**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

Abril, 2022.

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](#_Toc111895840)

[1. Introdução 3](#_Toc111895841)

[2. Cronograma do Trabalho 5](#_Toc111895842)

[3. Especificação Arquitetural da solução 6](#_Toc111895843)

[3.1 Restrições Arquiteturais 6](#_Toc111895844)

[3.2 Requisitos Funcionais 7](#_Toc111895845)

[3.3 Requisitos Não-funcionais 9](#_Toc111895846)

[3.4 Mecanismos Arquiteturais 10](#_Toc111895847)

[4. Modelagem Arquitetural 11](#_Toc111895848)

[4.1 Diagrama de Contexto 12](#_Toc111895849)

[4.2 Diagrama de Container 12](#_Toc111895850)

[4.3 Diagrama de Container 12](#_Toc111895851)

[Etapa 2 - Pendente 13](#_Toc111895852)

[Etapa 3 - Pendente 14](#_Toc111895853)

[Referências 14](#_Toc111895854)

## 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.

## 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 | 1. Contextualização do trabalho | Construção da contextualização deste projeto |
| 25/04/2022 | 26/04/2022 | 1. Definição dos requisitos Arquiteturais | Lista dos requisitos Arquiteturais identificados |
| 26/04/2022 | 27/04/2022 | 1. Definição dos requisitos Funcionais | Lista dos requisitos funcionais identificados |
| 27/04/2022 | 28/04/2022 | 1. Definição dos requisitos Não-funcionais | Lista dos requisitos Não-funcionais identificados |
| 28/04/2022 | 29/04/2022 | 1. Definição dos Mecanismos Arquiteturais | Lista dos Mecanismos Arquiteturais identificados |
| 02/05/2022 | 03/05/ 2022 | 1. Construção dos Diagramas de Contextos | Diagrama de contexto criado no Draw.io e documentado |
| 05/05/2022 | 15/05/2022 | 1. Revisão da Etapa 1 | Documento Etapa 1 revisado |
| 16/05/2022 | 18/05/2022 | 1. Apresentação em PPT da Etapa 1 | PPT |
| 30/07/2022 | 30/07/2022 | 1. Construção do vídeo de apresentação da Etapa 1 | Vídeo criado da Etapa 1 |
| 30/07/2022 | 30/07/2022 | 1. Submissão do vídeo da etapa 1 no Youtube | Vídeo disponibilizado publicamente no Youtube |
| 14/04/2022 | 14/08/2022 | 1. Publicação no repositório Github Etapa 1 | Arquivos produzidos no Github disponíveis abertamente |
| 16/08/2022 | 18/08/2022 | 1. Construção dos Diagramas de Contêineres | Diagramas de contêineres |
| 18/08/2022 | 20/08/2022 | 1. Construção dos Diagramas de Componentes | Diagramas de componentes |
| 20/08/2022 | 01/10/2022 | 1. 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 | 1. Publicação da plataforma na internet | Disponibilização da plataforma aberta na internet |

## 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.

## 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. |

## 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.

## 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 |

\*B=Baixa, M=Média, A=Alta.

