Universidade de Brasília - UnB

Faculdade do Gama - FGA

Centauri - Onde é o Jogo?

DOCUMENTO DE ARQUITETURA

Arquitetura Versão 1.0

Gama - DF

2024

Histórico de Revisão

Data	Versão	Descrição	Autor(es)
20/05/2024	1.0	Versão inicial da arquitetura	Equipe Centauri
01/07/2024	2.0	Versão final da arquitetura	Equipe Centauri

Autores:

Matrícula	Nome	Descrição do papel assumido na equipe	% de contribuição ao trabalho (*)[1]
211031566	Ana Joyce Guedes Amorim da Silva	Product Owner/Desenvolvedora	13.5%
221031111	Anne de Capdeville	Analista de Qualidade/Desenvolvedora	12.3%
222006712	Fábio Gabriel da Silva Barbosa	Desenvolvedor	12.5%
202016248	Gabriel Ferreira de Freitas Dutra	Product Owner	10.5%
211030792	Hauedy Wegener Soares	Product Owner/Desenvolvedor	12.3%
221008211	José Felipe Oliveira	Desenvolvedor	12.3%
211062277	Matheus Duarte da Silva	Desenvolvedor	12.3%
222007012	Matheus Rodrigues da Silva	Analista de Qualidade/Desenvolvedor	12.3%

SUMÁRIO

1. Introdução

1.1 Propósito

Este documento descreve a arquitetura do sistema sendo desenvolvido pelo grupo "Centauri", na disciplina de MDS – Métodos de Desenvolvimento de Software – edição do primeiro semestre de 2024. Do sistema "Onde é o Jogo", a fim de fornecer uma visão abrangente para desenvolvedores, testadores e demais interessados.

1.2 Escopo

Dada a dificuldade de encontrar informações sobre onde as partidas de futebol serão transmitidas, considerando a variedade de plataformas de streaming e canais de TV competindo pelos direitos de transmissão, o grupo Centauri identificou a necessidade de um software acessível e intuitivo. Este software visa fornecer informações cruciais sobre as partidas, como o canal/plataforma de transmissão, horário e placar, caso a partida esteja em andamento.

O objetivo é fornecer um serviço eficiente e acessível que simplifique o acesso às informações de transmissão de partidas de futebol, atendendo especialmente às necessidades de usuários com menos familiaridade tecnológica e oferecendo uma alternativa prática às pesquisas tradicionais na internet.

2. Representação Arquitetural

2.1 Definições

O sistema seguirá uma arquitetura, estilo arquitetural MVC (Model View Controller).

2.2 Justifique sua Escolha.

A escolha da arquitetura MVC para o desenvolvimento do sistema "Onde é o Jogo", foi baseada na capacidade de suportar um sistema modular, manutenível e escalável, que

atende diretamente às necessidades de nosso público-alvo e aos objetivos do projeto.

- **Separação de Responsabilidades:**
    - Model (Modelo): responsável pela lógica de negócios e gerenciamento de dados. No nosso sistema, o modelo gerencia informações sobre times, jogos e transmissões. Esta separação facilita a manutenção e atualização dos dados de transmissão sem afetar a interface ou a lógica de controle.
    - View (Visão): Encapsula toda a lógica de apresentação. A View do nosso sistema será responsável por renderizar páginas personalizadas para os usuários, mostrando informações como horários, canais e placares das partidas. Isso garante que as interfaces sejam atualizadas rapidamente e de forma independente dos dados e da lógica de controle.
    - Controller (Controlador): Intermediário que processa as solicitações dos usuários, interage com o Model e seleciona a View apropriada. No nosso caso, os controladores irão gerenciar as requisições para buscar dados de partidas e exibir informações personalizadas, facilitando a implementação de funcionalidades de busca e filtragem.

A arquitetura MVC permite que cada componente do sistema seja desenvolvido, testado e mantido de forma independente. Isso é crucial para um sistema como o Transmissão de Jogos, que pode necessitar de frequentes atualizações de dados de transmissão e melhorias na interface do usuário para manter-se competitivo e útil. A separação clara entre Model, View e Controller facilita a adição de novas funcionalidades, como suporte a novos campeonatos ou integração com diferentes APIs de transmissão, sem grandes reestruturações no código existente.

- Reuso de Componentes:
    - Os componentes da View podem ser reutilizados para diferentes partes do sistema, como a home page personalizada e as páginas de detalhes de jogos. Isso reduz o esforço de desenvolvimento e garante uma experiência de usuário consistente.
    - Os modelos e controladores podem ser reutilizados para diferentes

funcionalidades, como a busca de jogos e a atualização de informações em tempo real.

- Facilidade de Testes:
    - A arquitetura MVC facilita a realização de testes unitários e de integração. Cada componente pode ser testado isoladamente, garantindo que o sistema funcione corretamente. Isso é particularmente importante para o nosso público-alvo, que inclui usuários com pouca proficiência tecnológica, exigindo um sistema robusto e confiável.

