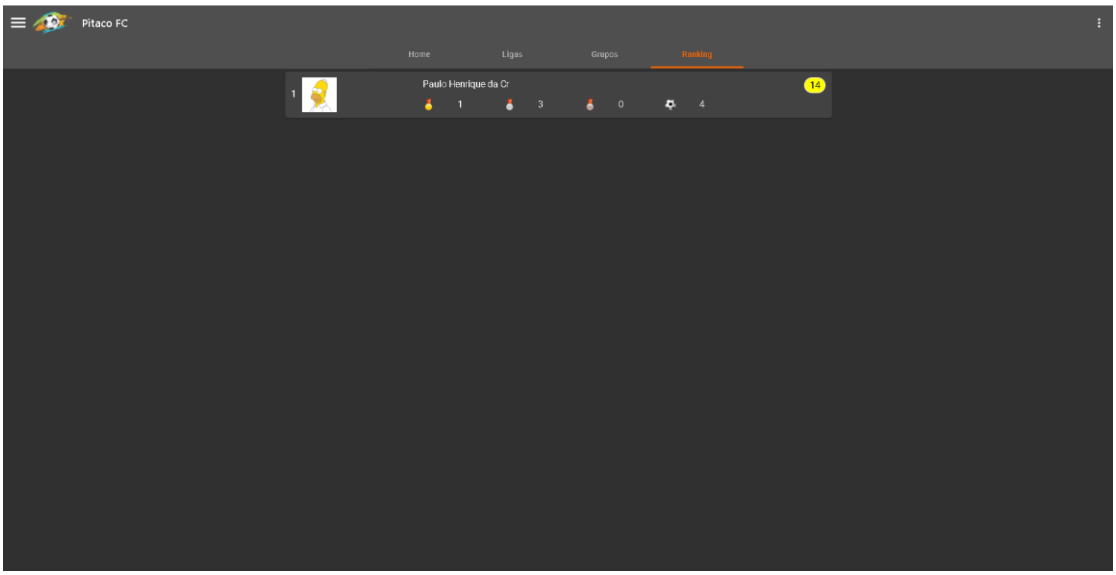

Figura 12 – Tela de palpites em partida pós início das partidas.

Todos os usuários que possuem um palpite são listados no ranking, independente de terem pontuado ou não, caso o usuário ainda não tenha realizado nenhum palpite ele não é exibido nesta tela.

Afim de tornar o processo transparente, todos os usuários listados no ranking permitem clique em seu card, abrindo uma tela com os detalhes do usuário, onde é possível visualizar as informações de quais grupos ele participa, e todo o seu histórico de partidas palpitadas e sua respectiva pontuação em cada uma, confirme é apresentado na figura 13.

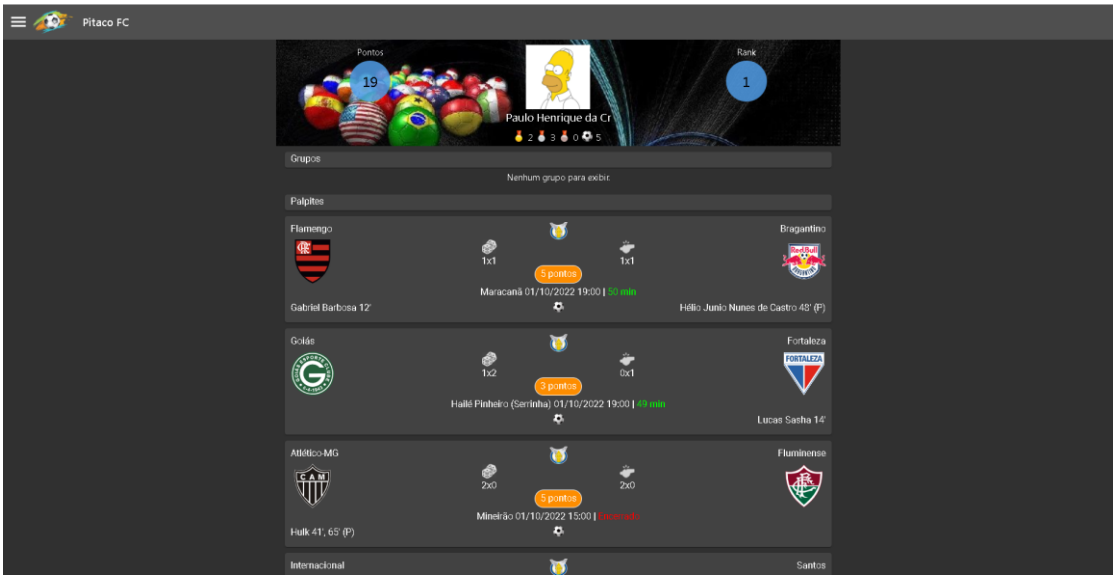

Figura 13 – Tela de detalhes do ranking do usuário.

5.2 Código da Aplicação

A sessão a seguir exemplifica a nível de código, a estrutura da aplicação representando 3 requisitos funcionais já citados na respectiva sessão 3.2. O código completo da aplicação está disponível no github e pode ser acessado pelo endereço abaixo:

Github: https://github.com/phcruz/puc-minas

A aplicação também está disponível para execução de testes no endereço https://pitacofc.com.br/, disponibilizada para realização de testes com os dados de usuário:

