RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Uni-FACEF Centro Universitário Municipal de Franca
Vol. 13 Nro. 1 2023

BOLA AO GRÁFICO: desenvolvimento de uma aplicação para visualizar dados provenientes de jogos de basquete

Vinícius Poloni Crisol
Graduado em Ciência da Computação - Uni-FACEF
vinicius.crisol@hotmail.com

Carlos Eduardo de França Roland
Docente do Uni-FACEF
roland@facef.br

Resumo
O crescente interesse dos fãs de basquete por estatísticas têm impulsionado ligas, como a NBA, a investir cada vez mais em processos de distribuição de dados ao público. Contudo, apesar dessas iniciativas, os analistas especializados ainda se deparam com desafios ao minerar e visualizar essas informações. Muitos, ao tentarem criar visualizações personalizadas, são forçados a utilizar planilhas preenchidas manualmente, um processo não só ineficaz, mas também propenso a falhas. Diante desses desafios, este trabalho visa apresentar um sistema projetado para sanar estas dificuldades. No seu desenvolvimento, foram utilizadas técnicas de *UI Design*, Engenharia de Software e Ciência de Dados. Com os devidos aperfeiçoamentos, os artefatos obtidos podem evoluir para um sistema comercializável.

Palavras-chave: Aplicação *web*. Visualização de dados. Basquete

Abstract
The growing interest of basketball fans in statistics has driven leagues, such as the NBA, to increasingly invest in data distribution processes to the public. However, despite these initiatives, specialized analysts still face challenges in mining and visualizing this information. Many, when trying to create customized visualizations, are compelled to use manually-filled spreadsheets, a process that is not only inefficient but also error-prone. In light of these challenges, this study aims to introduce a system designed to address these difficulties. In its development, techniques in UI Design, Software Engineering, and Data Science were employed. With the appropriate refinements, the produced artifacts can evolve into a marketable system.

Keywords: *Web application. Data visualization. Basketball.*

1 Introdução

É cada vez mais comum que espectadores de basquete utilizem estatísticas para compreender melhor os acontecimentos do esporte. Com isso, ligas esportivas como a National Basketball Association (NBA) trabalham ativamente em mecanismos de distribuição de dados ao público geral. No entanto, mesmo com esses esforços, a comunidade especializada ainda enfrenta desafios relacionados à mineração e visualização de informações.

De forma geral, quando um analista deseja criar visualizações personalizadas, utilizando os dados disponibilizados pelas ligas, ele é obrigado a recorrer a planilhas preenchidas manualmente. Esse processo, além de ser ineficaz, está sujeito a falhas. Diante dessas dificuldades, o objetivo deste trabalho é apresentar

um sistema voltado para a comunidade especializada em basquete, que visa otimizar os processos de análise e visualização estatísticas.

Durante a idealização do sistema, foram utilizadas técnicas de *UI Design* (*Design* de Interface do Usuário), juntamente com conceitos da Engenharia de Software e Ciência de Dados. Ao final, a apresentação deste trabalho exibe, juntamente com a análise dos resultados obtidos, ideias para uma nova versão.

2 Referencial Teórico

Nesta seção, são apresentados os principais conceitos, ferramentas e metodologias empregadas na execução do projeto.

2.1 Estatísticas no basquetebol

As estatísticas desempenham um papel fundamental no esporte, uma vez que permitem mensurar o desempenho dos atletas e equipes, avaliar estratégias e identificar pontos fortes e fracos em sistemas táticos.

O basquetebol, enquanto um jogo de cooperação e oposição, oferece uma vasta gama de situações passíveis de análise (MORENO, 1994). Times profissionais contam com setores especializados dedicados ao estudo minucioso dos dados coletados durante as partidas, já que essas informações desempenham um papel essencial na fundamentação das decisões estratégicas (STEINBERG, 2015). Este processo é conhecido como tomada de decisão orientada a dados, ou em inglês, *data-driven decision-making*.

Um exemplo célebre da aplicação dessa técnica é encontrado no livro *Moneyball*, escrito por Michael Lewis em 2003. Na obra, o autor explora como o uso inteligente das estatísticas revolucionou a forma como as equipes abordam as decisões tanto dentro quanto fora das quadras.

No entanto, não são apenas as organizações esportivas que utilizam as estatísticas como fonte de conhecimento. Torcedores estão cada vez mais interessados em consumir conteúdo analítico, buscando compreender melhor as nuances do jogo por meio dos números. Por essa razão, associações esportivas estão empenhadas em criar mecanismos de distribuição de informações ao público geral (STEINBERG, 2015).

Como resultado desse crescente interesse, transmissões televisivas têm incorporado análises estatísticas por meio de gráficos e infográficos, com o intuito de proporcionar aos espectadores uma visualização clara e acessível dos eventos ocorridos em quadra. Isso permite que a audiência interprete de forma clara e embasada em dados o desempenho dos atletas e equipes.

2.2 *Application Programming Interface* (API)

API é a sigla, em inglês, para o termo Interface de Programação de Aplicativos, que se refere a um conjunto de protocolos e definições que permitem a comunicação entre diferentes componentes de software. Uma API essencialmente fornece um contrato de como os módulos de um programa devem interagir entre si (MULESOFT, [S. D.]).

2.3 Python

A linguagem de programação Python foi criada por Guido van Rossum, matemático holandês, nos anos 90. Foi concebida com o propósito de aprimorar a legibilidade dos códigos e fomentar a produtividade dos usuários.

Conforme descrito por Kriger (2022), Python é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada a objetos. Por ser uma linguagem de sintaxe relativamente simples e de fácil compreensão, ganhou popularidade entre profissionais da indústria tecnológica, incluindo engenheiros, matemáticos, cientistas de dados e pesquisadores. Um de seus maiores atrativos é a vasta quantidade de bibliotecas de terceiros disponíveis, que facilita a criação de programas com a linguagem.

2.4 *Web scraping*

Web scraping é um método usado para extrair dados de páginas *web* de maneira automatizada. O processo é amplamente utilizado em uma variedade de contextos digitais, incluindo análise de dados, monitoramento de preços e muito mais (MITCHELL, 2018).

2.5 Go

A linguagem de programação Go, foi criada em 2007 por Robert Griesemer, Rob Pike e Ken Thompson. O projeto foi concebido pela Google com o objetivo de desenvolver uma linguagem simples e altamente performática.

> Go ostenta uma semelhança superficial com C e, como ela, é uma ferramenta para programadores profissionais, alcançando o máximo de efeito com o mínimo de recursos. Porém, Go é muito mais que uma versão atualizada de C. Ela empresta e adapta boas ideias de várias outras linguagens, ao mesmo tempo que evita funcionalidades que resultaram em complexidade e em códigos não confiáveis. Seus recursos para concorrência são novos e eficientes, e sua abordagem para abstração de dados e programação orientada a objetos é surpreendentemente flexível. Ela tem gerenciamento de memória automático, isto é, coleta de lixo (*Garbage Collection*) (DONAVAN e KERNIGHAN, 2017, p. 13).

2.6 JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada, e dinâmica. É reconhecida principalmente pela sua utilização em páginas *web*. A linguagem permite a manipulação do Document Object Model (DOM) para criar efeitos e funcionalidades variadas. Além do lado do cliente, JavaScript também é utilizado em servidores através de ambientes como o Node.js (FLANAGAN, 2013).