Conforme detalhado na Declaração de Escopo do Produto, nosso público-alvo inclui indivíduos com dificuldades em utilizar ferramentas de pesquisa complexas. A arquitetura MVC nos permite criar uma interface de usuário simples e intuitiva, com respostas rápidas às interações do usuário, melhorando a usabilidade e acessibilidade do sistema.

2.3 Detalhamento

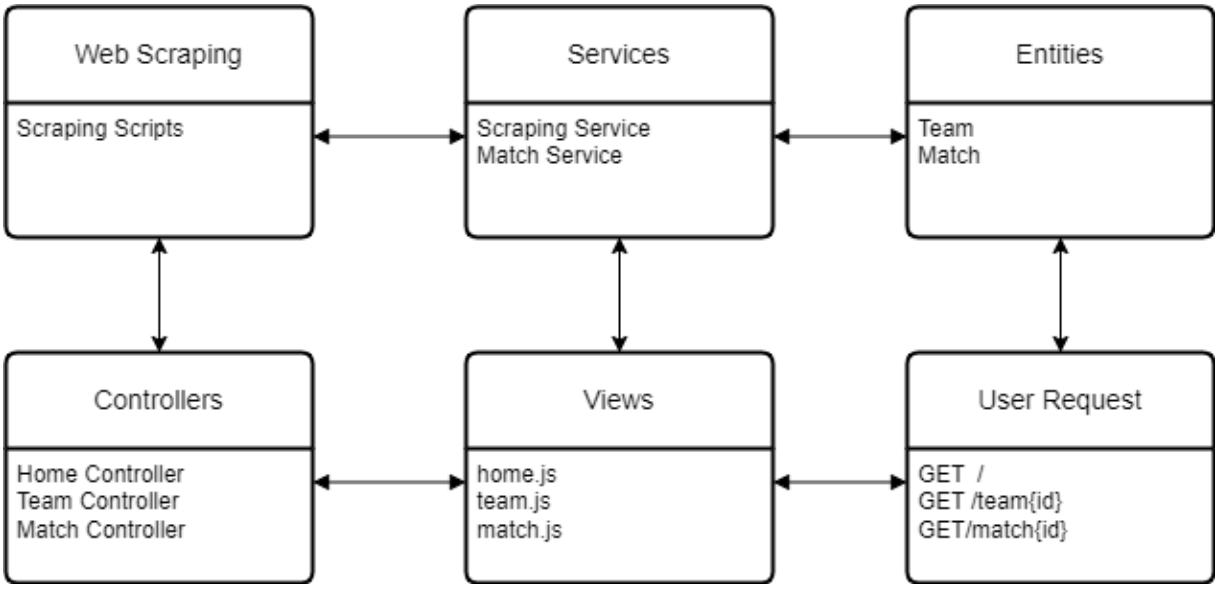

Figura 01: Detalhes de Interação do Sistema

- Web Scraping: Scripts que coletam dados de diferentes fontes (sites de transmissão, canais de TV, plataformas de streaming) sobre as partidas, como horários, canais e placares.
- Entities: Representam os dados do sistema (times e jogos).
- Services:

  Scraping Service: Responsável por executar os scripts de scraping e processar os dados coletados.

  Match Service: Contém a lógica de negócios para obter jogos por time, campeonato e canal de transmissão.

- Controllers:

  Home Controller: Recebe requisições para a home page, obtendo os jogos do time preferido do usuário e passando-os para a view home.html.

Team Controller: Processa requisições para a página de um time específico, obtendo informações detalhadas sobre o time e seus jogos, e passando-os para a view team.js.

Match Controller: Processa requisições para a página de um jogo específico, obtendo informações detalhadas sobre a partida e passando-as para a view match.js.

- Views:

  Templates Javascript: Páginas renderizadas que exibem os dados ao usuário, como home.js, team.js e match.js.

2.4 Metas e restrições arquiteturais

- Resposta rápida: O sistema deve responder a 95% das consultas de usuários em até 2 segundos. O público-alvo inclui indivíduos com pouca paciência para ferramentas de pesquisa complexas. Respostas rápidas garantem uma experiência de usuário satisfatória e aumentam a retenção.
- Suporte a Crescimentos: A arquitetura deve ser escalável para suportar um aumento no número de usuários e na quantidade de dados. O sistema deve ser capaz de lidar com o aumento no tráfego e na quantidade de informações sem degradação do desempenho, especialmente durante eventos de grande interesse (como finais de campeonato).
- Código Modular: A arquitetura deve permitir fácil manutenção e atualização do código, com componentes bem definidos e desacoplados. Facilitar a correção de bugs, a implementação de novas funcionalidades e a atualização dos dados de transmissão sem grandes reestruturações.
- Interface Intuitiva: O sistema deve ter uma interface de usuário intuitiva e fácil de usar, especialmente para indivíduos com pouca familiaridade tecnológica. O público-alvo inclui usuários de idade avançada e indivíduos sem proficiência no uso

de tecnologia. Uma interface intuitiva melhora a acessibilidade e a satisfação do usuário.