- **Usuário**: pucminas@pitacofc.com.br
- **Senha**: 123qwe

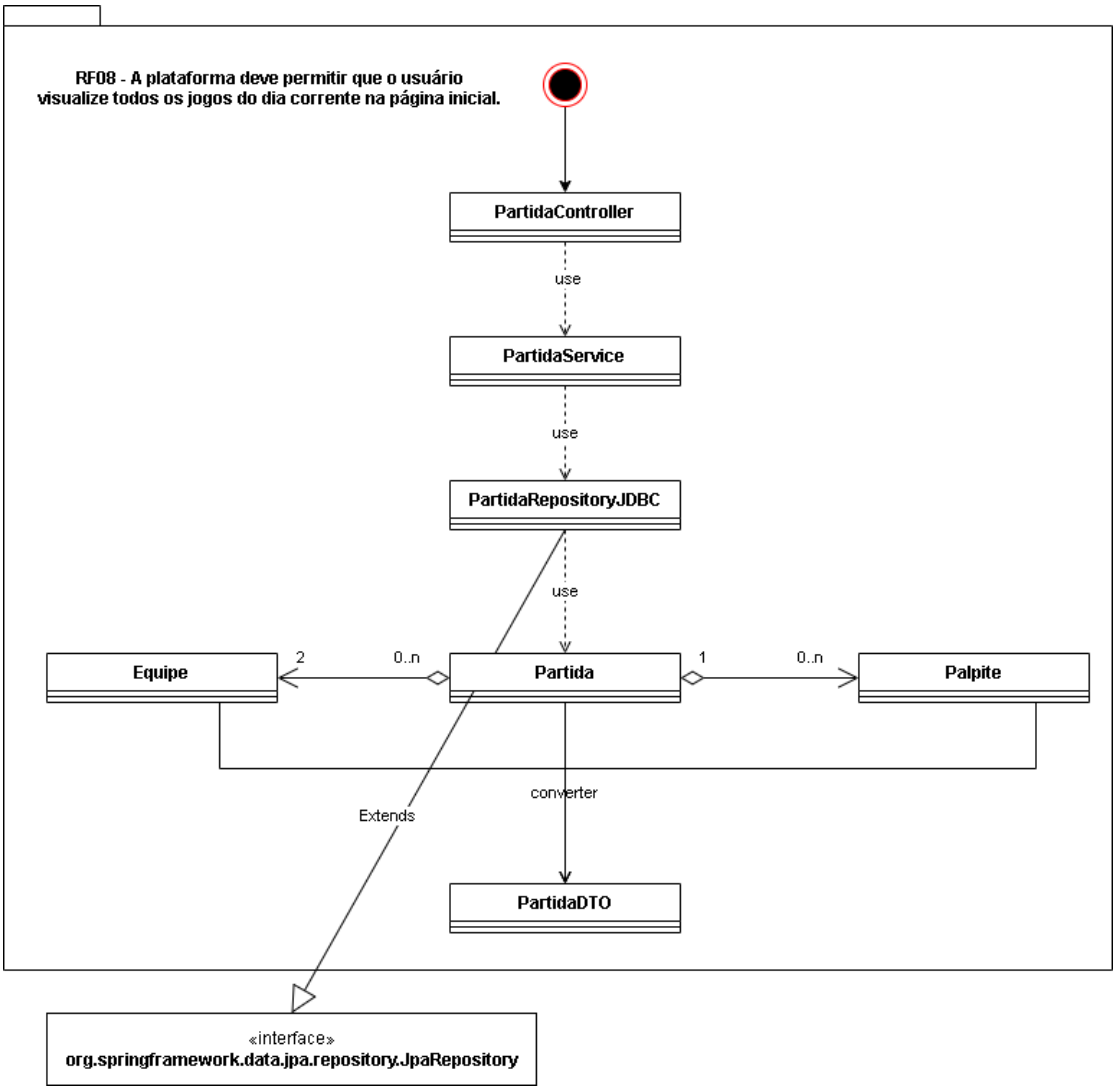

Figura 14 – RFO8 – Permitir que o usuário visualize os jogos do dia corrente na página inicial.

A figura 14, exibe o funcionamento do RF08, onde a solicitação recebida do front-end é interceptada pela classe PartidaController. Na classe Partida service ocorre a injeção de dependencia do repositorio de partida, PartidaRepositoryJDBC, que realiza a consulta das informações das partidas do corrente dia, de acordo com o usuário logado, filtrando por campeonatos que o mesmo participa, realizando joins com as tabelas de equipe e palpite, com o objetivo de montar o objeto de retorno que será convertido em um DTO para ser devolvido ao chamador do endpoint.

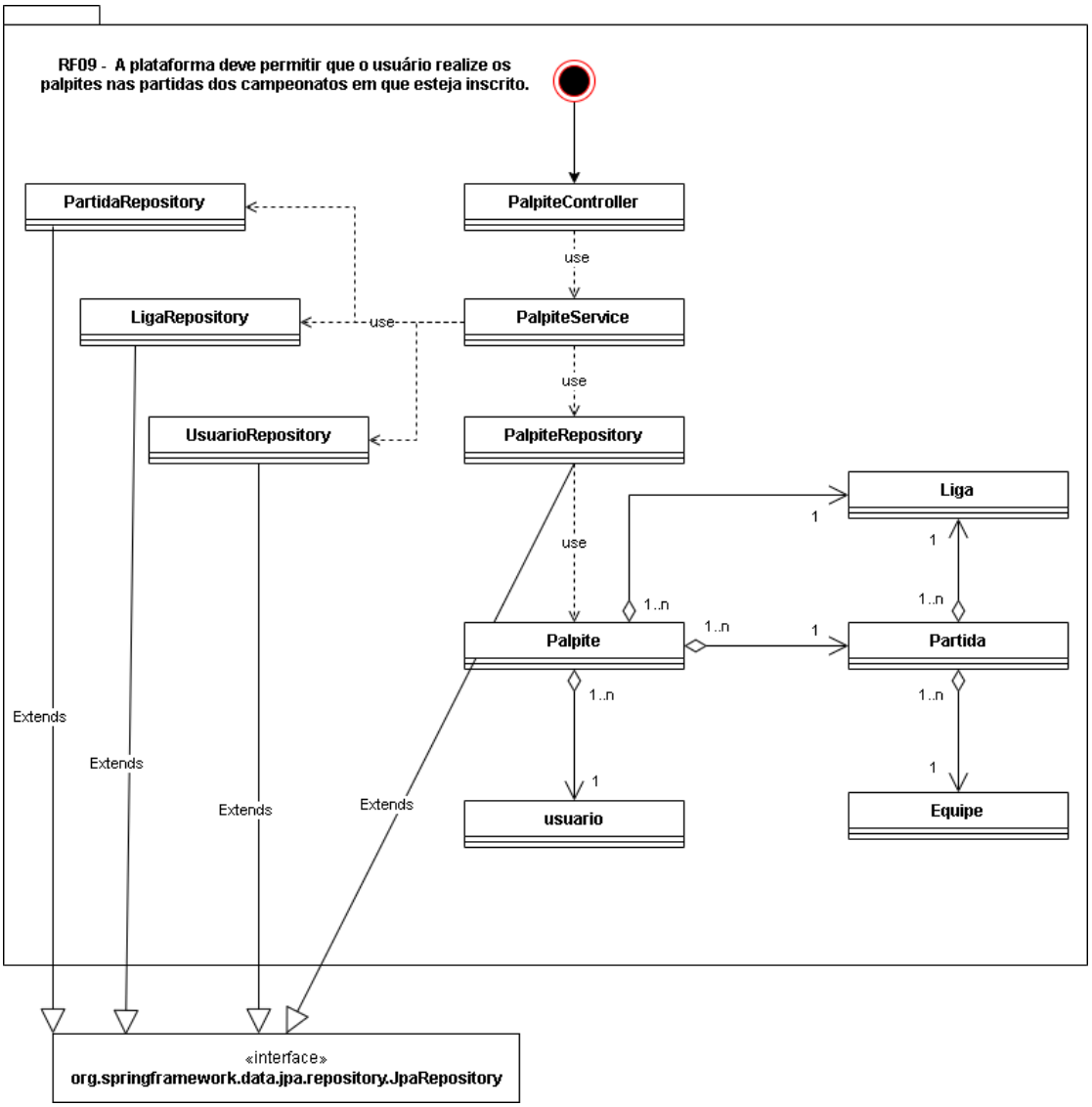

Figura 15 – RF09 – Permitir que o usuário palpite nas partidas.

