UNIVERSIDADE
LUSÓFONA

Football BeTTing Strategies

Trabalho Final de curso

Relatório Final

Nome do Aluno: Rodrigo Miguel Marques Taciano

Orientador: Duarte Neves

Trabalho Final de Curso | LEI | 28/06/2024

www.ulusofona.pt

Direitos de cópia

Resumo

O Trabalho Final de Curso Football BeTTing Strategies tem como objetivo desenvolver uma aplicação móvel de uma aplicação *web* já existente (*Football BeTTing Strategies* (FBTTS) é uma aplicação *web* criada e gerida pelo professor Duarte Neves), cujo objetivo é tornar as apostas de futebol lucrativas através de análise estatística de várias equipas e/ou competições.

Na sua essência esta aplicação é uma ferramenta que combina a análise desportiva, a criação de estratégias e métodos de apostas, com o objetivo de não só aumentar o lucro do apostar, mas também facilitar o trabalho de análise ao utilizador.

O objetivo deste projeto é desenvolver uma aplicação móvel que estende a imagem da FBTTS. A aplicação permite aos utilizadores a consulta das suas estratégias e métodos de apostas, podendo observar várias estatísticas das mesmas (como o número de apostas, o lucro obtido, as ligas presentes na análise estatística da estratégia). A criação e edição de estratégias não são funcionalidades da aplicação móvel, pois consideramos que não é prático para utilizador criar ou editar uma estratégia num dispositivo móvel.

A aplicação FBTTS foi alvo de testes com utilizadores, que consideraram a aplicação *user-friendly* e responderam aos questionários propostos.

Palavras-chave: Football BeTTing Strategies (FBTTS), métodos de apostas e apostas desportivas.

Abstract

The objective of the Final Course Work is to develop a mobile application for an existing web application (Football BeTTing Strategies (FBTTS) is a web application created and managed by Professor Duarte Neves), which the main goal is to make football betting profitable by analysing the statistics of various teams and/or competitions.

In essence, this application is a tool that combines sports analysis, the creation of betting strategies and methods, with the aim of not only increasing the profit from betting, but also making the job of analysing easier for the user.

The aim of this project is to develop a mobile app that extends the FBTTS website. The app allows users to consult their strategies and betting methods and to see various statistics about them (such as the number of bets made, the profit made, the leagues present in the statistical analysis of the strategy, etc). Creating and editing strategies are not features of the mobile app, as we believe it is not practical for users to create or edit a strategy on a mobile device.

The FBTTS app was tested with users, who found it user-friendly and answered the proposed surveys.

Keywords: Football Betting Strategies (FBTTS), betting methods and betting strategies.

Índice

Lista de Figuras

Lista de Tabelas

1 Identificação do Problema

1.1 Contexto

A aplicação *web* FBTTS é a principal representação da plataforma *"Football BeTTing Strategies"*. Sendo apenas uma aplicação *web,* a mesma apresenta um problema para uma parte importante dos utilizadores, pois não se encontra otimizada para dispositivos móveis. O *design* da aplicação *web* e a forma como a mesma foi construída, apesar de responsiva, não é ideal, nem prático para utilizadores que tentem aceder à plataforma a partir de dispositivos móveis, o que é prejudicial, não só para a FBTTS, mas também para os seus utilizadores, pois cada vez mais se depende dos dispositivos móveis.