- Proteção de Dados: O sistema deve seguir as melhores práticas de segurança para proteger os dados dos usuários e evitar ataques cibernéticos. Justificativa: Garantir a confidencialidade e a integridade dos dados dos usuários é fundamental para a confiança e a conformidade com os regulamentos de proteção de dados.
- Seguir o padrão REST: As APIs desenvolvidas devem seguir o padrão REST (Representational State Transfer). O padrão REST é amplamente aceito e facilita a integração com outros sistemas e serviços, além de simplificar a comunicação cliente-servidor.
- Frontend com Next.js: O frontend deve ser desenvolvido utilizando Next.js. Permite a criação de interfaces dinâmicas, rápidas e responsivas, alinhadas com as metas de desempenho e usabilidade.
- Multiplataforma: O sistema deve ser compatível com os principais navegadores e dispositivos móveis. Garantir que o sistema seja acessível a partir de diversos dispositivos e navegadores aumenta o alcance e a conveniência para os usuários.
- Uso de Web Scraping: Os scripts de web scraping devem ser executados periodicamente (ex: a cada hora) para garantir que os dados de transmissão estejam sempre atualizados. Informações atualizadas sobre as transmissões são cruciais para a utilidade do sistema.
- Uso de Cloud Services: O sistema deve ser hospedado em uma infraestrutura de nuvem (como AWS, Azure ou GCP) para garantir escalabilidade e disponibilidade. A infraestrutura em nuvem oferece flexibilidade, escalabilidade automática e alta disponibilidade, essenciais para um serviço com potencial de grande volume de acessos.

2.5 Visão de Casos de Uso (Escopo do Produto)

O sistema “Onde é o Jogo” foi projetado para ajudar os usuários a encontrar informações sobre onde serão transmitidas as partidas dos campeonatos Brasileirão e

Libertadores. O site oferecerá uma interface intuitiva, fornecendo detalhes cruciais das partidas, como canal/plataforma de transmissão, horário e placar em tempo real. As funcionalidades principais incluem a visualização de partidas do dia, detalhes das partidas, personalização da home page com o time favorito do usuário, busca por time ou competição, e filtros de transmissão. Além disso, os usuários poderão criar contas, fazer login e configurar preferências de notificação.

A arquitetura MVC (Model-View-Controller) foi escolhida devido à necessidade de organizar dados complexos obtidos via web scraping e apresentá-los de forma rápida e clara. O uso de Next.js para o frontend permitirá criar uma experiência de usuário dinâmica e responsiva, essencial para a personalização e rapidez das interfaces. A separação de responsabilidades entre Model, View e Controller facilitará a manutenção e a adição de novas funcionalidades, garantindo que o sistema possa escalar conforme a demanda.

Para uma visão detalhada dos casos de uso, consulte o diagrama na página 13 da Declaração de Escopo do Produto. Esta escolha arquitetural foi influenciada pelos requisitos de desempenho, personalização e experiência do usuário, bem como pela experiência técnica da equipe com Next.js e web scraping.

2.6 Visão lógica

O sistema é subdividido nos seguintes módulos :

1. Web Scraping:
    - **Função**: Coleta dados de diferentes fontes sobre as partidas, como horários, canais e placares.
    - **Justificativa**: Automatiza a obtenção de informações atualizadas de fontes externas, garantindo que o sistema tenha dados precisos e em tempo real.
2. Entities:
    - **Função**: Representam os dados do sistema (times e jogos).
    - **Justificativa**: Mantêm a estrutura de dados organizada e permitem fácil manipulação e acesso às informações necessárias.

3. Services:
    - **Scraping Service**:
        - **Função**: Executa os scripts de scraping e processa os dados coletados.
        - **Justificativa**: Centraliza a lógica de coleta e processamento dos dados de transmissão.
    - **Match Service**:
        - **Função**: Contém a lógica de negócios para obter jogos por time, campeonato e canal de transmissão.
        - **Justificativa**: Gerencia a lógica de negócios relacionada às partidas, facilitando a busca e filtragem das informações.
4. Controllers:
    - **Home Controller**:
        - **Função**: Recebe requisições para a home page, obtendo os jogos do time preferido do usuário e passando-os para a view home.html.
        - **Justificativa**: Facilita a navegação inicial e a personalização da experiência do usuário.
    - **Team Controller**:
        - **Função**: Processa requisições para a página de um time específico, obtendo informações detalhadas sobre o time e seus jogos, e passando-os para a view team.js.
        - **Justificativa**: Fornece detalhes específicos sobre os times, melhorando a usabilidade e acessibilidade das informações.
    - **Match Controller**:
        - **Função**: Processa requisições para a página de um jogo específico, obtendo informações detalhadas sobre a partida e passando-as para a view match.js.
        - **Justificativa**: Garante que os usuários possam acessar informações detalhadas sobre qualquer partida.
5. Views:
    - **Templates Javascript**:
        - **Função**: Páginas renderizadas que exibem os dados ao usuário, como

home.js, team.js e match.js.

- **Justificativa**: Geram interfaces dinâmicas e responsivas, melhorando a experiência do usuário.