O RF09, exibido na figura 15, exemplifica a funcionalidade de palpitar nas partidas. O usuário preenche os valores na tela e o front-end envia os dados para o serviço. A controller PalpiteController aplica a validação do período de abertura de apostas, período configurável que inicialmente é de 30 minutos antes da partida, permitindo ou não a criação alteração do palpite. Caso o tempo para início da partida seja inferior a 30 minutos, é retornado um erro validado para o front-end, caso ainda esteja em um período valido de apostas, a classe PalpiteService obtém informações complementares dos objetos de relacionamento para preencher a enteidade Palpite com as chaves primárias das tabelas usuário, liga e partida e assim definir a entidade que será persistida na base de dados utilizando a interface PalpiteRepository que faz uso da implementação do Spring Data JPA.

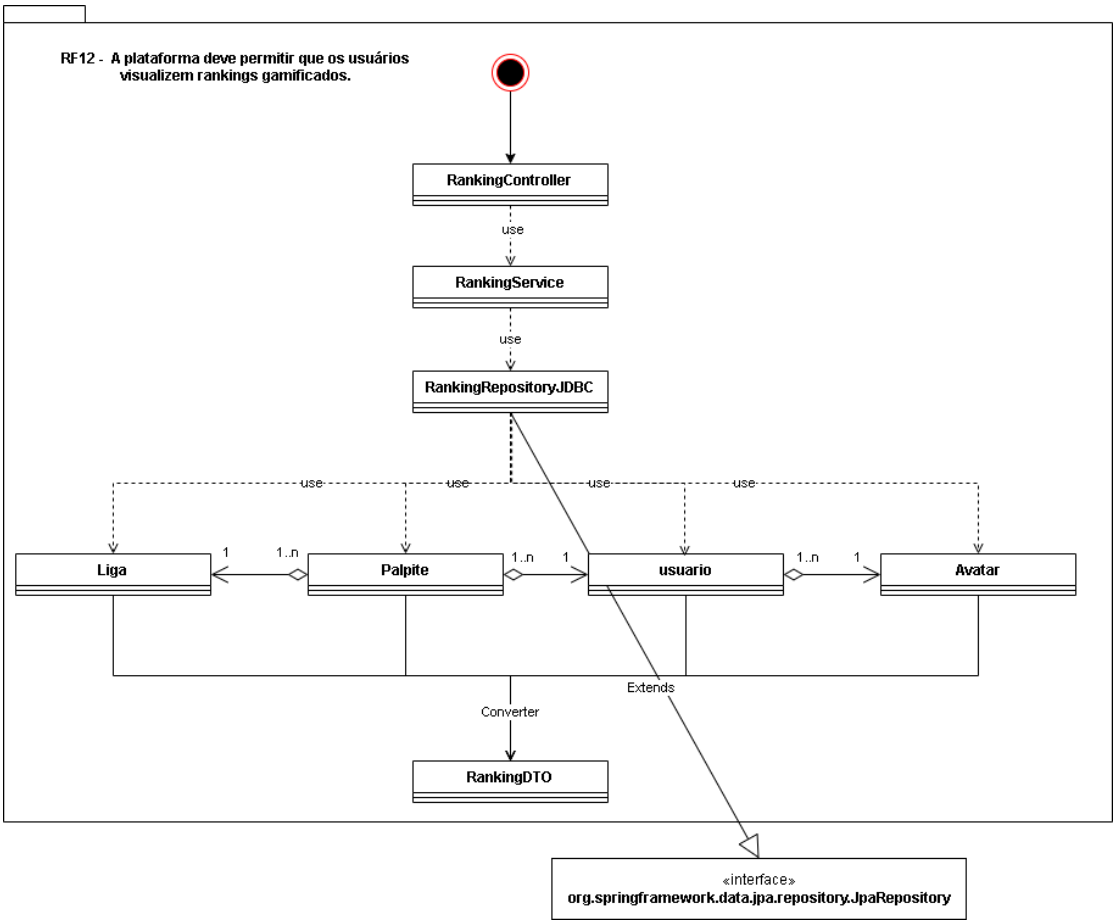

Figura 16 – RF12 – Visualizar rankings gamificados.

A figura 16 define o funcionamento do RF12, que permite aos usuários visualizarem os rankings gamificados de cada campeonato, assim como também o ranking global de palpites em partidas. Quanto mais partidas palpitadas e pontuadas melhor é a classificação do usuário no ranking. A classe RankingController utiliza a chamada para a servisse RankingService, injetada por dependência, e que é responsável por obter os dados da base através do RankingRepositoryJDBC. O repositor realiza a consulta com diversos joins entre as tabelas da base, afim de agrupar as informações de pontuação de cada usuário por liga, assim como também quantidade de partidas com cada pontuação (0, 1, ou 3 pontos). Após obter os dados é realizada a conversão em um DTO que será retornado para o front-end via http/json.

6. *Avaliação da Arquitetura (ATAM)*

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

6.1. Análise das abordagens arquiteturais

Atributos de Quailidade	Cenários	Importância	Complexidade
Acessibilidade	Cenário 1: A plataforma deve operar e estar disponível para acesso todos os dias e permitir ser acessada por navegadores web e mobile.	A	M
Usabilidade	Cenário 2: A plataforma deve permitir o acompanhamento dos resultados em tempo real, atualizando placar e tempo das partidas de forma automatizada sempre que estiver em período com partidas.	M	M
Usabilidade	Cenário 3: A atualização de dados do jogos deve acontecer de forma automática, e paralela para não afetar os acessos on-line.	M	M
Desempenho	Cenário 4: As consultas de ranking, mais onerosas, devem utilizar cache e serem atualizados sempre que os placares forem alterados.	A	A
Monitoramento	Cenário 5: A plataforma deve permitir o monitoramento dos microserviços.	M	M

6.2. Cenários

Cenário 1 - Acessibilidade: Acessar a plataforma de diferentes sistemas operacionais, diferentes browsers e dispositivos com diferentes tamanhos de telas. A adaptabilidade da aplicação deve ser aplicada em todos os cenários.