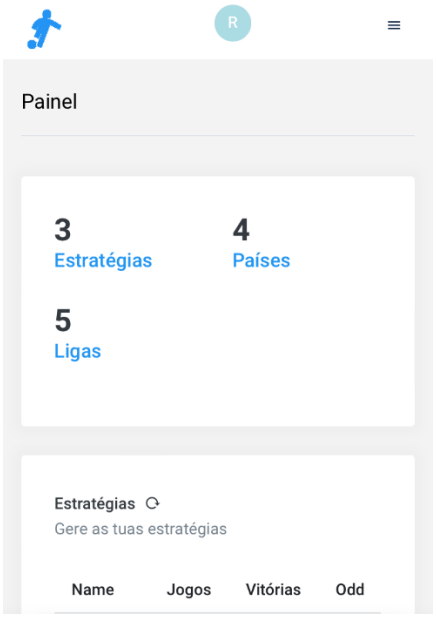


Figura 1 - Atual Painel do *Website*

1.2 Objetivos

Tendo em consideração o problema anteriormente apresentado e a proposta de resolução do mesmo, a aplicação móvel a desenvolver tem o intuito de tornar a consulta de estratégias de apostas desportivas mais acessível e *user-friendly* ao utilizador, tornando a aplicação mais intuitiva.

Com a criação da aplicação móvel, os utilizadores teriam mais um canal de acesso à plataforma e a possibilidade de gerir o seu dia de apostas a partir do seu smartphone.

Embora a aplicação *web* forneça, uma interface responsiva (como já foi mencionado anteriormente), que permite o acesso dos utilizadores a partir de dispositivos móveis, existe conteúdo, funcionalidades e botões (ações), que não se encontram disponíveis nesse mesmo modo responsivo. A aplicação móvel a ser desenvolvida tem como objetivo resolver essas limitações.

A aplicação móvel permitirá aos utilizadores consultar as suas estratégias (e estatísticas das mesmas) de uma forma mais prática e móvel, pois não estarão limitados a fazê-lo no seu

computador ou portátil. É de realçar que a disponibilização da aplicação móvel permitirá à própria FBTTS a criação de uma campanha de *marketing* para se publicitar a si mesma.

2 Viabilidade e Pertinência

2.1 Viabilidade

Relativamente à viabilidade do projeto, vale realçar que não se trata apenas de um trabalho académico. O propósito fundamental deste projeto é expandir a plataforma/marca FBTTS, fornecendo mais uma forma de aceder à mesma.

Atualmente, existe uma grande quantidade de dados e estatísticas que abrangem vários jogos e competições de futebol. Esses dados serão organizados em categorias relevantes, de acordo com as diretrizes da aplicação *web* já desenvolvida. Os modelos estatísticos usados para analisar os dados e produzir resultados de apostas são cuidadosamente selecionados e adaptados às necessidades do contexto das apostas desportivas de futebol.

2.2 Pertinência

A pertinência deste Trabalho Final de Curso está intrinsecamente ligada ao seu objetivo de abordar uma necessidade genuína identificada no âmbito da plataforma FBTTS. Abordada esta necessidade com o professor Duarte Neves, já havia a ideia do desenvolvimento de uma aplicação móvel da FBTTS. Ao longo do tempo foram realizadas várias reuniões com o professor Duarte Neves, onde se foi otimizando e melhorando o desenvolvimento da aplicação móvel.

A indústria das apostas desportivas é uma atividade em crescimento, que atrai cada vez mais novos utilizadores. No entanto, a complexidade das estatísticas e análises necessárias para tomar as decisões corretas muitas vezes representam um desafio para esses novos "apostadores", que não só procuram maximizar os seus lucros, mas também usufruir de uma boa experiência ao longo desse processo. (Fonte: *https://www.statista.com/outlook/amo/online-gambling/online-sports-betting/worldwide#revenue*)