6. Comunicação entre os Módulos (Interfaces)
    - **Web Scraping ↔ Scraping Service**:
        - O módulo de Web Scraping coleta os dados e envia para o Scraping Service, que processa e armazena as informações.
    - **Scraping Service ↔ Entities**:
        - O Scraping Service atualiza os dados nas entidades correspondentes.
    - **Controllers ↔ Services**:
        - Os controladores fazem requisições aos serviços para obter dados específicos e enviá-los para as views.

- **Views ↔ Controllers**:
    - As views recebem dados dos controladores e os exibem ao usuário.

Diagrama de Estados:

O diagrama de estados detalha o fluxo da aplicação "Onde é o Jogo", mostrando como o usuário interage com diferentes partes do sistema desde o início até a consulta de informações e eventual logout. Cada estado representa uma etapa distinta no uso da aplicação, com transições que descrevem as ações do usuário.

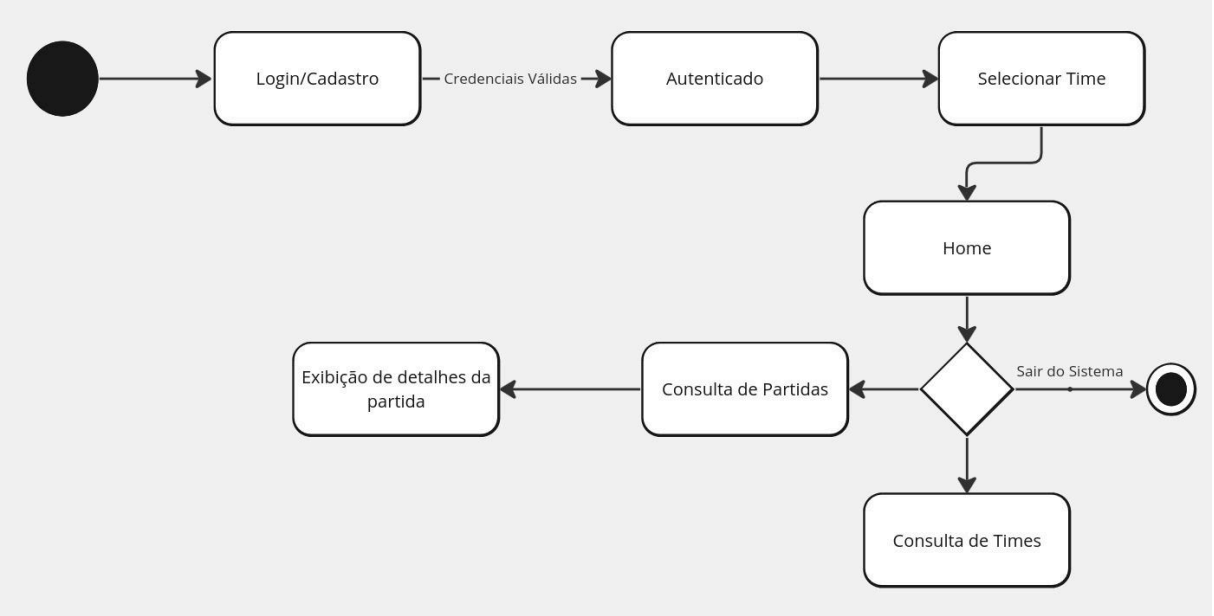

Figura 02: Diagrama de Estados

Explicação dos Estados:

1. **Iniciar Programa**:
    - **Descrição**: Estado inicial da aplicação, onde o programa é iniciado e a interface de login é apresentada ao usuário.
    - **Transição**: Se o login for bem-sucedido, o estado muda para "Autenticado".
2. **Autenticado**:
    - **Descrição**: Estado em que o usuário foi autenticado com sucesso após fazer login.
    - **Transição**: O usuário autenticado pode selecionar seu time favorito.
3. **Selecionar Time**:
    - **Descrição**: Estado em que o usuário escolhe seu time de futebol favorito.
    - **Transição**: Após a seleção do time, o estado muda para "Home".
4. **Home**:
    - **Descrição**: Estado em que a página inicial personalizada do usuário é exibida com informações relevantes sobre o time escolhido.
    - **Transição**:
        - O usuário pode sair do sistema, mudando o estado para "Sair do

Sistema".

- O usuário pode consultar informações sobre outros times, mudando o estado para "Consulta de Times".
- O usuário pode consultar informações sobre partidas, mudando o estado para "Consulta de Partidas".

5. **Sair do Sistema:**
    - **Descrição**: Estado em que o usuário escolhe fazer logout, encerrando a sessão e retornando ao estado inicial.
    - Transição: Retorna ao estado "Iniciar Programa".
6. **Consulta de Times:**
    - Descrição: Estado em que o usuário solicita informações sobre outros times.
    - Transição: As informações dos times são exibidas, mudando o estado para "Exibição de Times".
7. **Consulta de Partidas:**
    - **Descrição**: Estado em que o usuário solicita informações sobre partidas.
    - **Transição**: As informações das partidas são exibidas, mudando o estado para "Exibição de Partidas".
8. **Exibição de Times:**
    - **Descrição**: Estado em que as informações sobre os times são exibidas ao usuário.
    - **Transição**: O usuário pode retornar à página inicial, mudando o estado para "Home".
9. **Exibição de Partidas:**
    - **Descrição**: Estado em que as informações sobre as partidas são exibidas ao usuário.
    - **Transição**: O usuário pode retornar à página inicial, mudando o estado para "Home".