Cenário 2 - Usabilidade: A plataforma deve ser automatizada na busca e atualização de informações das partidas em andamento (tela Home), atualizando a tela sem a necessidade de o usuário realizar qualquer ação, tornando a experiência de acompanhamento em tempo real ainda mais perceptível pelos usuários da plataforma.

Cenário 3 - Usabilidade: As atualizações dos dados dos jogos em andamento devem ocorrer em serviço apartado dos serviços de acesso online pelo usuário, sendo um processo totalmente apartado do fluxo do aplicativo, sendo imperceptível qualquer concorrência entre as requests dos usuários com as atualizações e processamento de informações de update.

Cenário 4 - Desempenho: Para maximizar a performance de acesso as funcionalidades, a plataforma deve ser capaz de cachear as funcionalidades mais onerosas ao banco de dados, o que permite uma resposta mais rápida para o usuário, sem que o mesmo tenha sua navegabilidade afetada por lentidão de consultas ao backend.

Cenário 5 - Monitoramento: A plataforma deve ser monitorada, de forma simples, fácil e intuitiva, permitindo a visualização dos serviços em execução, consumo de memória, CPF, controle de requests, erros e traces de requisições, através de gráficos e alertas visuais.

6.3. Evidências da Avaliação

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A plataforma deve operar e estar disponível para acesso todos os dias e permitir ser acessada por navegadores web e mobile.
Preocupação:	
Para atingir o máximo de usuários, a plataforma deve ser responsiva, se adaptando a diferentes dispositivos e tamanho de telas, inclusive mantendo a compatibilidade com diferentes navegadores e sistemas operacionais.	

Cenário(s):	
Cenário 1	
Ambiente:	
Navegador Google Chrome, celular com sistema operacional Android e iOS.	
Estímulo:	
Utilização do sistema em diferentes dispositivos e tamanhos de tela.	
Mecanismo:	
Angular, Angular Material Design, HTML, CSS	
Medida de resposta:	
Adaptação do sistema web em diferentes tamanhos de telas e navegadores.	
Considerações sobre a arquitetura:	
Riscos:	Navegadores antigos com baixa ou nenhuma compatibilidade de componentes responsivos.
Pontos de Sensibilidade:	Quebras de compatibilidade navegador e bibliotecas.
Tradeoff:	Não há

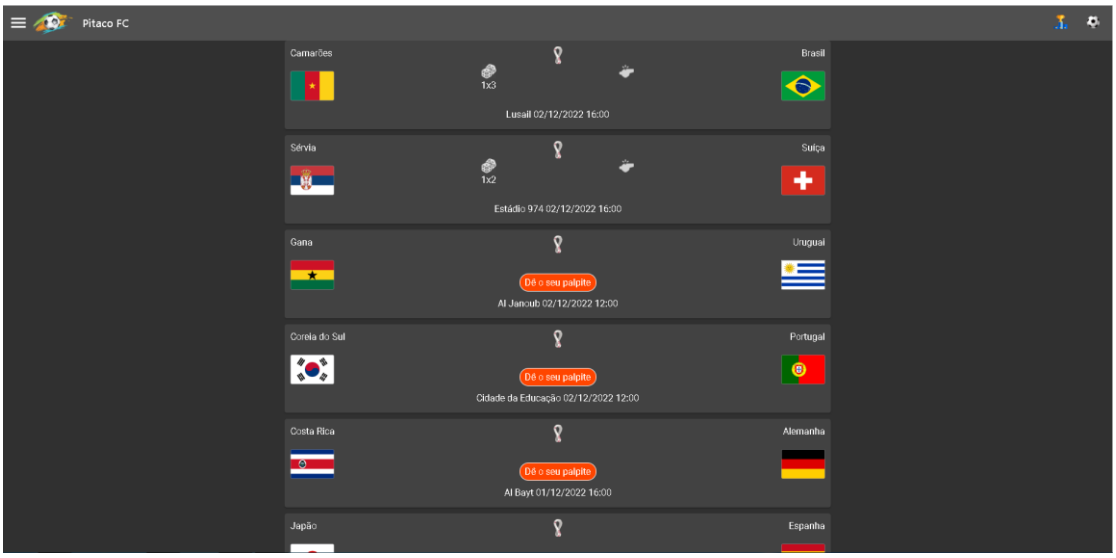

Figura 17 – Navegador Google Chrome Desktop

Figura 18 – iPhone 12 Pro (esquerda) e Sansung Galaxy S20 Ultra (direita)

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A plataforma deve permitir o acompanhamento dos resultados em tempo real, atualizando placar e tempo das partidas de forma automatizada sempre que estiver em período com partidas.
Preocupação:	

Pensando em manter o usuário o máximo de tempo envolvido utilizando a plataforma, a tela principal (Home) deve atualizar-se automaticamente a cada 30 segundos, sem a necessidade de ação manual do usuário.	
Cenário(s):	
Cenário 1	
Ambiente:	
Usuário visualizando a Home da plataforma com partidas em andamento.	
Estímulo:	
Usuário acompanhando partidas em tempo real.	
Mecanismo:	
Timer de atualização no front-end de 30 segundos	
Medida de resposta:	
Exibição de loading para fácil percepção do usuário de que a tela está em atualização.	
Considerações sobre a arquitetura:	
Riscos:	Interrupção da rede do usuário por queda.
Pontos de Sensibilidade:	Lentidão na rede do usuário.
Tradeoff:	Não há

A tela de Home exibe as partidas que acontecem no dia, com o objetivo de lembrar o usuário dos jogos ainda não palpitados, e assim que as o primeiro jogo começa, é inicializado um cron que atualiza a interface a cada 30 segundos, construindo a experiência de acompanhamento da partida em tempo real, e para evidenciar que está ocorrendo uma atualização nos cards, é exibido um loading verde logo acima do primeiro card. Essa atualização hoje acontece do front para o back, ou seja, requisições HTTP para o servidor solicitando as atualizações. A imagem 19 exibe o momento de atualização da Home.