2.7 *UI Design*

UI Design, abreviação de *User Interface Design* (*Design* de Interface do Usuário), é uma disciplina do *design* que se dedica à criação de interfaces intuitivas, funcionais e visualmente atraentes para sistemas, aplicativos e *websites*. Seu objetivo principal é proporcionar aos usuários uma experiência satisfatória e eficiente ao interagir com um produto digital (SYOSI, 2023).

O *UI Design* desempenha um papel fundamental na criação de interfaces que atendam às necessidades dos usuários, visando uma maior usabilidade e

satisfação. Aspectos como acessibilidade e consistência visual são considerados essenciais para garantir que as interfaces sejam visualmente agradáveis e de fácil navegação.

2.7.1 Rabiscoframe

Dentro do campo do *UI Design*, o termo rabiscoframe é uma expressão bem-humorada que combina as palavras rabisco e *wireframe*. Essa expressão se refere a uma fase inicial e informal do processo de *design* de interfaces, na qual os *designers* criam esboços rápidos e simplificados, sem se preocupar com detalhes estéticos específicos (TEIXEIRA, 2010).

Esses esboços, conhecidos como rabiscos, têm o propósito de visualmente representar a estrutura e a organização básica dos elementos da interface, de forma semelhante a um *wireframe*. O objetivo do rabiscoframe é capturar e explorar ideias de *design* de maneira ágil e flexível, permitindo que os *designers* visualizem e experimentem diferentes conceitos antes de seguirem para etapas mais avançadas do processo de *design*.

2.7.2 *Wireframe*

Os *wireframes* são representações visuais em escala de cinza que definem a estrutura e o fluxo de navegação de um sistema. Essa etapa inicial do processo de prototipagem tem como objetivo estabelecer o esqueleto da interface, fornecendo uma visão clara da disposição dos elementos e da arquitetura das informações, bem como da navegação entre telas, antes da inclusão do conteúdo final (LUCIDCHART, [S. D.]).

Os *wireframes* desempenham um papel crucial ao permitir que os *designers* e as equipes de desenvolvimento visualizem e avaliem a interação do usuário, facilitando a identificação de problemas de usabilidade e a realização de ajustes desde as fases iniciais do projeto. Essa prática desempenha um papel fundamental no desenvolvimento de produtos digitais bem projetados, fornecendo uma base sólida para a criação de interfaces intuitivas e eficientes.

2.7.3 Protótipo de alta fidelidade

No campo do *UI*, um protótipo de alta fidelidade desempenha um papel fundamental ao oferecer uma representação interativa do produto que se aproxima do *design* final em termos de detalhes e funcionalidade (TERA, [S. D.]).

Esse tipo de protótipo proporciona às partes interessadas uma visão antecipada e tangível da interface final, permitindo uma melhor compreensão e avaliação do produto em desenvolvimento.

3 Desenvolvimento do projeto

Esta seção descreve o processo de desenvolvimento do projeto, abordando as metodologias e ferramentas aplicadas durante sua execução.

3.1 Implementação do software

O sistema foi estruturado em três bases de código distintas, cada uma delas representando um módulo específico do software. A seguir, serão apresentados alguns detalhes da implementação de cada um destes componentes.

3.1.1 Importação de dados

Este módulo é responsável por coletar e persistir no banco de dados informações cadastrais dos atletas, bem como estatísticas individuais e coletivas das partidas. Tais dados são essenciais para outros componentes do sistema, sendo utilizados na criação de listas, relatórios e visualizações estatísticas.

Um dos principais focos durante o planejamento do sistema foi minimizar o acoplamento entre a origem e o local de consumo dos dados. Isso aumentou a flexibilidade do software, permitindo que informações de qualquer competição sejam incorporadas ao programa, por meio da simples substituição deste módulo.

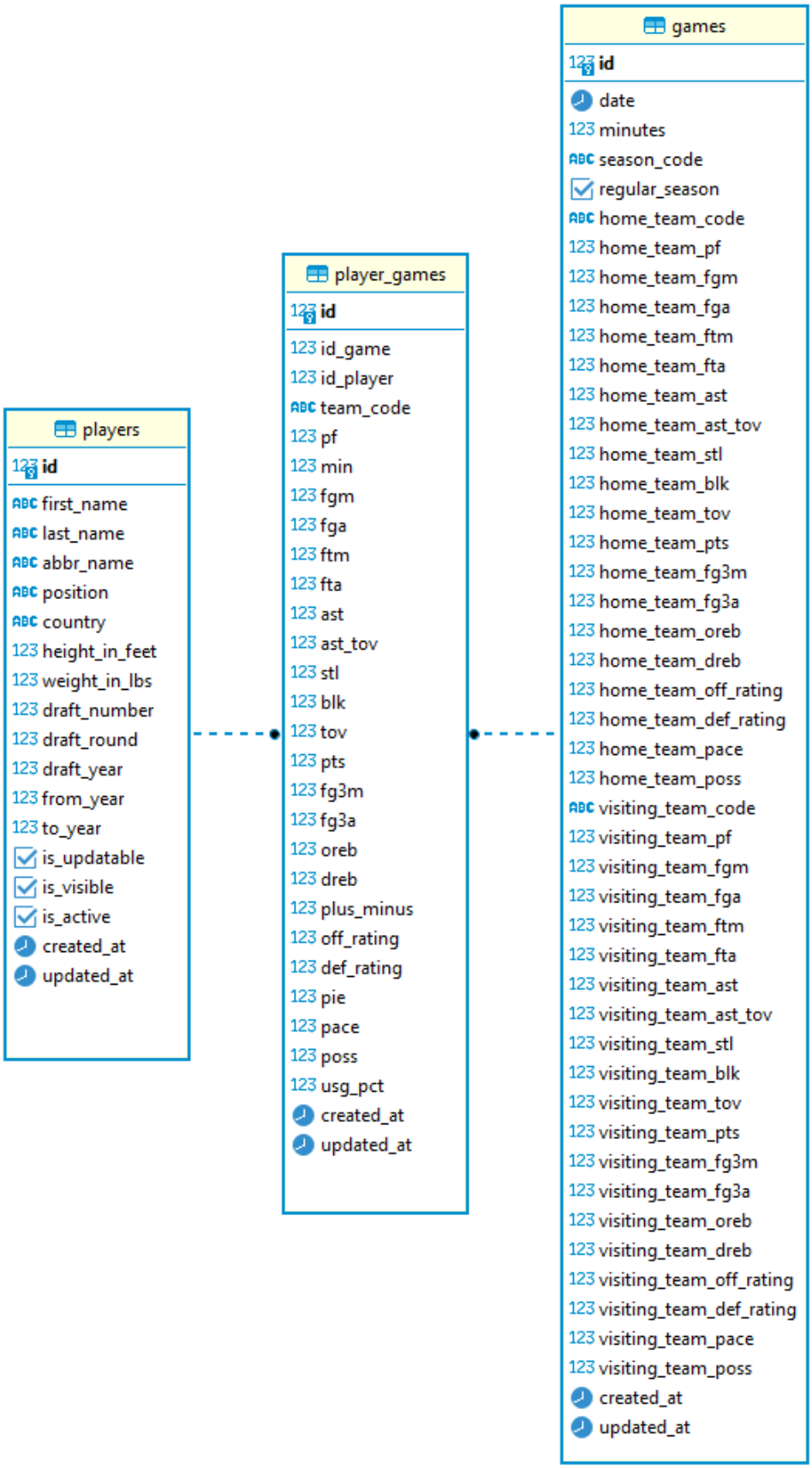
A NBA foi utilizada como fonte de dados para a construção do software, devido à quantidade considerável de dados disponibilizados ao público. Entretanto, como a liga não fornece API para a integração de seus dados com sistemas de terceiros, a interface de importação foi implementada utilizando a linguagem de programação Python. Essa escolha foi motivada pela existência da biblioteca de código aberto NBA_API, que apesar do nome é mantida pela comunidade de desenvolvedores e não tem nenhum vínculo oficial com a NBA. A ferramenta facilita a coleta de informações do site oficial da liga, abstraindo processos de *web scraping*.