Diagrama de Atividades:

O diagrama de atividades detalha o fluxo de ações que o usuário pode realizar dentro da aplicação "Onde é o Jogo", desde o início até a consulta de informações e eventual logout.

Cada atividade representa uma etapa específica do processo, com transições que descrevem as ações do usuário.

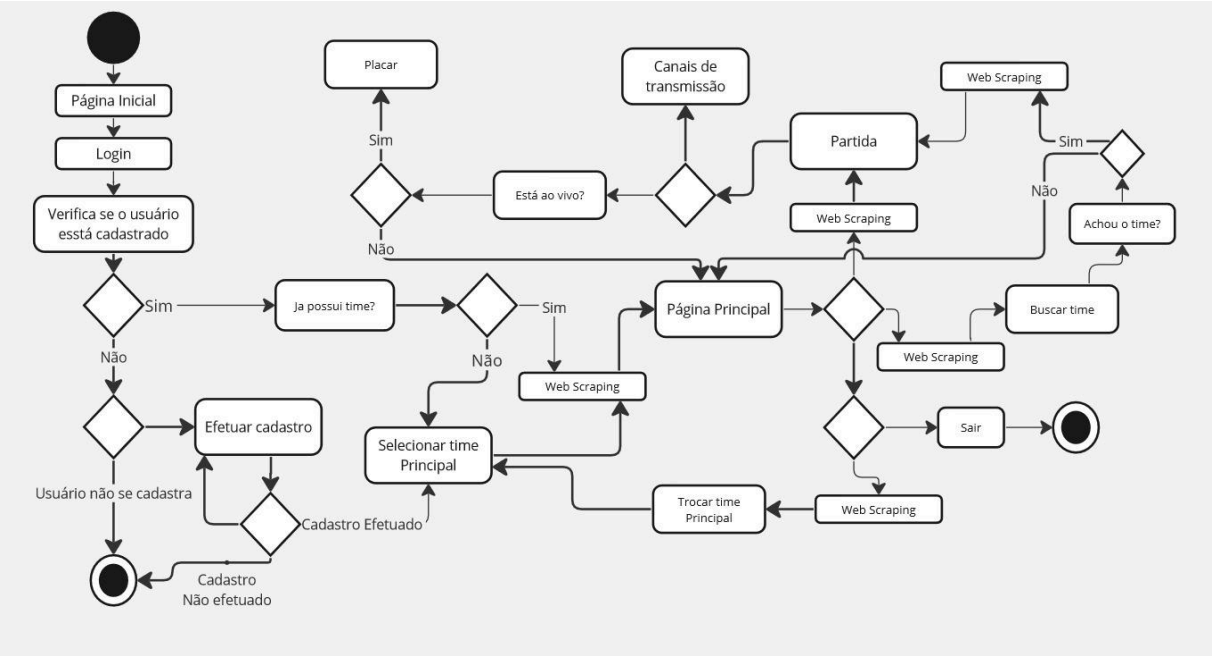

Figura 03: Diagrama de Atividades

Explicação das Atividades:

1. **Início**:
    - **Descrição**: Estado inicial da aplicação onde o usuário é direcionado à página de login.
2. **Login**:
    - **Descrição**: O usuário insere suas credenciais para autenticação.
3. **Usuário Autenticado?**:
    - **Descrição**: Verifica se o login foi bem-sucedido.
    - **Transição**: Se sim, verifica se o usuário já possui um time principal escolhido. Se não, exibe a página de cadastro/login.
4. **Possui Time Principal?**:
    - **Descrição**: Verifica se o usuário já escolheu um time principal.
    - **Transição**: Se sim, direciona para a página principal. Se não, direciona para a busca de time.

5. **Buscar Time**:
    - **Descrição**: Permite ao usuário buscar um time de futebol para saber sobre suas partidas ao vivo ou próximas partidas.
6. **Achou Time?**:
    - **Descrição**: Verifica se o time buscado foi encontrado.
    - **Transição**: Se sim, permite ao usuário selecionar o time. Se não, exibe uma mensagem de erro informando que o time não foi encontrado.
7. **Selecionar Time**:
    - **Descrição**: Permite ao usuário selecionar seu time favorito.
    - **Transição**: Direciona para a página principal.
8. **Página Principal**:
    - **Descrição**: Exibe a página inicial personalizada do usuário com informações relevantes sobre o time escolhido.
9. **Exibir Partida**:
    - **Descrição**: Exibe as informações sobre a próxima partida do time do usuário.
10. **Partida ao Vivo?**:
    - **Descrição**: Verifica se a partida está ao vivo.
    - **Transição**: Se sim, exibe o placar ao vivo. Se não, exibe os canais de transmissão.
11. **Exibir Placar ao Vivo**:
    - **Descrição**: Exibe o placar em tempo real da partida ao vivo.
12. **Exibir Canais de Transmissão**:
    - **Descrição**: Exibe os canais de transmissão da partida.
13. **Trocar Time Principal**:
    - **Descrição**: Permite ao usuário selecionar um novo time principal.
    - **Transição**: Direciona para a seleção de time.
14. **Logout**:
    - **Descrição**: O usuário faz logout, encerrando a sessão e saindo do sistema.
15. **Web Scraping**:
    - **Descrição**: será chamado sempre que precisar extrair dados de sites de forma automatizada, ou seja quando for exibir algum conteúdo / página relacionado a

jogos e times.