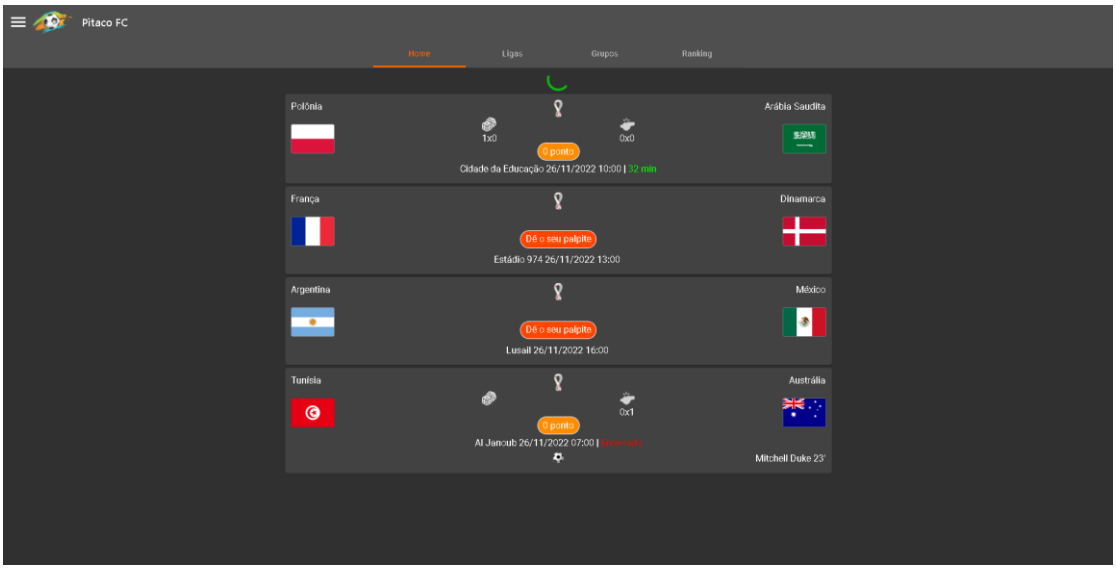

Figura 19 – Atualização em tempo real via cron de 30 segundos.

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	As consultas de ranking, mais onerosas, devem utilizar cache e serem atualizados sempre que os placares forem alterados.
Preocupação:	
Responder rapidamente ao usuário mesmo em consultas onerosas do banco de dados. Melhorar a performance de consultas de forma a impactar o menos possível as transações online do banco de dados.	
Cenário(s):	
Consulta do ranking do campeonato.	
Ambiente:	
Usuário visualizando a tela de ranking da plataforma.	
Estímulo:	
Usuário verificar a sua posição dentro do campeonato de palpites.	
Mecanismo:	
Cache de consultas onerosas, atualizadas periodicamente de acordo com o andamento das partidas.	
Medida de resposta:	
Backend devolvendo respostas imediatas sem consulta em base de dados.	

Considerações sobre a arquitetura:	
Riscos:	Estouro do limite configurado para cache.
Pontos de Sensibilidade:	Lentidão de consulta em base de dados.
Tradeoff:	Não há

Cachear as consultas mais utilizadas e onerosas no banco de dados aumenta a performance da aplicação, uma vez que a consulta é feita e armazenada em cache por 60 segundos, o que permite reduzir o tempo de resposta para o usuário da plataforma, com uma visível percepção. As figuras 20 e 21 abaixo, são respectivamente uma consulta em base de dados, onde o front-end teve uma espera de 670.05 ms, enquanto a consulta cacheada foi de apenas 93.93 ms, o que rende uma melhoria de pouco mais de 71.33% do tempo gasto.

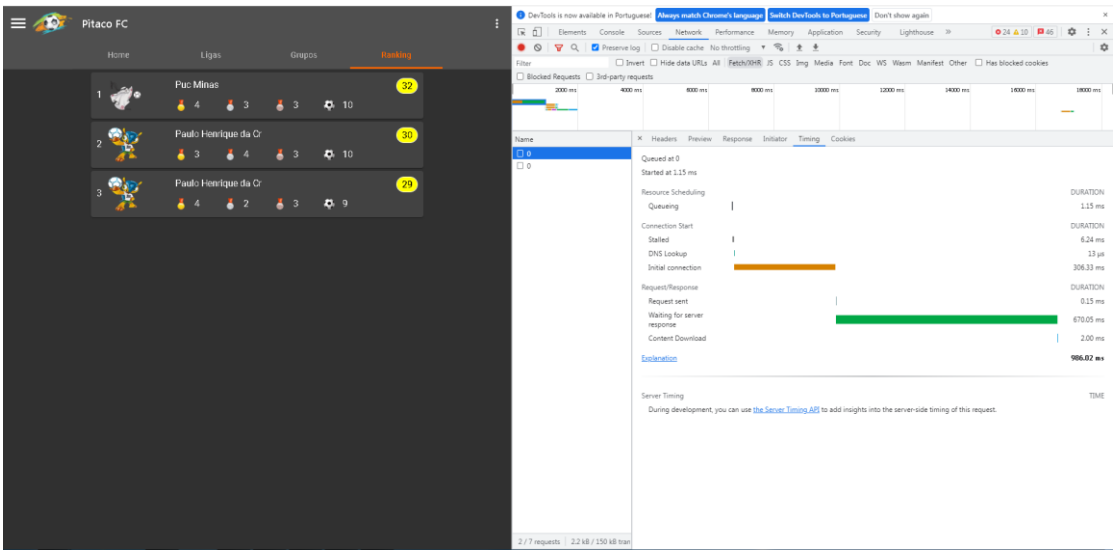

Figura 20 – Tempo de chamada com consulta em base de dados.

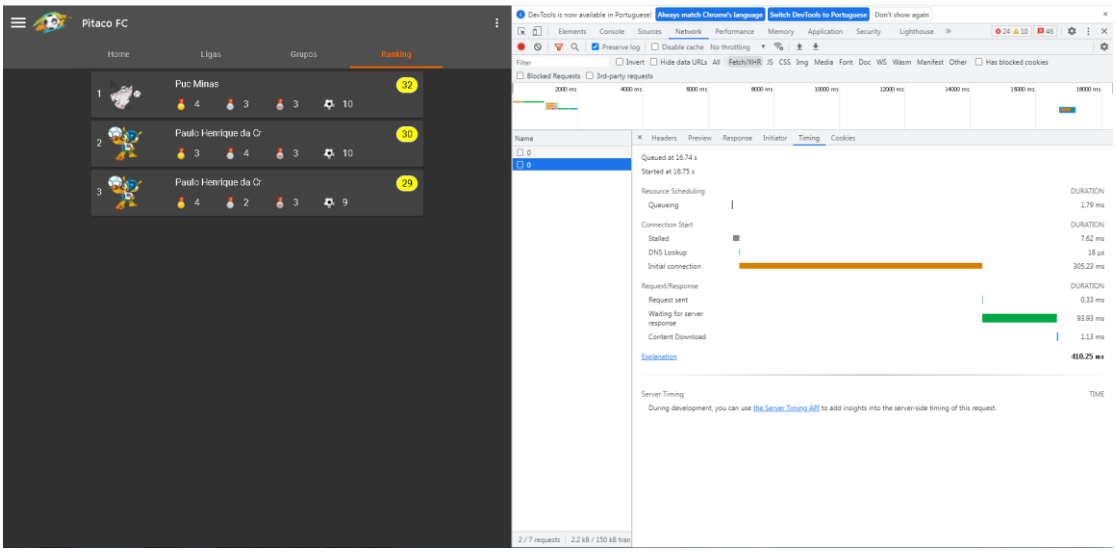

Figura 21 – Tempo de chamada com consulta em cache.