A rotina foi programada para ser executada diariamente às 5:00 da manhã. Durante este processo, o software primeiramente atualiza as informações cadastrais dos jogadores, realizando uma comparação com os dados existentes na base de dados. Concluída esta etapa, as estatísticas individuais e coletivas das partidas do dia anterior são persistidas na base de dados, assim, marcando a finalização do programa. Tal processo assegura que os usuários sempre interajam com as informações mais atualizadas da liga.

Para garantir a integridade do programa, foram desenvolvidos testes unitários para os principais métodos da aplicação. Estes, têm o papel de detectar quebras na lógica do programa em casos de alteração do código-fonte.

A Figura 1 demonstra a organização desses dados no Sistema Gerenciador de Banco de Dados (SGBD). Enquanto as Figuras 2 e 3 apresentam fragmentos da implementação deste módulo, exibindo trechos da importação de jogadores e partidas, respectivamente.

Figura 1 - Estrutura dos dados importados



Fonte: os autores

Figura 2 - Importação de jogadores

```
class ImportPlayersUsecase(object):
    def __init__(self, players_repository):
        self.players_repository = players_repository

    def execute(self):
        stored_players = self.players_repository.get_stored_players()
        nba_api_player_ids = self.players_repository.get_nba_api_player_ids()

        for id in nba_api_player_ids:
            stored_player = stored_players.get(id)
            if (
                    stored_player is not None and
                    (stored_player.is_visible is False or stored_player.is_updatable is False)
            ):
                continue

            nba_api_player = self.players_repository.get_nba_api_player(id)
            if (
                    stored_player is None or
                    self.is_player_updated(stored_player, nba_api_player) is False
            ):
                self.save_player(nba_api_player)

    def save_player(self, nba_api_player):
        is_visible = self.is_player_visible(nba_api_player)
        is_updatable = self.is_player_updatable(is_visible)
        self.players_repository.save_player(
                app.Player(
                        id=nba_api_player.id,
                        first_name=nba_api_player.first_name,
                        last_name=nba_api_player.last_name,
                        abbr_name=nba_api_player.abbr_name,
                        position=nba_api_player.position,
                        country=nba_api_player.country,
                        height_in_feet=nba_api_player.height_in_feet,
                        weight_in_lbs=nba_api_player.weight_in_lbs,
                        draft_number=nba_api_player.draft_number,
                        draft_round=nba_api_player.draft_round,
                        draft_year=nba_api_player.draft_year,
                        from_year=nba_api_player.from_year,
                        to_year=nba_api_player.to_year,
                        is_updatable=is_updatable,
                        is_visible=is_visible,
                        is_active=nba_api_player.is_active
                )
        )

    @staticmethod
    def is_player_visible(nba_api_player):
        return nba_api_player.to_year is None or nba_api_player.to_year >= 1996

    @staticmethod
    def is_player_updatable(is_visible):
        return is_visible
```

Fonte: os autores

Figura 3 - Importação de partidas

```
class ImportGamesUsecase(object):
    def __init__(self, games_repository, players_repository):
        self.games_repository = games_repository
        self.players_repository = players_repository

    def execute(self, date, season_code):
        nba_api_games = self.games_repository.get_nba_api_games(date, season_code)

        for nba_api_game in nba_api_games:
            team_box_score, player_box_score = self.games_repository.get_box_scores(nba_api_game.id)
            self.save_game(nba_api_game, team_box_score)
            self.save_player_games(nba_api_game, player_box_score)

    def save_game(self, nba_api_game, team_box_score):
        app_game = app.Game(...)
        self.games_repository.save_game(app_game, box_score)

    def save_player_games(self, nba_api_game, player_box_score, stored_players):
        for box_score in player_box_score:
            if stored_players.get(box_score.id_player) is None:
                continue

            nba_api_player_game = self.games_repository.get_nba_api_player_game(
                nba_api_game.id,
                box_score.id_player,
            )
            if nba_api_player_game is not None:
                self.save_player_game(nba_api_player_game, box_score)

    def save_player_game(self, nba_api_player_game, box_score):
        self.games_repository.save_player_game(app.PlayerGame(...))
```

Fonte: os autores

3.1.2 *Back-end*

O *back-end* da aplicação, desenvolvido utilizando a linguagem de programação Go, tem como função se comunicar com o *front-end* fornecendo e recebendo dados por meio de requisições Hypertext Transfer Protocol (HTTP).

As informações usadas na criação de listas, relatórios e visualizações estatísticas derivam de *materialized views*, as quais fazem uso dos dados importados pelo componente mencionado no tópico anterior. Estas estruturas foram implementadas no banco de dados para aprimorar a eficiência das *queries*. Sendo representações armazenadas de consultas, elas evitam a execução repetida da mesma *query*, poupando tempo e recursos computacionais. No contexto do sistema, isso se mostra especialmente útil ao acelerar significativamente a entrega de dados para o *front-end*, resultando em uma melhor experiência para o usuário.

A Figura 4 apresenta a organização desses dados no SGBD, evidenciando a distribuição das informações consumidas e persistidas pelo *back-end*. Das seis estruturas exibidas, as quatro primeiras representam visualizações, enquanto as duas últimas, constituem em tabelas convencionais.