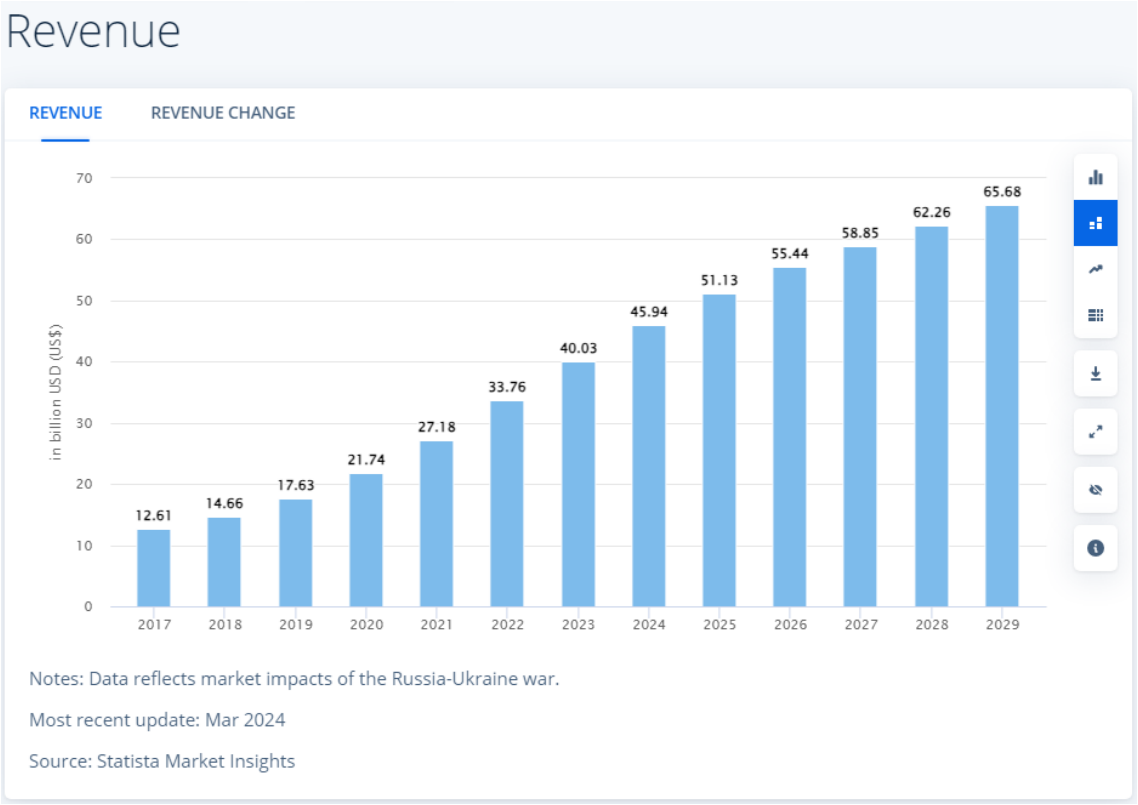


Figura 2 - Receitas de apostas desportivas (março 2024)