## 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.

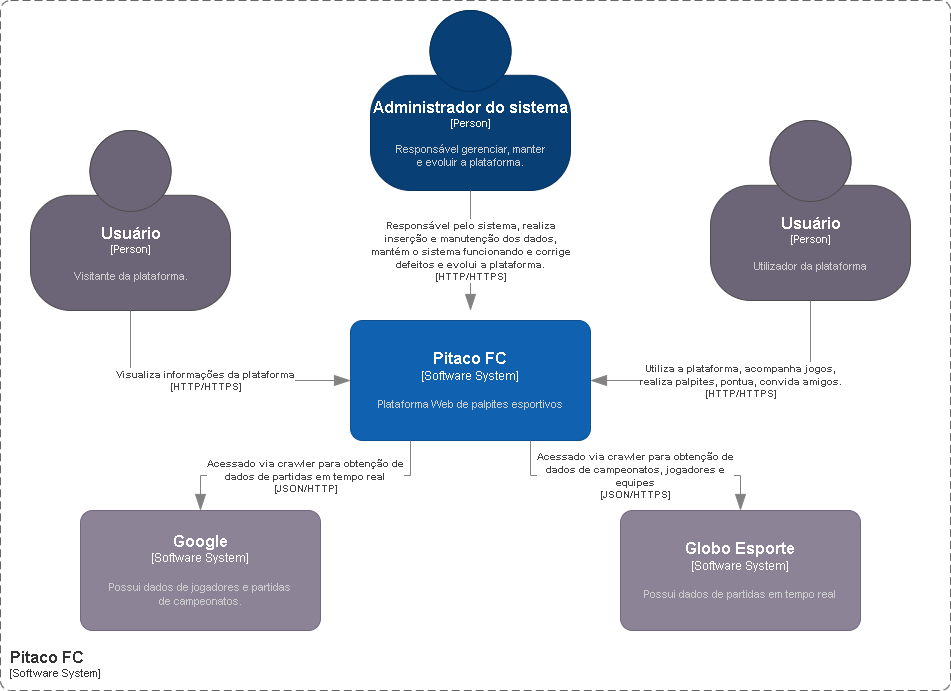
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | Hibernate com utilização do Spring Data JPA |
| Persistência | SGBD | MySQL |
| Front end | SPA – Single Page Application | Angular/TypeScript |
| Back end | Microserviço | Java 11 |
| Back end | 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 |