Diagrama de Classes:

A aplicação "Onde é o Jogo" é um sistema que permite aos usuários gerenciar preferências de times e campeonatos, visualizar partidas e consultar informações detalhadas sobre os eventos esportivos. O diagrama de classes detalha a estrutura do sistema, incluindo as entidades principais e as relações entre elas.

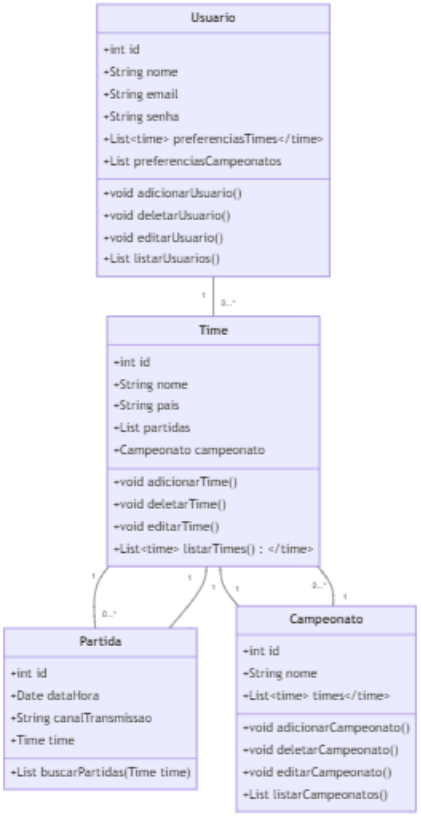

Figura 03: Diagrama de Classes

1. **Classe Usuário**
    - Atributos:
        - id (int): Identificador único do usuário.
        - nome (String): Nome do usuário.
        - email (String): Email do usuário.
        - senha (String): Senha do usuário.
        - preferenciasTimes (List<Time>): Lista de times favoritos do usuário.

        - preferenciasCampeonatos (List<Campeonato>): Lista de campeonatos favoritos do usuário.
    - Métodos:
        - adicionarUsuario(): Adiciona um novo usuário ao sistema.
        - deletarUsuario(): Remove um usuário existente do sistema.
        - editarUsuario(): Atualiza as informações de um usuário.
        - listarUsuarios(): Retorna uma lista de todos os usuários cadastrados.

2. **Classe Time**
    - Atributos:
        - id (int): Identificador único do time.
        - nome (String): Nome do time.
        - pais (String): País do time.
        - partidas (List<Partida>): Lista de partidas em que o time participa.
        - campeonato (Campeonato): Campeonato ao qual o time pertence.
    - Métodos:
        - adicionarTime(): Adiciona um novo time ao sistema.
        - deletarTime(): Remove um time existente do sistema.
        - editarTime(): Atualiza as informações de um time.
        - listarTimes(): Retorna uma lista de todos os times cadastrados.
3. **Classe Partida**
    - Atributos:
        - id (int): Identificador único da partida.
        - dataHora (Date): Data e hora em que a partida será realizada.
        - canalTransmissao (String): Canal de TV ou plataforma que transmitirá a partida.
        - time (Time): Time participante da partida.
    - Métodos:
        - buscarPartidas(Time time): Retorna uma lista de partidas de um determinado time.
4. **Classe Campeonato**
    - Atributos:

        - id (int): Identificador único do campeonato.
        - nome (String): Nome do campeonato.
        - times (List<Time>): Lista de times que participam do campeonato.
    - Métodos:
        - adicionarCampeonato(): Adiciona um novo campeonato ao sistema.
        - deletarCampeonato(): Remove um campeonato existente do sistema.
        - editarCampeonato(): Atualiza as informações de um campeonato.
        - listarCampeonatos(): Retorna uma lista de todos os campeonatos cadastrados.

5. **Relações entre Classes**
    - Usuario - Time: Os usuários têm uma lista de times preferidos (preferenciasTimes). Esta é uma relação de associação.
    - Usuario - Campeonato: Os usuários têm uma lista de campeonatos preferidos (preferenciasCampeonatos). Esta é uma relação de associação.
    - Time - Partida: Um time possui uma lista de partidas (partidas). Esta é uma relação de composição, indicando que as partidas dependem da existência do time.
    - Time - Campeonato: Um time pertence a um campeonato (campeonato). Esta é uma relação de agregação, indicando que um campeonato pode existir independentemente dos times que o compõem.
    - Partida - Time: Cada partida está associada a um time específico (time). Esta é uma relação de associação simples.
    - Campeonato - Time: Um campeonato possui uma lista de times (times). Esta é uma relação de agregação, indicando que um campeonato pode existir independentemente dos times que o compõem.