A figura abaixo mostra o monitoramento da utilização dos caches da aplicação.

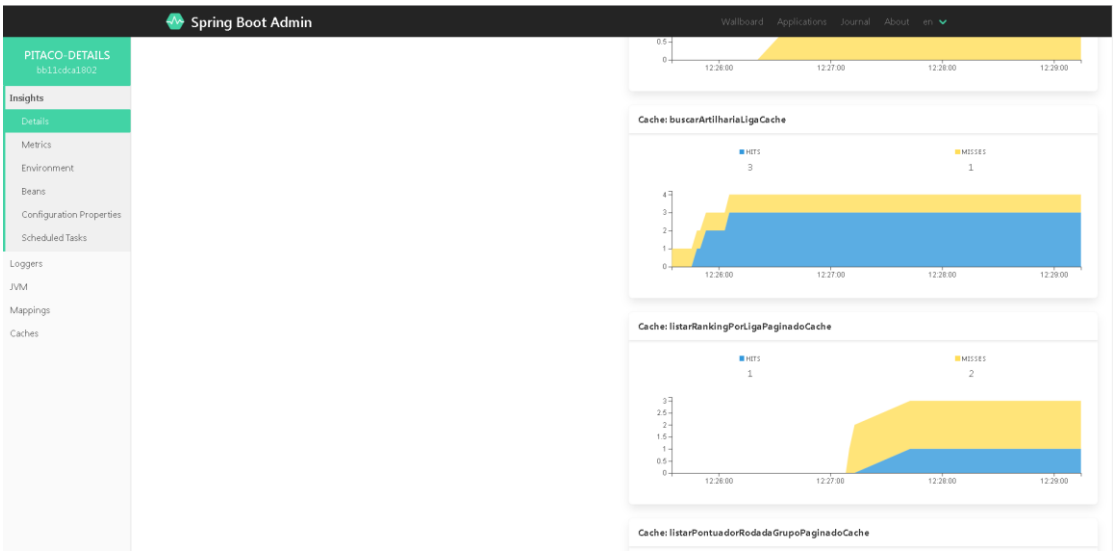

Figura 22 – Monitoramento de chamadas em com consulta de cache.

Atributo de Qualidade:	Monitoramento
Requisito de Qualidade:	A plataforma deve permitir o monitoramento dos microserviços.
Preocupação:	
Para evitar e minimizar problemas de acesso dos usuários a plataforma, é essencial que o monitoramento dos serviços em execução alerte o time de desenvolvimento de possíveis quedas e instabilidades para que possíveis atitudes sejam tomadas imediatamente.	
Cenário(s):	
Queda de um microserviço ou instabilidade que o deixe inoperante.	
Ambiente:	
Comunicação entre os microserviços ou front-end.	
Estímulo:	
O microserviço com problemas notifica o sistema de service-discovery (pitaco-service-discovery) que está inoperante, que imediatamente é refletido pelo service-admin (pitaco-admin) e o mesmo notifica/alerta o administrador do incidente.	
Mecanismo:	
Netflix eureka, Spring boot Admin	
Medida de resposta:	
Exibição de mensagem tratada para o cliente e exibição no dashboard de serviço indisponível par ao administrador.	
Considerações sobre a arquitetura:	
Riscos:	Instabilidades de rede, sobrecarga de memória, entre outros motivos que possam deixar os microserviços inoperantes.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Para atendimento a este requisito foi utilizado o Spring Boot Admin que auxilia no monitoramento completo e integrado dos microserviços, através de interfaces limpas e intuitivas, realizando o monitoramento e saúde dos serviços desde a utilização da CPU e memória, ao trafego de requisições e disponibilidade dos mesmos. Abaixo as imagens

mostram o dashboard com todos os serviços disponíveis, seguida da imagem que apresenta a indisponibilidade do serviço “pitaco-authentication”, o que gerou um alerta para o administrador.

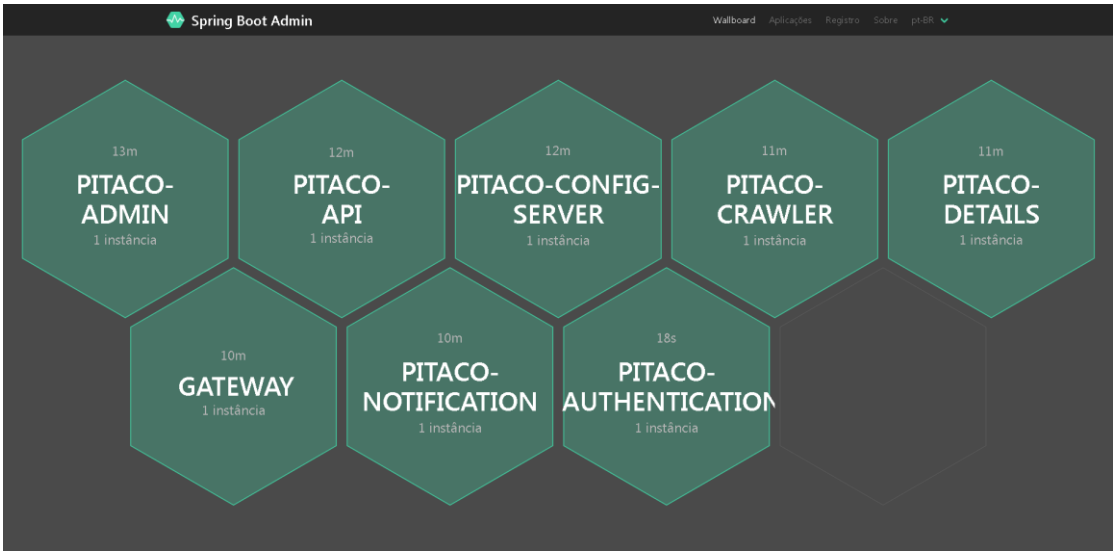

Figura 23 – Monitoramento com todos os serviços disponíveis.