## 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.

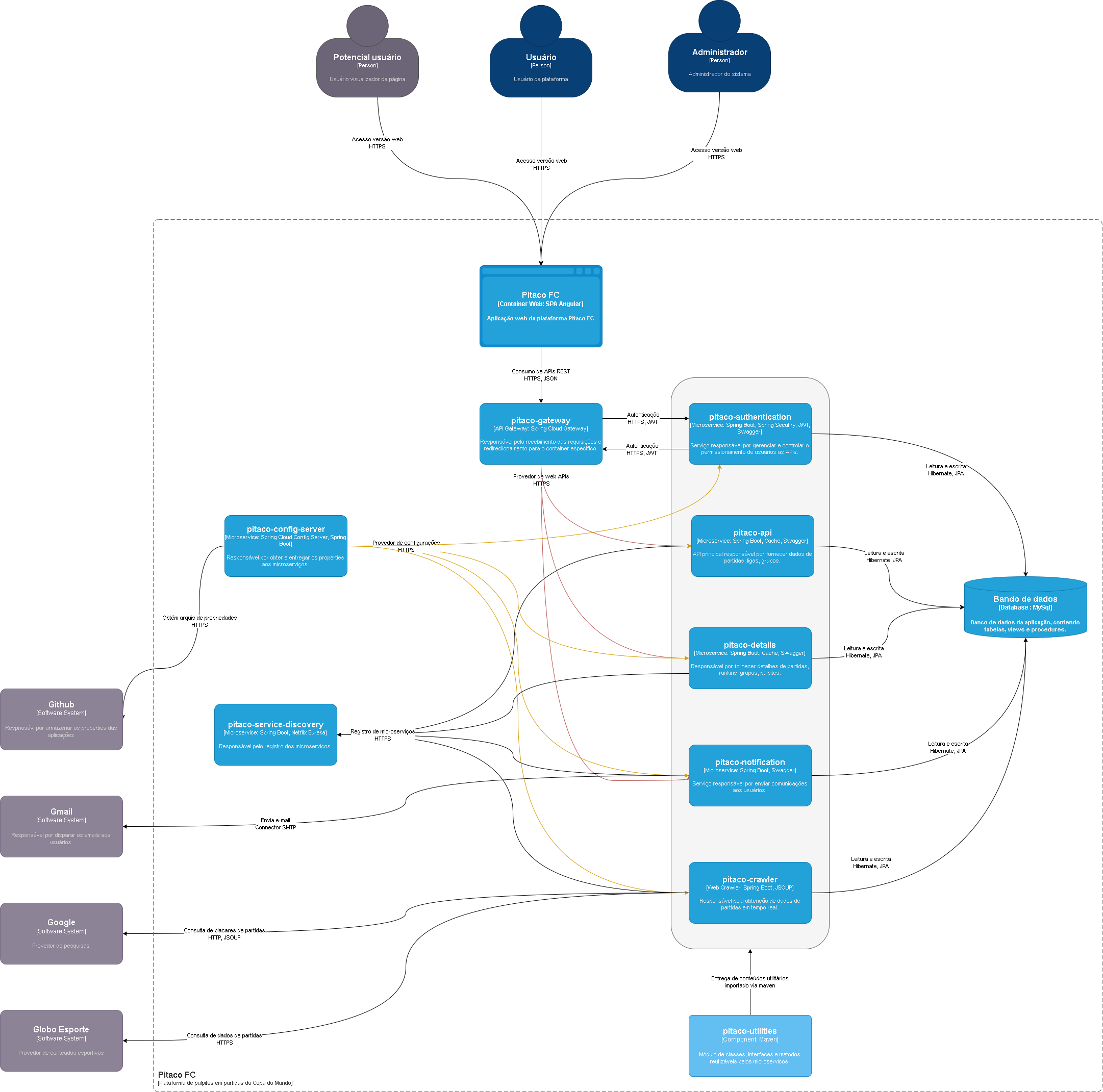
## 4.1 Diagrama de Contexto



**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.

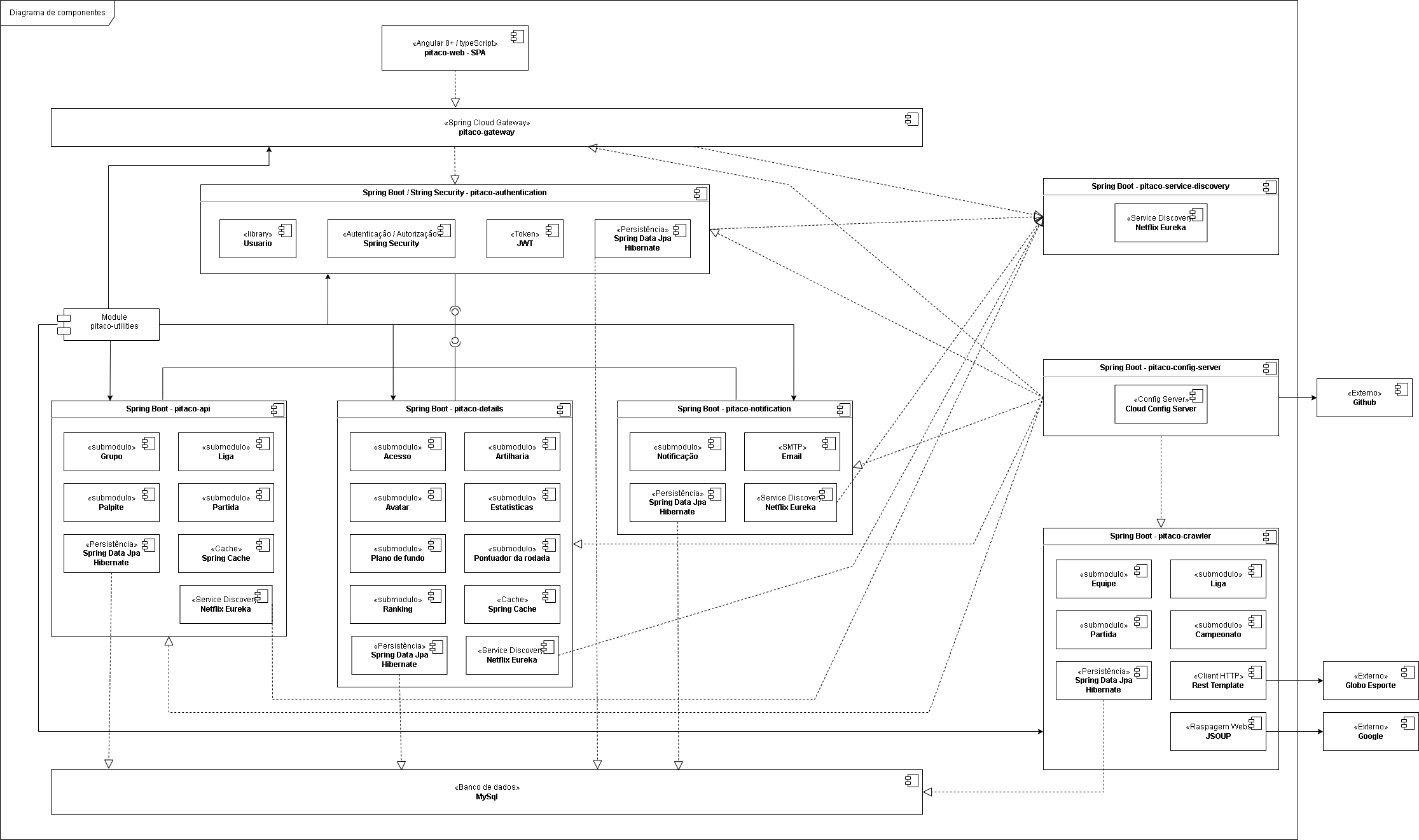
O pitaco-service-config é responsável por obter os arquivos de configuração (.properties) de um repositório no github e entrega-los a cada um dos microserviços da arquitetura, o que permite que cada microserviço tenha seu arquivo especifico e também um arquivo de configurações compartilhadas. A utilização 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 rankins, 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 literalemten 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.