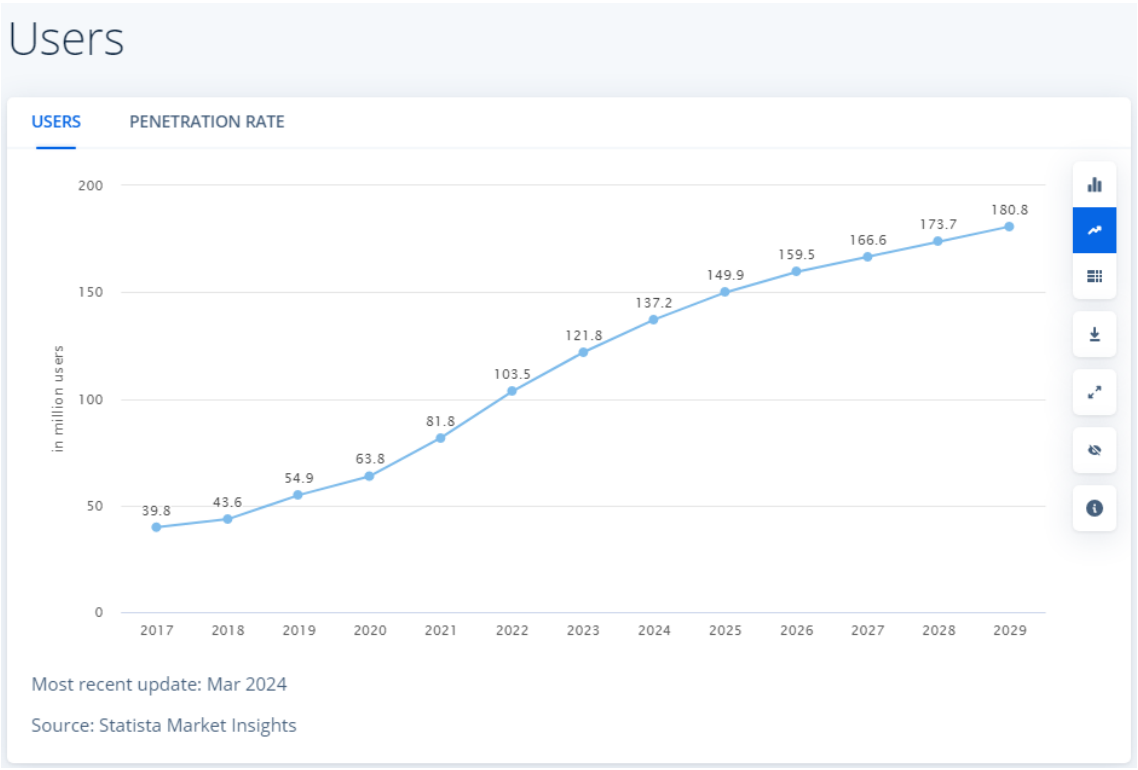


Figura 3 - Número de utilizadores registados em plataformas de apostas despostivas

É de realçar que a aplicação proposta simplifica o processo de análise estatística e torna-a acessível a um público amplo, mesmo para utilizadores com pouca experiência (mas algum

conhecimento) em apostas desportivas, demonstrando que a aplicação torna a análise mais acessível.

Em suma, a aplicação proposta não é apenas um projeto académico, mas uma ferramenta com potencial continuidade que visa contribuir positivamente para a resolução de desafios reais no mundo das apostas desportivas, proporcionando aos utilizadores uma forma mais simples, menos demorada e mais acessível de realizarem a análise pré-aposta.

3 Benchmarking

Para perceber melhor a posição da solução apresentada no cenário atual do mundo de estratégias de apostas de futebol é preciso analisar outras alternativas que já existem no mercado. Foi realizada uma pesquisa para identificar como outras alternativas funcionam e constatar a necessidade do desenvolvimento deste Trabalho Final de Curso.

Foi identificada uma alternativa que fornece estratégias de apostas desportivas de futebol em aplicações web e/ou móveis, Footy Amigo.

3.1 Footy Amigo

O Footy Amigo fornece estratégias de apostas desportivas. Os utilizadores da aplicação conseguem criar as suas próprias estratégias, porém o número de estratégias é limitado caso sejam utilizadores grátis. Conseguem também consultar estratégias existentes ou comprar estratégias feitas por outros utilizadores, tal como se encontra representado na Figura 4 - Footy Amigo