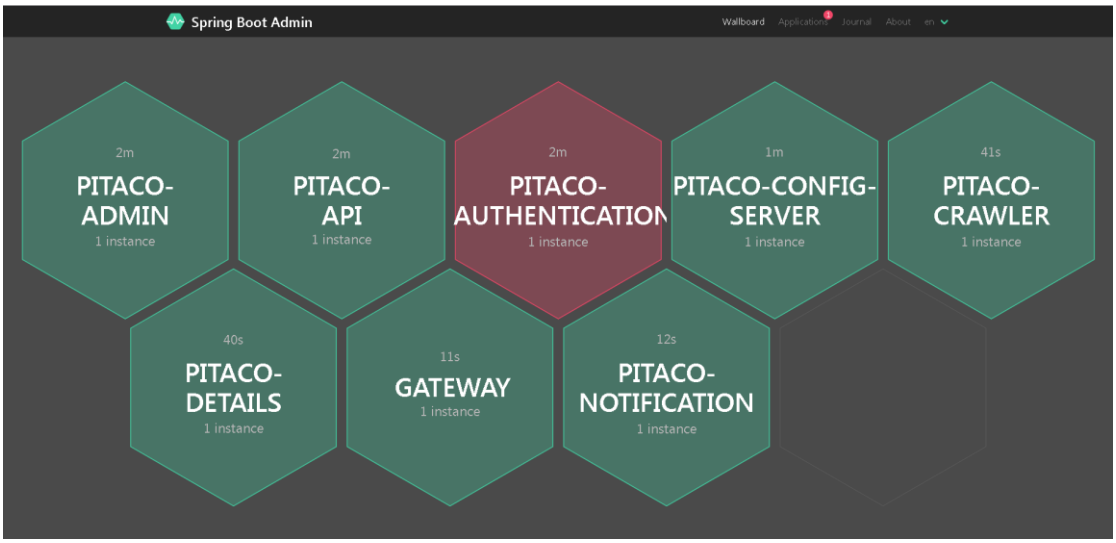

Figura 24 – Monitoramento com o serviço pitaco-authentication indisponível.

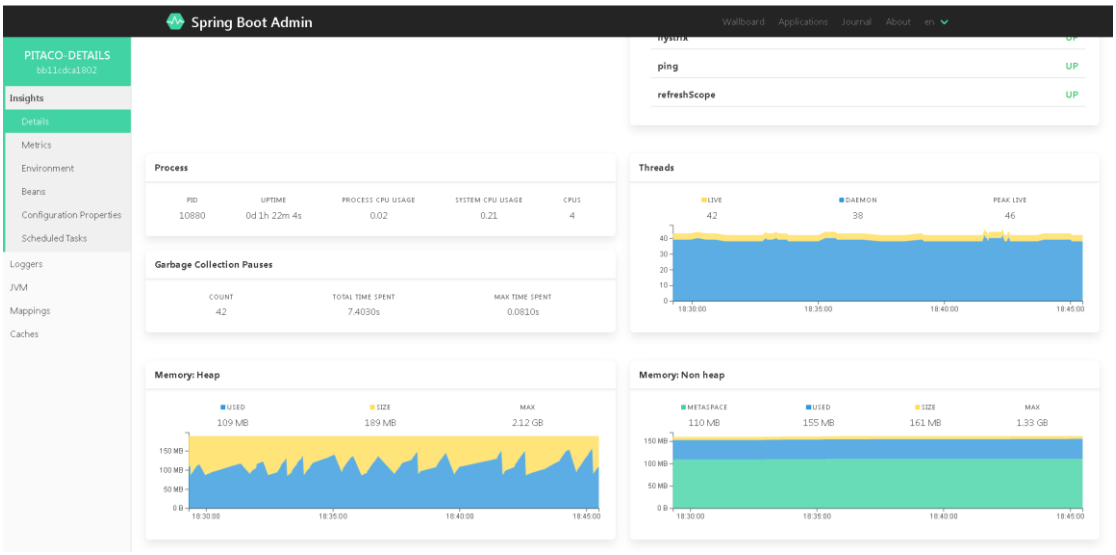

Figura 25 – Dados de utilização do serviço pitaco-details (CPU, memória, cache, threads, etc) em monitoramento.

6.4. Resultados Obtidos

Apresente os resultados da arquitetura produzida, indicando seus pontos fortes e suas limitações. A título de sugestão construa uma tabela apresentando esses resultados, como no exemplo que segue:

Requisitos Não Funcionais	Teste	Homologação
RNF01: A plataforma deve operar e estar disponível para acesso todos os dias e permitir ser acessada por navegadores web e mobile.	OK	OK
RNF03: A plataforma deve permitir o acompanhamento dos resultados em tempo real, atualizando placar e tempo das partidas de forma automatizada sempre que estiver em período com partidas.	OK	OK
RNF07: A atualização de dados do jogos deve acontecer de forma automática, e paralela para não afetar os acessos on-line.	OK	OK
RNF08: As consultas de ranking, mais onerosas, devem utilizar cache e serem atualizados sempre que os placares forem alterados.	OK	OK
RNF10: A plataforma deve permitir o monitoramento dos microserviços.	OK	OK

Obs: N.A.: não se aplica.

7. Avaliação Crítica dos Resultados

Abaixo é apresentado um resumo com o objetivo de evidenciar os pontos positivos e negativos da arquitetura proposta par ao desenvolvimento de uma plataforma escalável.

Ponto avaliado	Descrição
Arquitetura Cloud	O uso de uma arquitetura descentralizada utilizando microserviços com Spring Cloud, traz ganhos muito evidentes a plataforma, uma vez que para realização de uma manutenção é possível deixar indisponível apenas o microserviço responsável pela funcionalidade, mantendo a plataforma operacional em todas as outras funcionalidades. Um ponto de atenção é a evolução da arquitetura para um padrão multi cloud e multi region, que em casos de desastres e interoperabilidade de um provedor, outro assume e mantem a aplicação online e disponível para os usuários. Um ponto muito positivo da utilização do conjunto framework spring cloud, é realizar o refresh de dos properties da aplicação sem a necessidade de um novo deploy, sendo necessário apenas uma chamada http e toda a atualização acontece.
Plataforma responsiva	O uso de responsividade na plataforma, permite o ganho de usuários de dispositivos móveis de todos os tipos e todos os sistemas operacionais. A responsividade nesse quesito traz a mesma experiência para o usuário de um desktop e para o os usuários mobile, e a unificação dessa experiência independente do dispositivo agrega maior sensação de conhecimento por parte do usuário.

Atualizações em tempo real	Mostras em tempo real as partidas, de forma automatizada, traz a real sensação dos usuários de acompanhar os jogos em tempo real, pois é exibido o tempo da partida, os nomes dos jogadores que marcaram junto com o minuto marcado, e também quando existem, as penalidades. Essa atualização ocorre atualmente via HTTP, o que pode onerar o servidor com requisições, uma vez que todos os usuários têm sua tela atualizada a cada 30 segundos quando tem partidas em andamento. Um ponto mapeado de melhoria e evolução da plataforma é a utilização de Websocket para essa atualização, mantendo uma comunicação aberta entre o servidor e os clientes, e o servidor enviar a atualização para os clientes realizarem o processamento da atualização das informações.
Gamificação	Estimula a competição entre os usuários, fornecendo um feedback em tempo real dos rankings, na mesma proporção que das partidas em tempo real, pois sempre que um gol é feito o ranking é atualizado, sendo que no começo da partida, o mesmo usuário pode se ver em diferentes posições de ranking com o andamento das partidas.