## Prova de Conceito (PoC)

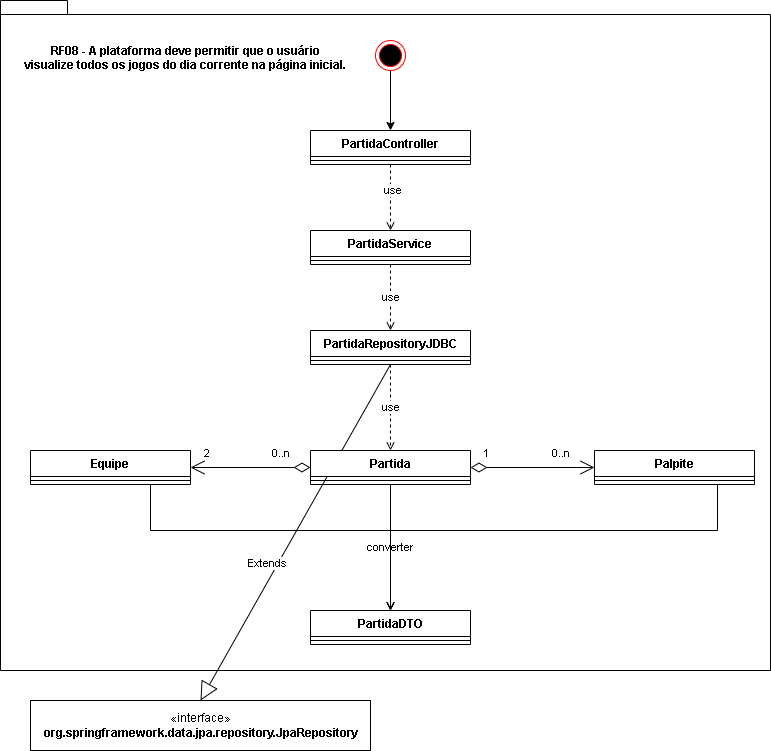
Nessa sessão, será aprofundado o detalhamento do projeto desenvolvido a prova conceitual, validando a arquitetura proposta e funcionalidades levantadas nos requisitos funcionais e não funcionais com o objetivo de disponibilizar o sistema desenvolvido na web.