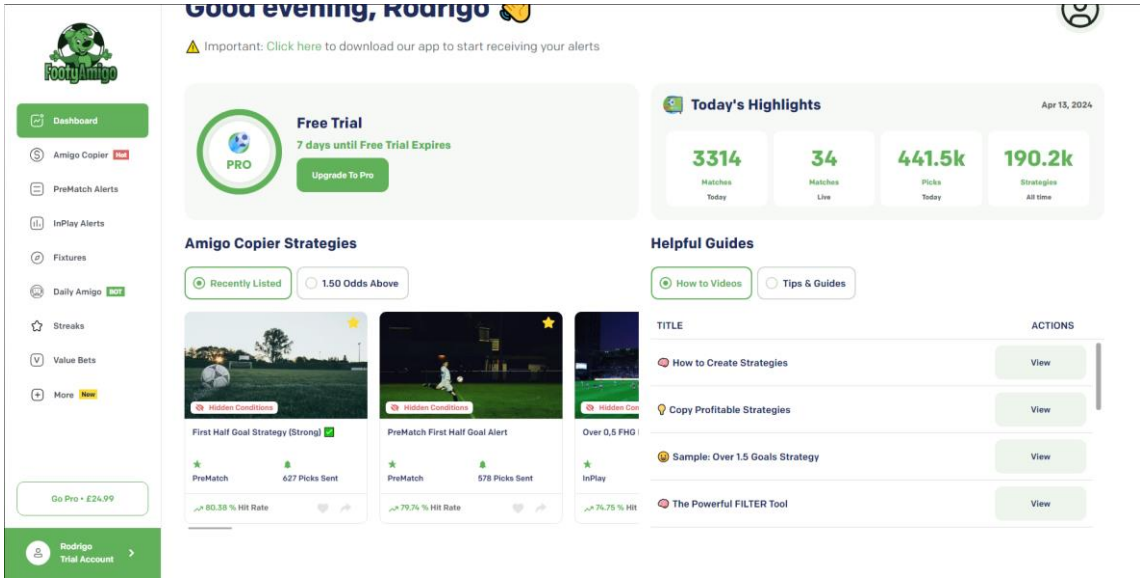


Figura 4 - Footy Amigo

Esta aplicação só se encontra completamente dísponivel para utilizadores “Pro”. A consulta de estratégias criadas por outros utilizadores é também pago.

Tabela 1 - Comparação FBTTS vs Footy Amigo

Funcionalidades	FBTTS	Footy Amigo
Criação/personalização de estratégias		
Acesso a estratégias de outros utilizadores		
Acesso gratuito a todas as funcionalidades da plataforma		
Visualização de jogos de competições e ligas		
A Aplicação encontra-se disponível em português		
Combinar condições criadas pelo próprio utilizador (não está limitado aos *inputs* existentes da plataforma)		

Como se pode observar na Tabela 1 - Comparação FBTTS vs Footy Amigo, a aplicação Footy Amigo possui uma grande quantidade de recursos "bloqueados", para um utilizador normal (que não pague pela versão premium do website/aplicação web). Comparativamente à FBTTS, o Footy Amigo permite ao utilizador personalizar as suas estratégias, contudo tem algumas limitações, pois o utilizador não tem a opção de inserir os seus próprios inputs e definir manualmente a sua estratégia.

A aplicação móvel da FBTTS está focada na consulta de estratégias, a definição das mesmas deve ser feita na aplicação web, para uma experiência melhor e mais intuitiva.

4 Engenharia

4.1 Levantamento e análise dos Requisitos

Para o desenvolvimento da aplicação móvel, tendo como objetivo uma melhor organização, de forma a concretizar os objetivos da solução proposta, foi efetuado o levantamento e análise dos requisitos necessários ao Trabalho Final de Curso.

Os requisitos estão divididos em funcionais e não funcionais, a sua importância é indicada usando a escala MoSCoW, que possui os seguintes classificadores: *must have*, *should have*, *could have*, *nice to have*.

Nesta escala um requisito classificado como "*must have*" é obrigatório, enquanto um requisito classificado como "*nice to have*", significa que a aplicação deve ter. Um requisito "*could have*" é um requisito que a aplicação pode ter, e um requisito classificado como "*nice to have*" representa um requisito que seria bom de ter.

A taxa de esforço esperado para o desenvolvimento de cada requisito foi medida sob a escala de Fibonacci, que associa os números da série ao tempo que se espera para o desenvolvimento do requisito. A escala de pontos será medida de acordo com a Tabela 2 - Escala Fibonacci.