6. **Fluxo de Interação**
    - Cadastro e Login:
        - O usuário se cadastra no sistema (adicionarUsuario) e faz login.
    - Gestão de Preferências:
        - O usuário adiciona times e campeonatos às suas preferências (preferenciasTimes, preferenciasCampeonatos).

- Consulta de Informações:
  - O usuário pode consultar a lista de partidas de seus times preferidos (buscarPartidas).
  - O usuário pode listar todos os times (listarTimes) e campeonatos (listarCampeonatos) disponíveis.
- Administração do Sistema:
  - Administradores podem adicionar, editar e deletar usuários (adicionarUsuario, editarUsuario, deletarUsuario).
  - Administradores podem gerenciar times (adicionarTime, editarTime, deletarTime) e campeonatos (adicionarCampeonato, editarCampeonato, deletarCampeonato).

2.7 Visão de Implementação

Apresentação do Software

1. **Camada de apresentação**
   - **Título:** Onde é o Jogo.
   - **Menu de navegação:**
     - **Usuário:** Será uma parte para login, em que o usuário poderá definir sua prioridade de time.
     - **Times:** Nesse setor serão disponibilizadas informações sobre os times e seus futuros jogos.
     - **Partidas:** Contém informações sobre as partidas, o canal que irá transmitir e os horários dos jogos.
     - **Campeonatos:** Serão exibidas informações sobre os campeonatos, os times que estão participando e os jogos que serão realizados.
   - **Detalhes do jogo:** ao selecionar determinado jogo serão exibidas informações de data, horário e o canal de transmissão do mesmo.

2. **Lógicas de negócios e regras de negócios**
    - **Lógicas de negócios**
        - **Gestão do perfil do usuário:** Permite ao usuário estabelecer sua preferência de time. Assim tal preferência será armazenada em um banco de dados que possibilitará personalizar a página do usuário de acordo com suas preferências.
        - **Informações sobre os times:** Interface que apresenta informações como o nome do time, país de origem e campeonato que está participando, de diferentes times.
        - **Informações das partidas:** Informa detalhes dos jogos, contendo dia, horário e canal de transmissão.
        - **Informações dos campeonatos:** Apresenta uma lista de campeonatos disponíveis e os times que participam.

    - **Regras de negócios**
        - **Integração:** Atualizações regulares com as informações de campeonatos, times e jogos.
        - **Personalização da experiência do usuário:** Utiliza a preferência informada pelo usuário para personalizar sua página.

3. **Comunicação com o banco de dados**
    - **Tabela do banco de dados:**
        - **Usuário:**
            1. ID.
            2. Nome do usuário.
            3. Email.

4. Senha.
5. Preferência de time.

- **Times:**
  1. ID.
  2. Nome do time.
  3. País do time.
  4. Campeonato que participa.
- **Partidas:**
  1. ID.
  2. Data e hora da partida.
  3. Canal de transmissão.
- **Campeonatos:**
  1. ID.
  2. Nome do campeonato.
  3. Lista de ID’s dos times participantes.

- **Operações de CRUD:**
  - **Usuário:**
    1. Criar um novo usuário.
    2. Atualizar informações do usuário.
    3. Recuperar informações do usuário.
  - **Time:**
    1. Criar um novo time.
    2. Recuperar informações do time.
    3. Listar todos os times.
  - **Partidas:**
    1. Criar uma nova partida.
    2. Recuperar informações de uma partida.
    3. Listar todas as partidas.
  - **Campeonatos:**

1. Criar um novo campeonato.
2. Recuperar informações de um campeonato.
3. Listar todos os campeonatos.

- **Comunicação com banco de dados**
  - Utilização do JavaScript para comunicar com o banco de dados.
  - Para o gerenciamento de banco de dados, utilização do JavaScript para gerenciar e criar tabelas para o banco de dados.