Figura 4 - Estrutura dos dados do *back-end*

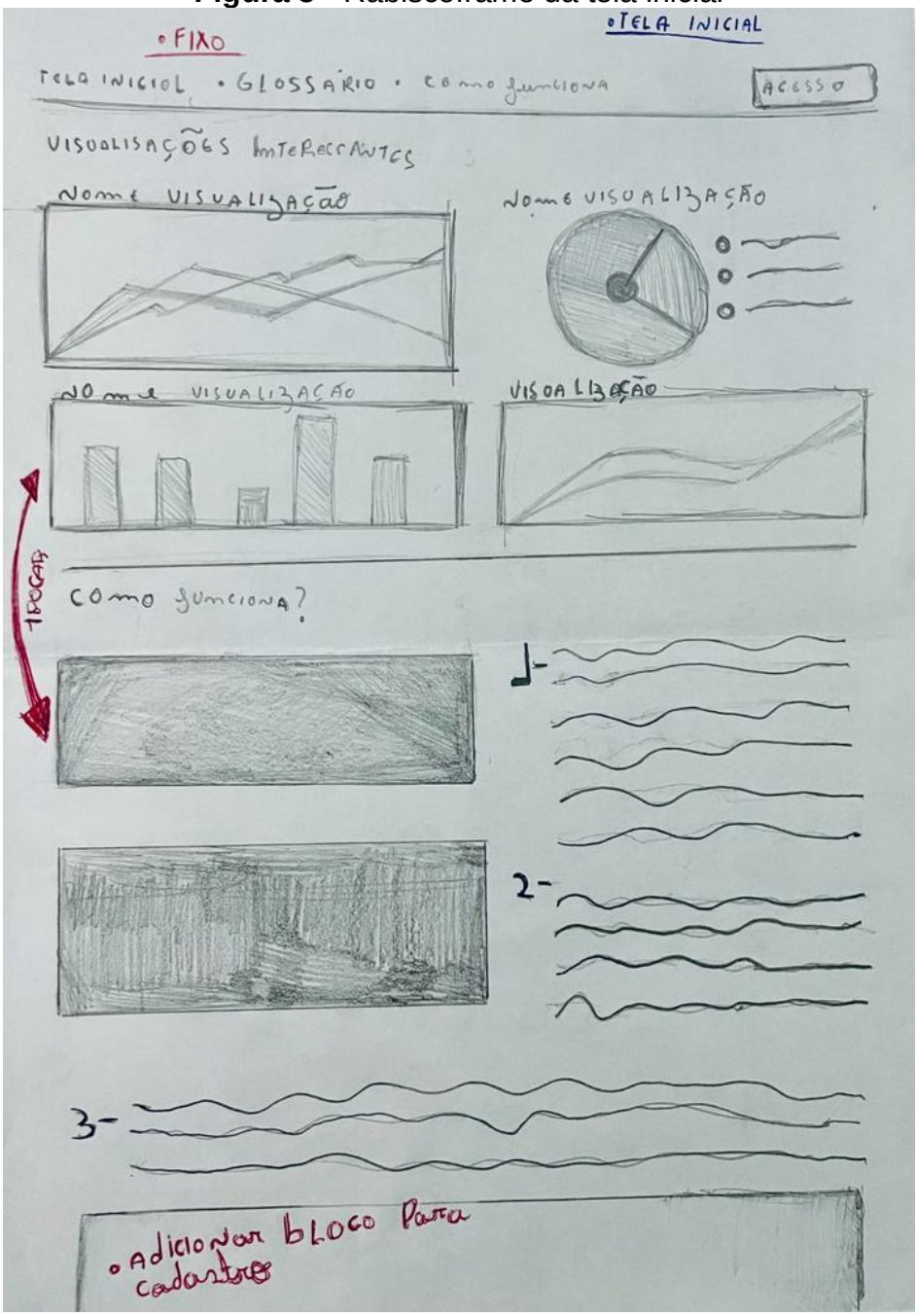
Fonte: os autores

3.1.3 *Front-end*

O desenvolvimento do *front-end*, componente encarregado de interagir diretamente com os usuários, foi feito com base no *framework* JavaScript Next.js. Essa ferramenta foi escolhida devido à sua versatilidade e eficiência na construção de aplicações *web*.

O processo de *design* da interface do usuário ocorreu de forma incremental, permitindo o refinamento contínuo dos elementos gráficos. Na fase inicial, rabiscoframes foram elaborados para as principais telas do sistema. Os esboços da tela inicial são representados na Figura 5.

Figura 5 - Rabiscoframe da tela inicial



Fonte: os autores

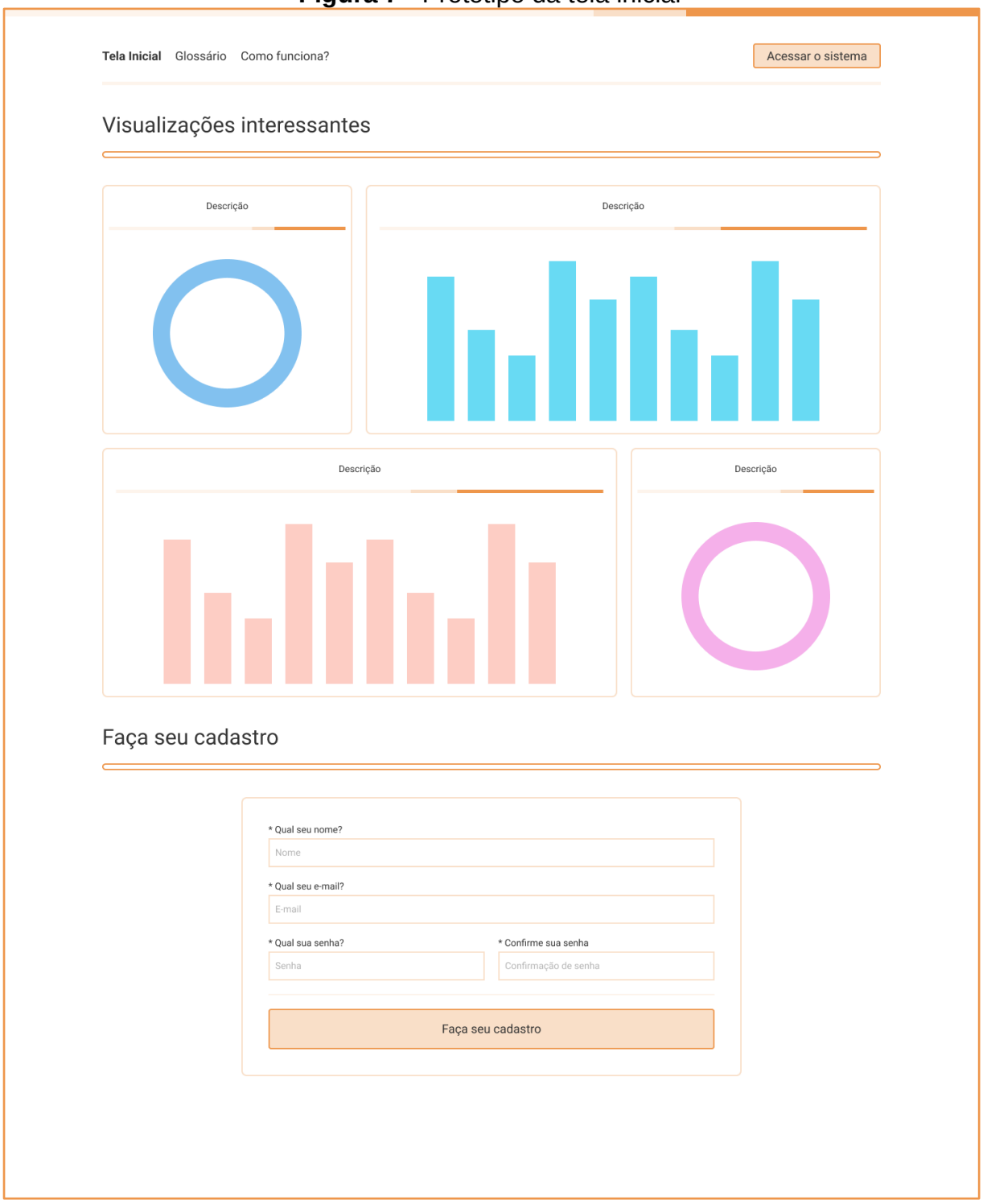
Na etapa seguinte da prototipação, os rascunhos gerados anteriormente serviram como referência para a construção dos *wireframes* de todas as telas do sistema (Figura 6). Para a criação do *design* das telas, utilizou-se o Figma, uma ferramenta *online* focada no *design* de interfaces *web* e *mobile*.