Monitoramento	O uso de uma arquitetura cloud juntamente com ferramentas como Netflix Eureka e Spring Boot Admin, fornecem controles facilitados de monitoramento dos serviços de forma isolada e individual, permitindo criação de monitores e alertas customizados para cada aplicação, como por exemplo a aplicação de login sendo mais crítica enviar SMS, enquanto um serviço de menos criticidade como o de notificações pode gerar um alerta apenas no dashboard. Um ponto de atenção é que a depender da quantidade de serviços monitorados, em algum momento pode ser necessário escalar o número de maquinas de forma horizontal para monitoramentos.

8. Conclusão

A Copa do mundo é sem dúvida um dos eventos mais aguardados pelos fãs de futebol no mundo inteiro, e a proposta do projeto em ser uma plataforma acessível em múltiplos dispositivos com diferentes sistemas operacionais e tamanho de tela, auxilia no ganho de usuários, utilizar de gamificação para estimular a competição entre os usuários torna, assim como fornecer prêmios são grandes atrativos. A mesma ideia de utilizar amantes de futebol, e utilizar o engajamento de seus usuários pode ser vista em utras plataformas de renome como Nubank com o Nubolão (https://blog.nubank.com.br/nubolao-app-nubank/), Cartola Express (https://jogue.cartolaexpress.globo.com/), entre outros.

A definição de uma arquitetura limpa, escalável e reutilizável, vai muito além das tecnologias aplicadas, ela precisa estar extremamente alinhada com o negócio e suas regras, com o objetivo de torná-la adaptável e evolutiva. A estrutura de microserviços permite uma fácil evolução e crescimento da plataforma, uma vez que é possível realizar manutenção e evolução em apenas um único microserviço, além de permitir maior controle de custos com escalabilidade, pois é possível aumentar a quantidade de pequenos contêineres com maior volume de utilização, inclusive configurar essa escalabilidade em momentos de pico de uso da plataforma. O uso de ferramentas Cloud como a framework Spring, traz ganhos evidentes durante o desenvolvimento, manutenibilidade, monitoramento entre outros.

A construção da plataforma teve um papel muito importante para aprofundamento no conhecimento sobre arquitetura, evolução sobre integrações, sobretudo sobre o Spring Framework, que foi um grande aliado no desenvolvimento de todo o backend, além do Angular utilizado no frontend. O entendimento de que a arquitetura distribuída possui muitos prós, porém a mesma não deve ser vista como bala de prata, pois também possui seus contras, assim como também acontece na arquitetura monolítica. Importante ressaltar que o mercado atualmente vive uma forte demanda de migração para Cloud, juntamente com a estrutura de microserviços, porém toda evolução e migração deve ser muito bem planejada e estruturada para de fato trazer os ganhos para todo o negócio.

Uma evolução já mapeada para a plataforma é a utilização de websocket na tela Home, que utiliza hoje chamadas HTTP para obter dados em tempo real das partidas, essa melhoria permitirá que o servidor envie atualizações para os clientes sempre que houver alteração dos placares e tempo, permitindo que a comunicação seja feita a cada 1 minuto, se tratando de alteração do tempo do jogo, ou a cada alteração de placar, uma vez que hoje o cliente realiza chamadas REST a cada 30 segundos.

Lições aprendidas:

1- Definir a regra de negócio e construir microserviços de responsabilidade única é extremamente difícil, mas se bem estruturado torna muito simples a evolução das funcionalidades, de forma independente do sistema como um todo, e permite manutenções programadas com a plataforma sendo utilizada, parando apenas uma funcionalidade.

2- É essencial ter um sistema integrado de monitoramento que auxilie a controlar de forma individual a saúde/disponibilidade de cada microserviço, trazendo ganhos e controle sobre a utilização de cada funcionalidade, controle de memória, CPU, etc, dessa forma é possível aumentar de forma isolada a quantidade de recursos disponíveis para cada microserviço.

3- Utilizar SPA em um frontend traz maior leveza na navegabilidade, uma vez que as informações são obtidas de acordo com a navegabilidade do usuário, apoiado na responsividade, possibilita uma gama maior de usuários, pois a mesma experiência pode ser encontrada em diferentes navegadores e dispositivos com diferentes tamanhos de tela.

4- A utilização de crawler para obtenção dos dados e atualização em tempo real, permite maior confiabilidade na atualização das informações, uma vez que não é necessária a execução manual de uma pessoa, pois o processo ocorre de forma automatizada de tempos em tempos, em um microserviço apartado para não impactar os acessos online.

5- A utilização de cache maximiza a performance da aplicação com relação ao tempo de resposta das requisições, mas requer muita atenção na utilização para não devolver informações desatualizadas para os usuários.

Link do vídeo de apresentação: https://www.youtube.com/watch?v=teH5y3-bFkQ

Referências

ALMEIDA, Raphael Bastos de; ALMEIDA, Vitor Manoel Cunha de; LIMA, Diego de Favari Pereira. **Comunidades de marca de fantasy sports games: identificação, engajamento, intenção de continuidade e valor da marca do patrocinador.** In: Revista Brasileira de Marketing. São Paulo. Vol. 14, n. 1, pp 33-48 (mar/2015).

BATISTA, Aron Rodrigo de Carvalho. **A Gamificação como Recurso Estratégico de Marketing de Conteúdo: Estudo de caso do fantasy game Cartola FC**. 2018. 128 f. Trabalho de Conclusão de Curso/Mestrado – Universidade Federal do Tocantins. Palmas/Tocantins 2018.

BROW, Simon. **O Modelo C4 de documentação para Arquitetura de Software**. Infoq.com.br. Disponível em: < https://www.infoq.com/br/articles/C4-architecture-model/ >. Acesso em: 28 de Maio de 2022.

CARTOLA FC. Disponível em: < https://ge.globo.com/cartola/ >. Acesso em: 25 de Abril de 2022.

JUNIOR, Gilson Cruz. **Burlando o círculo mágico: O esporte no bojo da Gamificação.** Movimento, vol. 20, núm. 3, 2014. Pp 941-963. Escola de Educação Física. Rio Grande do Sul.

NEVES, Renan Silva. **Cartola FC Bate recordes em 2019**. Medium.com. Disponível em: < https://medium.com/betaredacao/cartola-fc-bate-recordes-em-2019-c75ceb503fb5 >. Acesso em: 25 de Abril de 2022.