Tabela 2 - Escala Fibonacci

Pontos	Dias (úteis) de esforço
1	Menos de 1 dia
2	Entre 2 a 3 dias
3	Entre 3 a 5 dias
5	Entre 5 a 6 dias
8	Entre 6 e 7 dias
13	Entre 7 a 10 dias
21	Entre 11 a 13 dias
34	Entre 14 a 20 dias
55	Mais de 20 dias

A Tabela 3 - Requisitos FBTTS apresenta os requisitos definidos para a aplicação FBTTS.

Tabela 3 - Requisitos FBTTS

ID	Nome	Descrição	Importância	Esforço	Tipo
01	Autenticar Utilizador	Para realizar o login na aplicação, é necessário introduzir um endereço de e-mail válido e a palavra-passe correspondente associada à conta do utilizador. Esta autenticação é crucial para garantir o acesso seguro e individualizado aos recursos e funcionalidades oferecidos pela aplicação móvel.	Must Have	2	Funcional
02	Criar estratégia	O utilizador da FBTTS, após se autenticar é levado à página principal (*Dashboard*/Painel) onde pode criar uma estratégia (através de um botão).	Nice to Have	3	Funcional
03	Atualizar estratégia	O utilizador da FBTTS a partir da *Dashboard* (Painel) pode atualizar as suas estratégias.	Should Have	2	Funcional
04	Consultar estratégias	Na página principal (Dashboard) é apresentado o nome das estratégias do utilizador.	Must Have	1	Funcional
05	Consultar lucro obtido	Na *Dashboard* as estratégias possuem uma seta colorida que representa se uma determinada estratégia lucrou ou não para o utilizador.	Nice to Have	2	Funcional

06	Consultar estratégia detalhadamente	O utilizador da FBTTS, após ter clicado numa determinada estratégia (na *Dashboard*) é levado para uma página com o nome da estratégia bem como todas as características de uma determinada estratégia, isto é, as ligas, o número de jogos que a estratégia se aplica, o número de vitórias nessa estratégia, a odd média da estratégia e o lucro obtido na estratégia.	Must Have	3	Funcional
07	Incluir botões na página de estratégia	Na página da estratégia existem 3 botões principais, um que permite a edição da mesma, outro que mostra os jogos que se aplicam a essa estratégia (jogos no próprio dia) e um último botão que permite ao utilizador apagar a estratégia.	Nice to Have	1	Funcional
08	Editar estratégia	Na página de editar estratégia deve-se poder mudar qualquer atributo da estratégia.	Could Have	3	Funcional
09	Editar estratégia manualmente	Na página de editar estratégia o utilizador pode alterar a estratégia manualmente, escrevendo os comandos da mesma.	Could Have	5	Funcional
10	Consultar Ligas	Na página Ligas são mostradas as ligas/campeonatos incluídos na plataforma FBTTS.	Should Have	2	Funcional
11	Consultar Ligas em detalhe	O utilizador da FBTTS, após clicar numa liga em específico na página de Ligas é levado a uma	Nice to Have	5	Funcional

		página dessa liga em detalhe, onde pode observar a tabela classificativa, os jogos da presente jornada e informações relevantes para estratégias e apostas desportivas (i.e. mercado de golos, média de golos etc…)			
12	Enviar Feedback (através da página Suporte)	O utilizador pode enviar feedback ou relatar algum problema na aplicação móvel. A página de suporte possui 2 campos a preencher, um com o assunto e outro com a descrição do mesmo.	Should Have	2	Funcional
13	Consultar página de ajuda de estratégias	O utilizador da FBTTS pode consultar esta página caso tenha dúvidas acerca de como criar/editar estratégias.	Nice to Have	3	Funcional
14	Consultar Jogos	O utilizador da FBTTS pode consultar os jogos (incluídos nas ligas da plataforma) para uma determinada data.	Must Have	2	Funcional
15	Efetuar Compras	Na página de Compras o utilizador pode comprar mais cliques de forma a continuar a utilizar o site, ou desbloquear novas funcionalidades, caso seja a sua primeira compra na plataforma.	Could Have	5	Funcional
16	Consultar Menu	A navegação da aplicação móvel FBTTS é feita através do menu.	Must Have	1	Funcional
17	Visualizar Splash screen	Quando a aplicação móvel abre é apresentada uma splashscreen com uma animação, enquanto a aplicação abre.	Nice to have	1	Funcional