Figura 6 - *Wireframe* da tela inicial

Fonte: os autores

Por fim, na última etapa do *design* de interfaces, o protótipo de alta fidelidade foi feito baseando-se nos resultados das etapas anteriores. A Figura 7 exemplifica o produto final. Assim como na etapa anterior, o Figma foi usado no processo de concepção das telas.

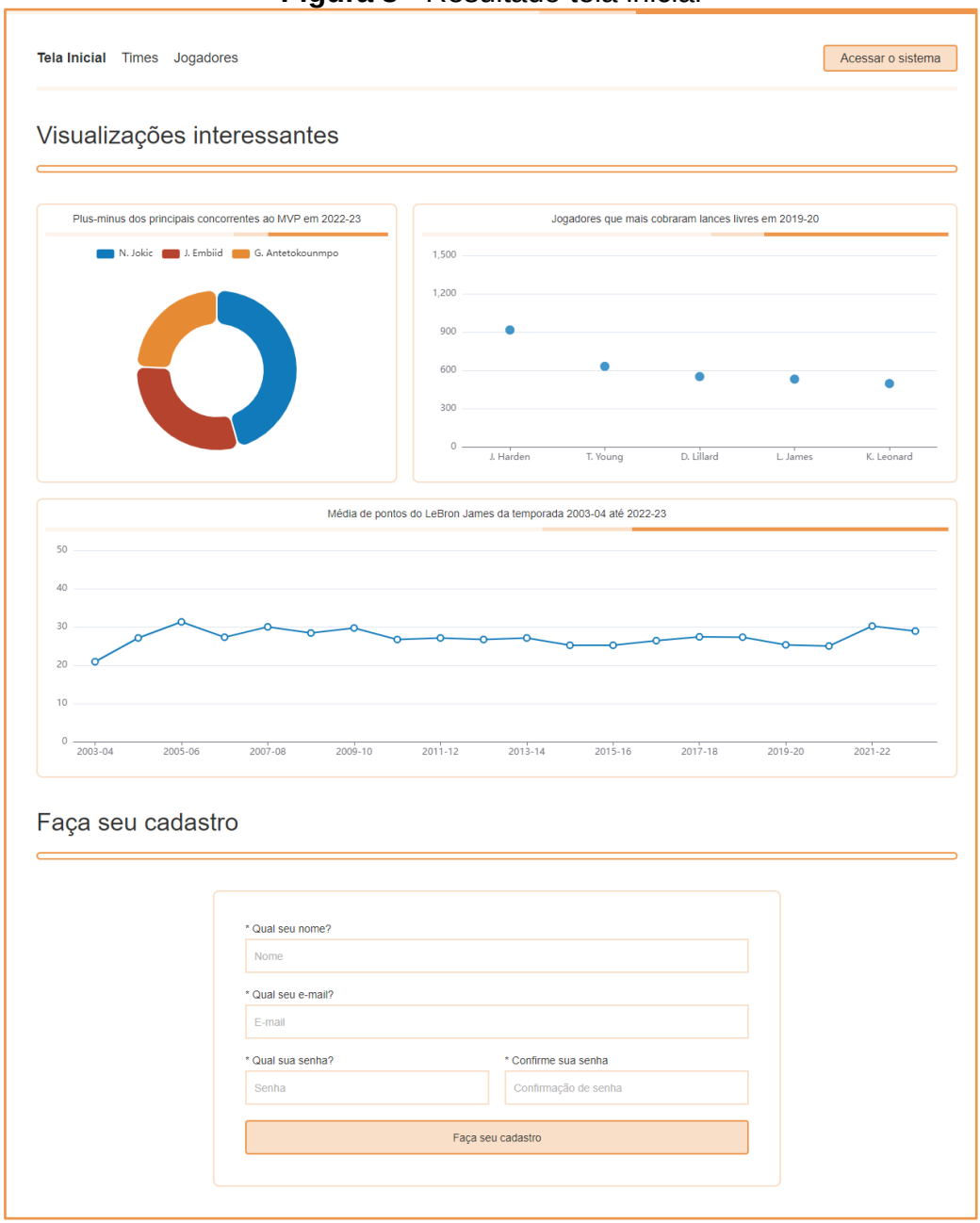
Figura 7 - Protótipo da tela inicial



Fonte: os autores

Após a codificação do *design* proposto, o resultado é um *front-end* funcional e visualmente agradável. A Figura 8 apresenta uma captura da tela inicial, demonstrando uma das interfaces finais com a qual os usuários irão interagir.

Figura 8 - Resultado tela inicial



Fonte: os autores

A biblioteca Apache ECharts foi empregada para a criação das visualizações estatísticas. Esta biblioteca JavaScript de código aberto foi selecionada devido à sua flexibilidade na criação de visualizações personalizadas.

4 Resultados da solução proposta

Neste capítulo, são apresentados os resultados obtidos, exibindo fragmentos do fluxo de cada uma das telas desenvolvidas.

4.1 Tela inicial

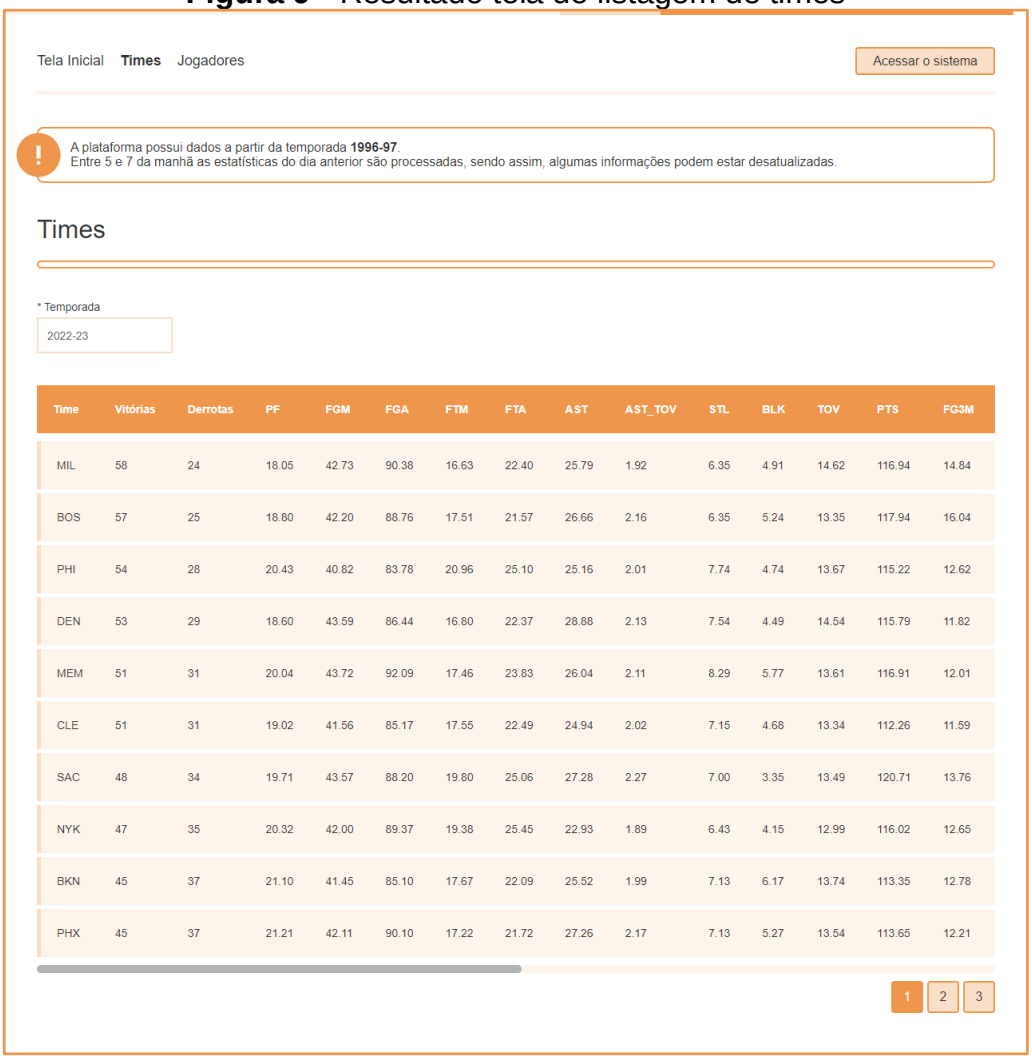
Conforme ilustrado previamente na Figura 8, a página inicial é composta por duas seções distintas. A primeira fornece uma exibição resumida de algumas estatísticas, oferecendo ao usuário uma visão geral das capacidades do sistema. A segunda parte, localizada mais abaixo na interface, fornece a opção de cadastro na plataforma.