## Integrações entre Componentes

*<Conteúdo a ser produzido – Data final 15 de dezembro>*

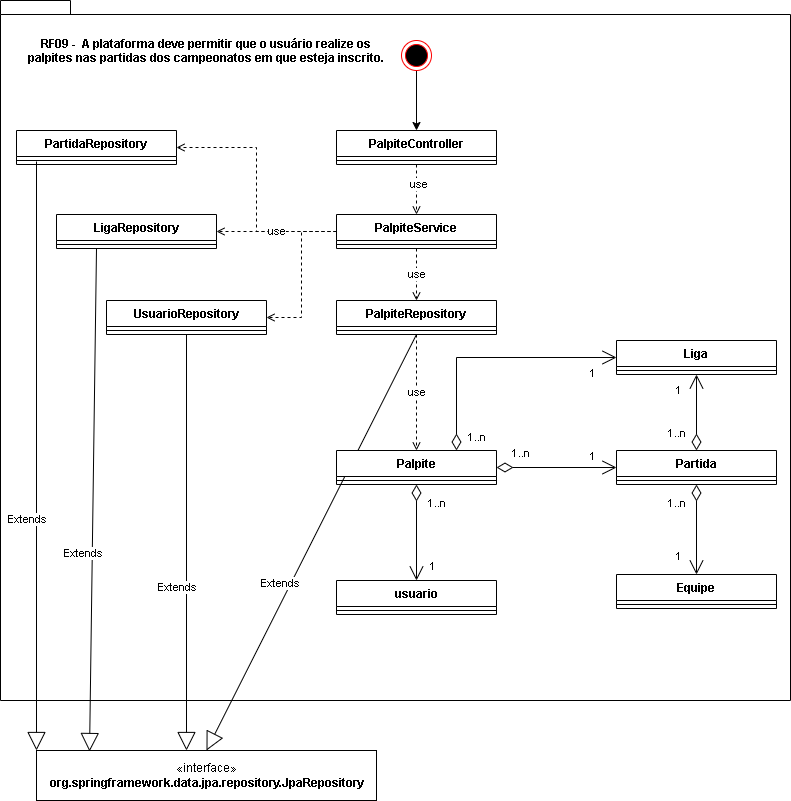
## Código da Aplicação

*<Conteúdo a ser produzido – Data final 15 de dezembro>*



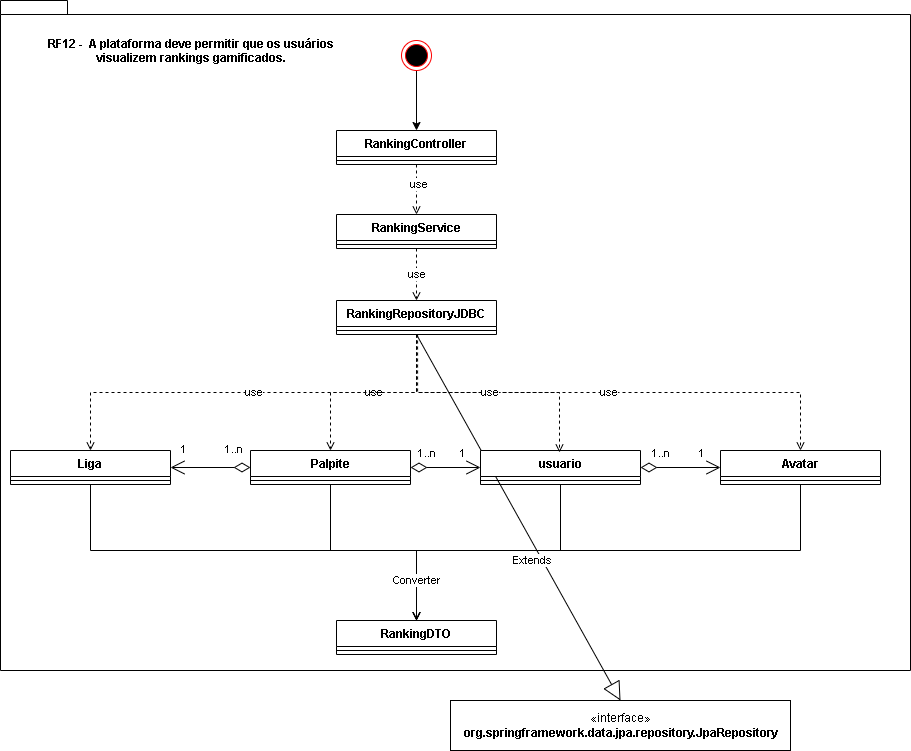
**Figura 4 – RFO8 – Permitir que o usuário visualize os jogos do dia corrente na página inicial.**

A figura 4, 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 5 – RF09 – Permitir que o usuário palpite nas partidas.**

O RF09, exibido na figura 5, 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 o modelo 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 6 – RF12 – Visualizar rankings gamificados.**

A figura 6 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.

## 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

Apresente aqui um breve resumo das principais características da proposta arquitetural. Para isto, utilize o método Architecture Tradeoff Analysis Method (ATAM), no qual são utilizados cenários para fazer essa análise.