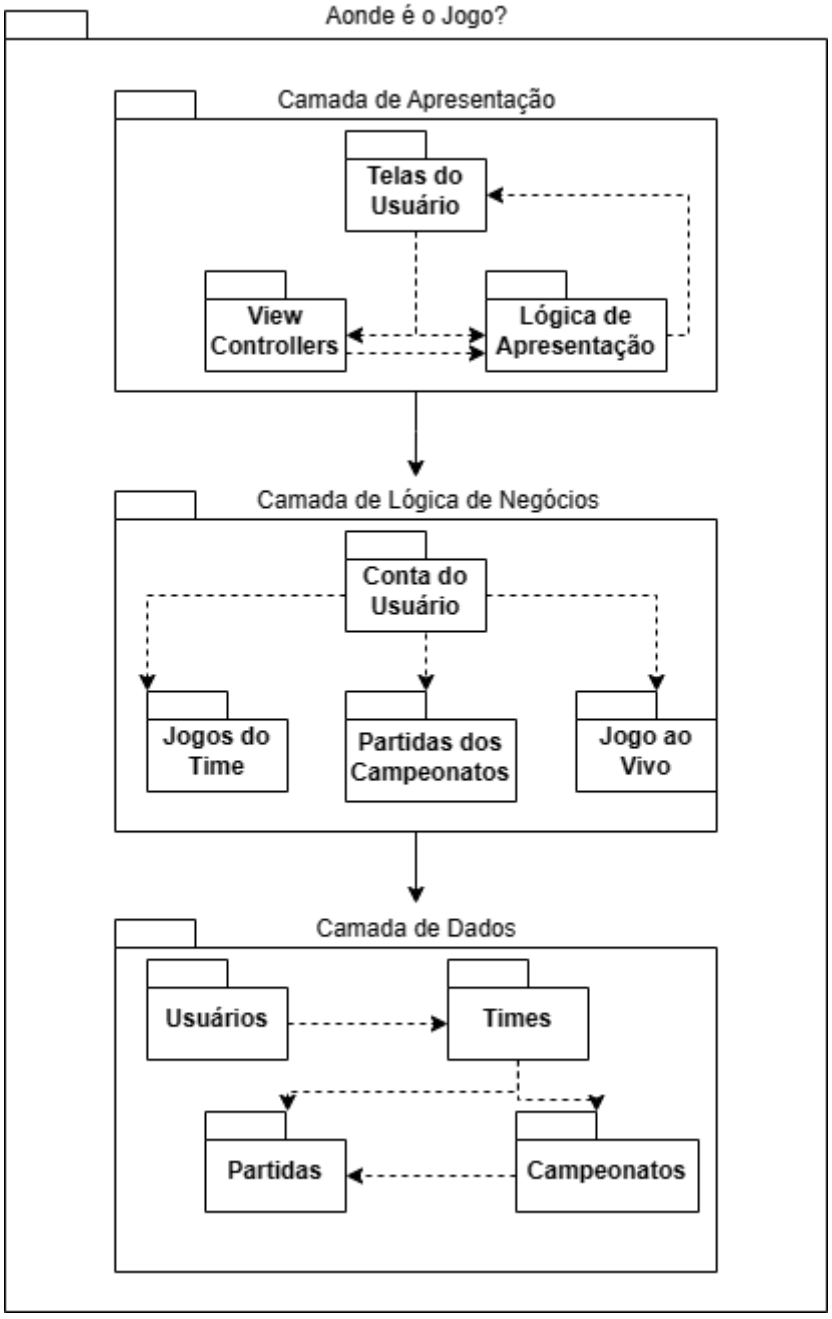

Figura 05: Diagrama de Pacotes

2.8 Visão de Implantação

Infraestrutura de Hardware para Implantação

A implantação do software se dará por uma infraestrutura baseada em nuvem, (como AWS, Azure ou GCP), essa decisão é baseada na facilidade de escalabilidade, disponibilidade e flexibilidade fornecidas por essas plataformas.

Tecnologias de Implantação

Para o front-end será utilizado um framework baseado em React, o Next.js, este framework possui uma característica de renderização híbrida de server-side rendering (SSR) e static site generation (SSG), essa renderização já nos garante o principal motivo de utilizar este framework, pois elas fornecem uma interface dinâmica e veloz.

O back-end será estruturado pelo node.js, um ambiente de execução de Javascript, que promove um ecossistema rico de funcionalidade, escalabilidade facilitada e uma boa performance.

Banco de Dados

O banco de dados utilizará o MySQL por conta da familiaridade que a equipe tem com a tecnologia, além dos benefícios que a mesma tem, como: segurança dos dados, alta escalabilidade e alta performance, o fornece um banco de dados robusto e viável para utilização.

2.9 Restrições Adicionais

- Solicitar que os usuários efetuem login para acessar os serviços do software para garantir a confiabilidade e portabilidade da plataforma, além da proposta da página inicial personalizada.
- O software deve ter uma fácil usabilidade do público alvo com interface intuitiva e limpa e essa característica será garantida através de testes de aceitação.
- Implementação de processos para validar e limpar os dados coletados através do Web Scraping, garantindo a qualidade dos dados assim atenderá as necessidades do software além de garantir um fluxo de informações para atender as necessidades do software.
- Garantir que o produto final seja acessível e de fácil utilização.

3. Bibliografia

VERCEL. Getting Started | Next.js. Disponível em: <https://nextjs.org/docs>.

KUGELL, A. NextJS Vs React: Choosing the Right Front-End Framework for 2024 - Trio. Disponível em: <https://trio.dev/nextjs-vs-react/>.

ROY, S. Node.JS as Backend Technology: Reasons and Trends in 2024. Disponível em: <https://minditsystems.com/node-js-as-backend-technology-reasons-and-trends-in-2024/>.

MCLEAN, M. Secrets of Using Quality Metrics to Manage DevOps Release Risk. Disponível em: <https://devops.com/8-advantages-using-mysql/>.