4.2 Listagens

No menu superior, ao clicar em Times ou Jogadores, o usuário é redirecionado para as páginas de listagem que contém estatísticas coletivas ou individuais, respectivamente. Essas telas oferecem uma visão geral do desempenho dos integrantes, apresentando uma ampla variedade de informações que podem ser utilizadas para comparar a performance geral dos mesmos. Ao selecionar uma linha específica, o usuário é encaminhado para o relatório correspondente.

A Figura 9 exemplifica as telas de listagem, exibindo a interface de equipes.

Figura 9 - Resultado tela de listagem de times



Fonte: os autores

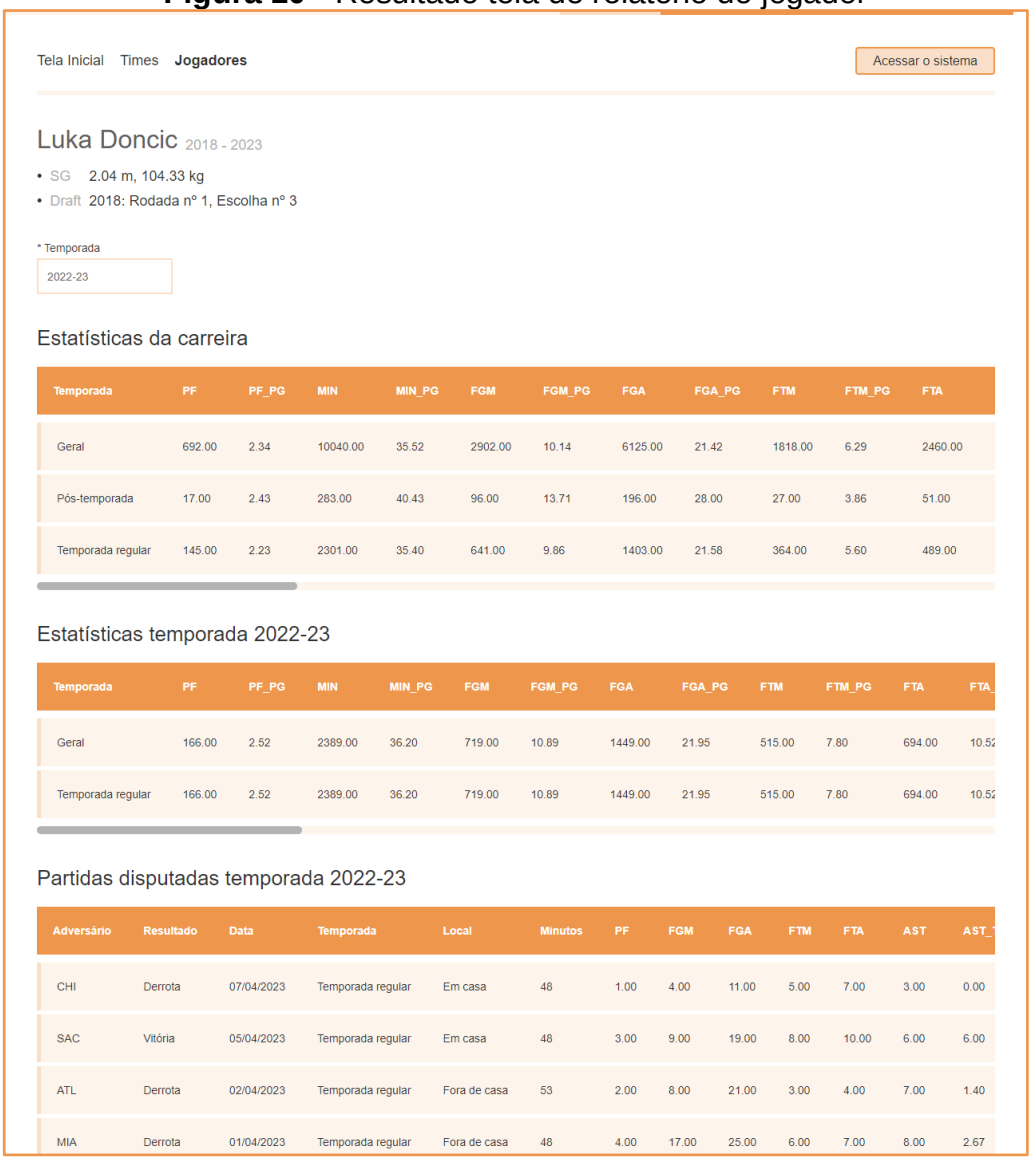
4.3 Relatórios

Assim como apresentado previamente, ao selecionar uma linha na listagem de jogadores ou equipes, o usuário é redirecionado para o relatório correspondente. Em linhas gerais, o relatório de equipes possui muitas semelhanças com o de jogadores. Na visão dos times, são apresentadas as estatísticas gerais, segmentadas por temporada regular e pós-temporada, bem como um resumo abrangente de ambas as categorias. A listagem das partidas disputadas também é mostrada, contendo as estatísticas pertinentes à exibição.

No caso do relatório de jogadores, além das informações cadastrais, como peso e altura do atleta, há um resumo das estatísticas de toda sua carreira, e não apenas da temporada selecionada. O relatório também inclui detalhes estatísticos individuais dos jogos disputados, oferecendo uma visão mais completa e personalizada das atuações do jogador ao longo das partidas.

A Figura 10 exemplifica as telas de relatório, exibindo a interface de jogadores.