Exemplo:

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de Quailidade** | **Cenários** | **Importância** | **Complexidade** |
| Interoperabilidade | Cenário 1: O sistema deve se comunicar com sistemas de outras tecnologias. | A | M |
| Usabilidade | Cenário 2: O sistema deve prover boa usabilidade. | M | B |
| Manutenibilidade | Cenário 3: O sistema deve ter a manutenção facilitada. | M | M |

## 6.2. Cenários

Mostre os cenários utilizados na realização dos testes da sua aplicação. Escolha cenários de testes que demonstrem os requisitos não funcionais (atributos de qualidade) sendo satisfeitos. Priorize os cenários para a avaliação segundo critérios quantitativos ou qualitativos.

*Exemplos de cenários:*

*Cenário 1 - Interoperabilidade: Ao acessar a URL do serviço de informações gerenciais via HTTP GET, o mesmo deve retornar as informações no formato JSON.*

*Cenário 2 - Usabilidade: Ao navegar na tela, o sistema deve apresentar boa usabilidade. A navegação deve apresentar facilidade e o acesso as funcionalidades deve ser bem objetivo para a função que precisar ser realizada, o usuário deve ser capaz de efetuar uma compra em no máximo 5 minutos, assim garantindo a agilidade e a usabilidade para ficar de acordo com um dos requisitos não funcionais.*

*Cenário 3 - Manutenibilidade: Havendo a necessidade de alterar o gateway de pagamento somente será necessário fazer alteração no broker da funiconalidade de pagamento, facilitando a manutenção e os testes.*

## 6.3. Evidências da Avaliação

Apresente as medidas registradas na coleta de dados. Para o que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento ao requisito não-funcional.

|  |  |
| --- | --- |
| Atributo de Qualidade: | Interoperabilidade |
| Requisito de Qualidade: | O sistema deve se comunicar com outras tecnologias. |
| Preocupação: | |
| O sistema deve ter como resposta a uma requisição uma saída de fácil leitura por outro componente. | |
| Cenário(s): | |
| Cenário 1 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| O sistema de monitoramento envia uma requisição para o serviço REST do módulo de informações gerenciais. | |
| Mecanismo: | |
| Criar um serviço REST para atender às requisições do sistema de monitoramento | |
| Medida de resposta: | |
| Retornar os dados requisitados no formato JSON | |
| Considerações sobre a arquitetura: | |
| Riscos: | Alguma instabilidade na rede pode deixar a conexão lenta ou mesmo a perda de pacotes. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

Acrescente imagens e descreva os testes realizados, de tal forma que se comprove a realização da avaliação.

Faça isto para todos os cenários apresentados no tópico 6.1.

## 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: O sistema deve ... | OK | OK |
| RNF02: O sistema deve ... | OK | N.A. |
| RNF03: ... | OK | N.A. |

Obs: N.A.: não se aplica.

## Avaliação Crítica dos Resultados

Apresente aqui, de forma resumida, os principais pontos positivos e negativos da arquitetura proposta. Adote uma postura crítica que permita entender as limitações arquiturais, incluindo os prós e contras das tecnologias. Você pode utilizar o formato textual ou produzir um quadro resumo.

Ex. de quadro resumo:

|  |  |
| --- | --- |
| **Ponto avaliado** | **Descrição** |
| xxxxxxxxxxxxxxx | xxxxxxxxxxxxxxxxxxxx |
|  |  |
|  |  |

## Conclusão

Descreva, de forma sucinta, quais foram as lições aprendidas na execução do seu projeto arquitetural. Procure apresentá-las de tal forma que fiquem configurados os *trade-offs* da arquitetura produzida, como por exemplo, Segurança X Desempenho, Granularidade X Manutenibilidade, etc.

Aqui deve ser apresentado também tudo que se aprendeu com esse projeto, de modo a servir como ajuda para outros profissionais.

Também se faz necessário evidenciar as possibilidades de melhoria do projeto, caso se deseje dar continuidade a ele. Nesse sentido, indique possíveis ajustes ou melhorias arquiteturais, que possam vir a ser realizados futuramente.

Lições aprendidas (ex.):

1. xxxxxxxxxxxxxxxxx
2. xxxxxxxxxxxxxxxxx
3. xxxxxxxxxxxxxxxxx

## 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.