Figura 10 - Resultado tela de relatório do jogador

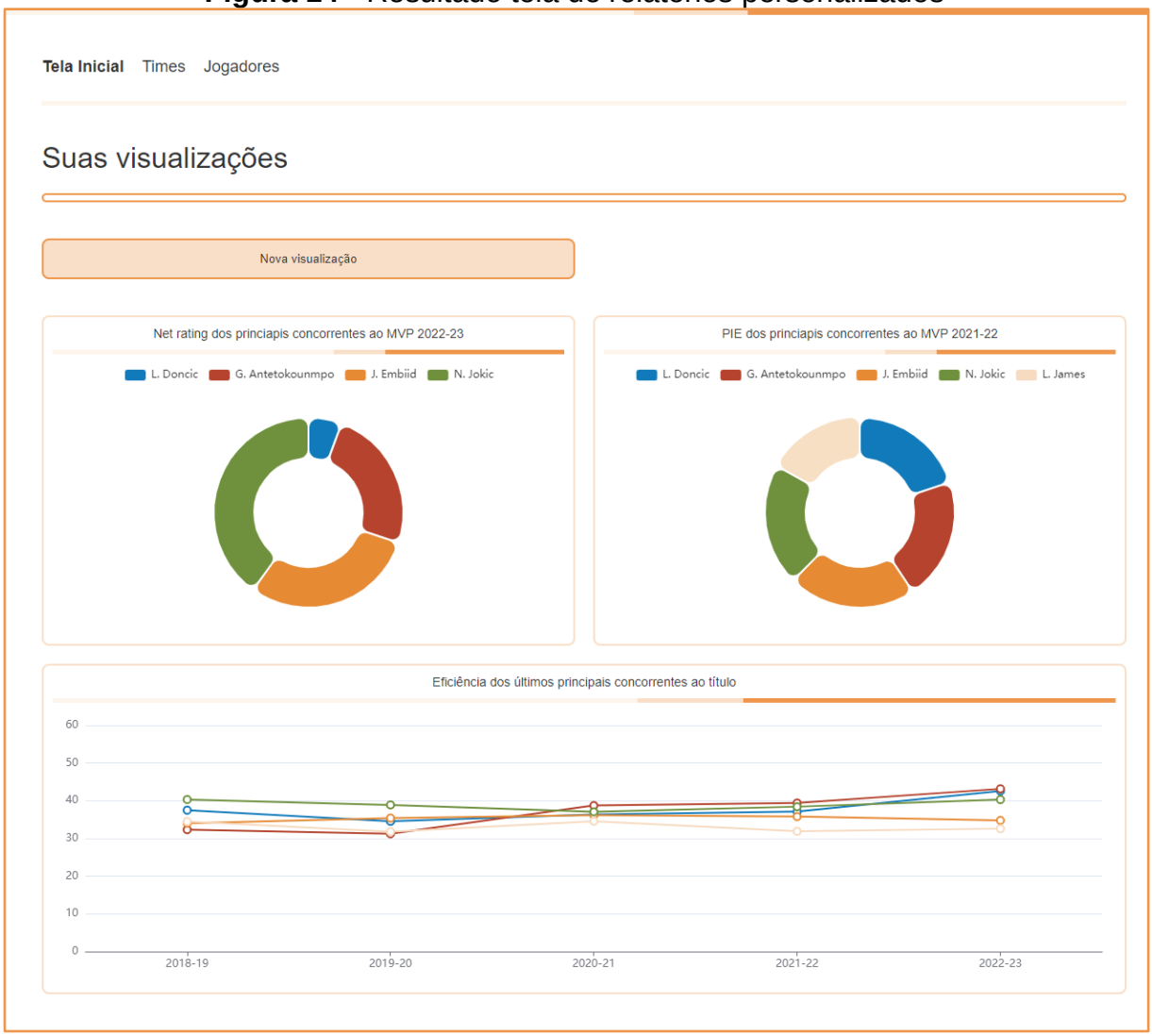


Fonte: os autores

4.4 Relatórios personalizados

A Figura 14 ilustra a tela de relatórios personalizados, cujo objetivo é exibir as visualizações criadas pelo usuário. O acesso a esta tela requer autenticação no sistema.

Figura 14 - Resultado tela de relatórios personalizados

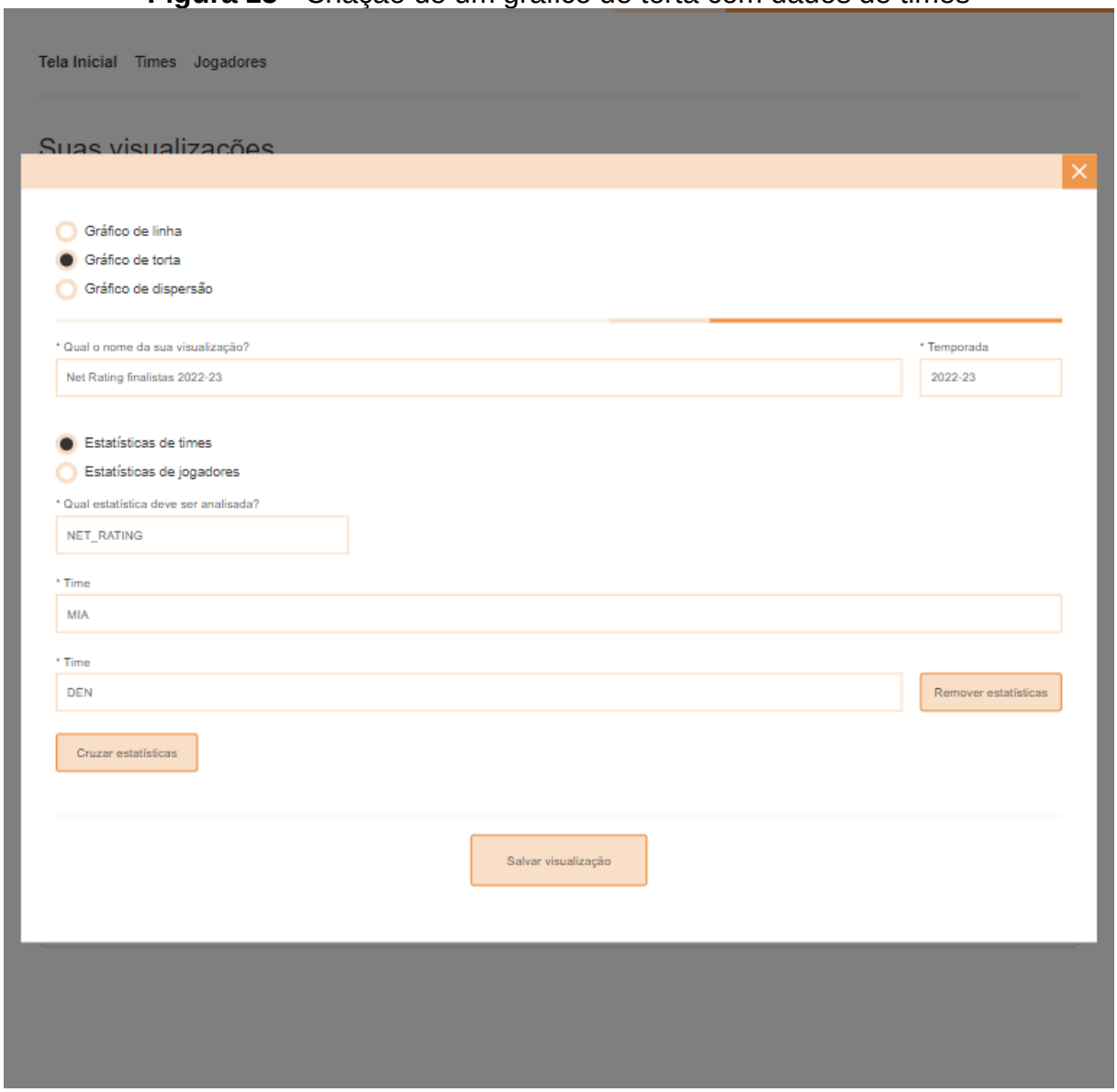


Fonte: os autores

Ao clicar em Nova visualização, o sistema disponibiliza a opção de criar diferentes tipos de gráficos, tais como dispersão, linha e torta, com base em dados de jogadores ou times. Enquanto os gráficos de dispersão e torta são restritos a uma única temporada, os de linha apresentam informações de todas as temporadas disponíveis. Esse processo é exemplificado nas Figuras 15 e 16.

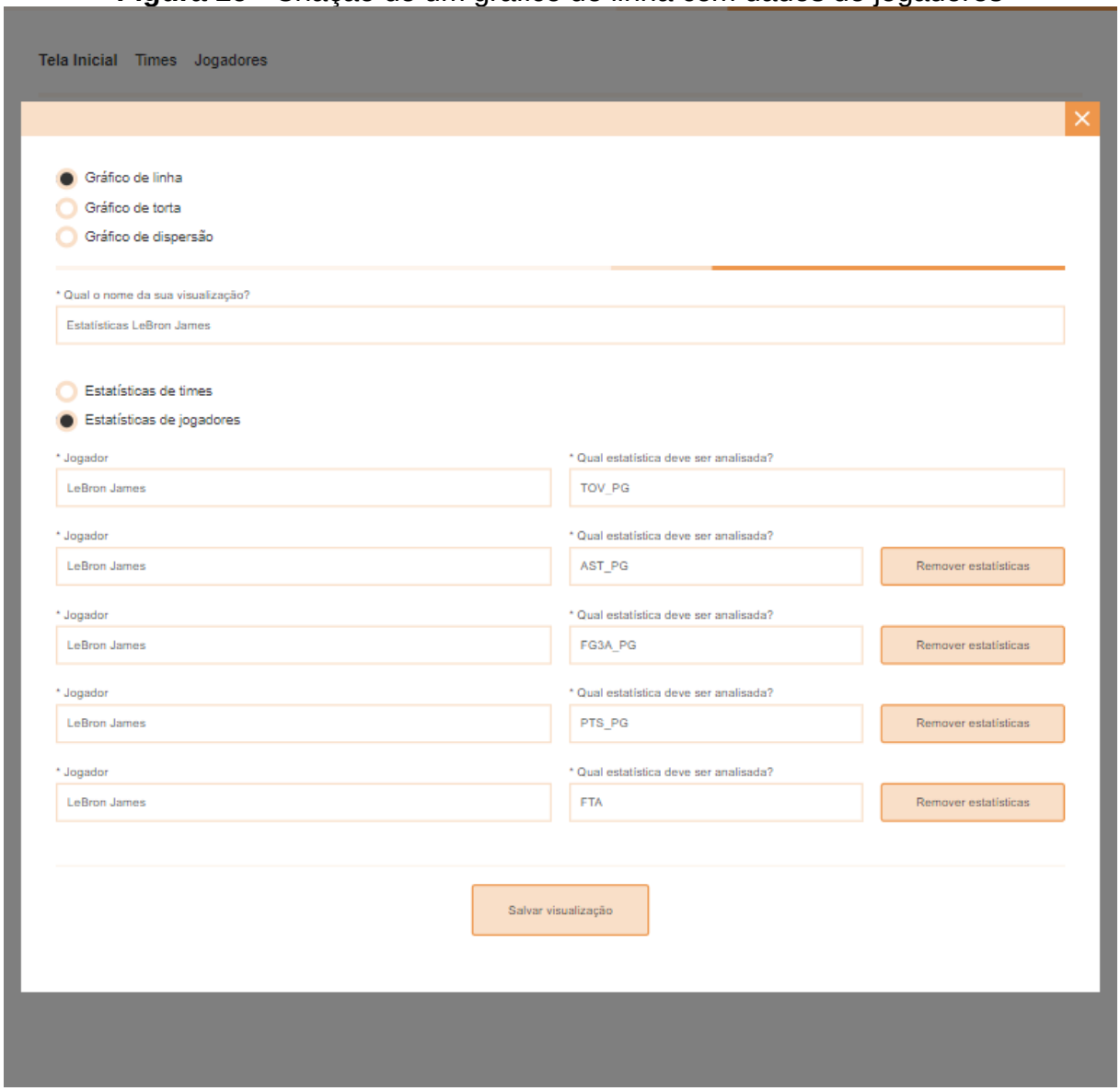
Diferentemente da tela inicial, na interface autenticada, os gráficos são interativos, permitindo ao usuário modificar ou deletar suas visualizações. Para fazer alterações, basta clicar no cabeçalho do gráfico desejado. Dessa forma, uma interface semelhante à utilizada na criação será exibida, possibilitando as devidas alterações.

Figura 15 - Criação de um gráfico de torta com dados de times



Fonte: os autores

Figura 16 - Criação de um gráfico de linha com dados de jogadores



Fonte: os autores

5 Considerações finais

O objetivo deste trabalho foi desenvolver um software destinado à comunidade especializada em basquete, oferecendo uma solução prática e eficiente para criar visualizações personalizadas das estatísticas das temporadas. Com a implantação do sistema, a proposta foi atingida.

Durante o desenvolvimento, foram empregados vários conceitos abordados no curso, incluindo técnicas de gerenciamento de bancos de dados, desenvolvimento de programas e princípios da Engenharia de Software.

Para o futuro do projeto, está prevista uma nova implementação do sistema, com melhorias nas interfaces e funcionalidades. O aumento na variedade e quantidade de parâmetros que constituem as visualizações é prioridade.

Referências

DONAVAN A. A. A.; KERNIGHAN W. B. A Linguagem de Programação Go. Santos: Novatec Editora, 2017.

FLANAGAN, D. JavaScript. O Guia Definitivo. Porto Alegre: Bookman Editora, 2013.
KRIGER, D. O que é Python, para que serve e por que aprender?. [S. l.], 2022. Disponível em: <https://t.ly/P7EHD>. Acesso em: 17 jul. 2023.

LEWIS, M. *Moneyball: The Art of Winning an Unfair Game.* Nova Iorque, W. W. Norton & Company, 2003.

LUCIDCHART. O que é *wireframe*? O que você quer fazer com *wireframes*?. [S. l.], [S. D]. Disponível em: <https://t.ly/TrW6E>. Acesso em: 14 jul. 2023.

MORENO, J. *Análisis de las estructuras del juego deportivo.* Barcelona, INDE Publicaciones, 1994.

MULESOFT. What is an API? [S. l.], [S. D.]. Disponível em: <https://t.ly/Q0sR9>. Acesso em: 23 jul. 2023.

MITCHELL, R. *Web Scraping with Python: Collecting More Data from the Modern Web*. Califórnia: O'Reilly Media, 2018.

STEINBERG, L. *CHANGING THE GAME: The Rise of Sports Analytics*. [S. l.], 2015. Disponível em: <https://t.ly/qsZN6>. Acesso em: 12 jul. 2023.

SYOSI, R. O que é *UI*? [*User interface*]. [S. l.], 2023. Disponível em: <https://t.ly/8zdVZ>. Acesso em: 13 jul. 2023.

TEIXEIRA, F. O valor do rabiscoframe. [S. l.], 2010. Disponível em: <https://t.ly/Vv7MP>. Acesso em: 14 jul. 2023.

TERA. Prototipagem de alta fidelidade: o que é, quando, por que e como usar? Tera Somos, [S. D.]. Disponível em: <https://t.ly/lduWk>. Acesso em: 15 jul. 2023.