18	Desempenho	A aplicação deve carregar páginas em menos de 3 segundos em conexões móveis padrão.	Must Have	5	Não funcional
19	Multiplataforma	A aplicação móvel FBTTS encontra-se disponível para ambos sistemas operativos Android e iOS.	Must Have	2	Não funcional
20	Usabilidade	A interface deve ser simples e intuitiva de forma a permitir uma navegação fácil.	Must Have	3	Não funcional
21	Escalabilidade	Deve suportar até 10.000 utilizadores simultâneos sem degradação significativa do desempenho.	Should Have	5	Não funcional

4.2 Diagramas de Casos de Uso

O perfil utilizador FBTTS possui várias ações de modo a responder às necessidades do utilizador.

Após efetuar a autenticação o utilizador consegue realizar várias ações:

- Consultar o Painel (Dashboard);
- Consultar Estratégia;
- Consultar Conta;
- Submeter Feedback;
- Consultar Ajuda;
- Consultar Jogos;
- Consultar um determinado jogo em detalhe;
- Consultar Ligas;
- Consultar uma determinada liga em detalhe;

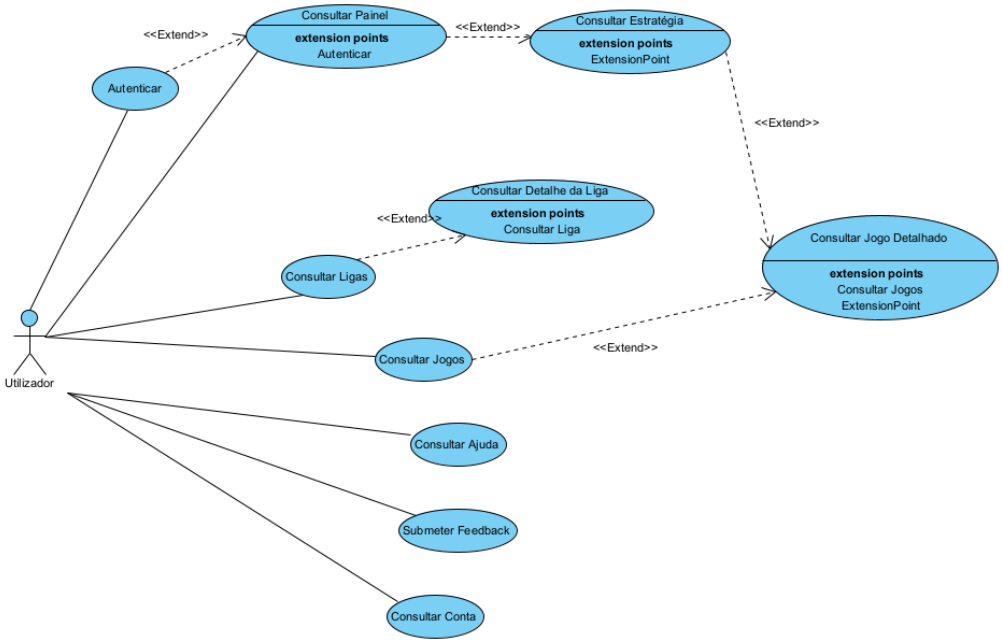


Figura 5 - Diagrama Caso de Uso

4.3 Diagramas de Atividades

Com o objetivo de oferecer uma melhor visualização das diferentes tarefas e fluxos de trabalho que compõem este TFC, foi desenvolvido um diagrama de atividades que pode ser visto nas próximas figuras. Este exibe as diferentes tarefas da aplicação, fluxos de trabalho e interações.

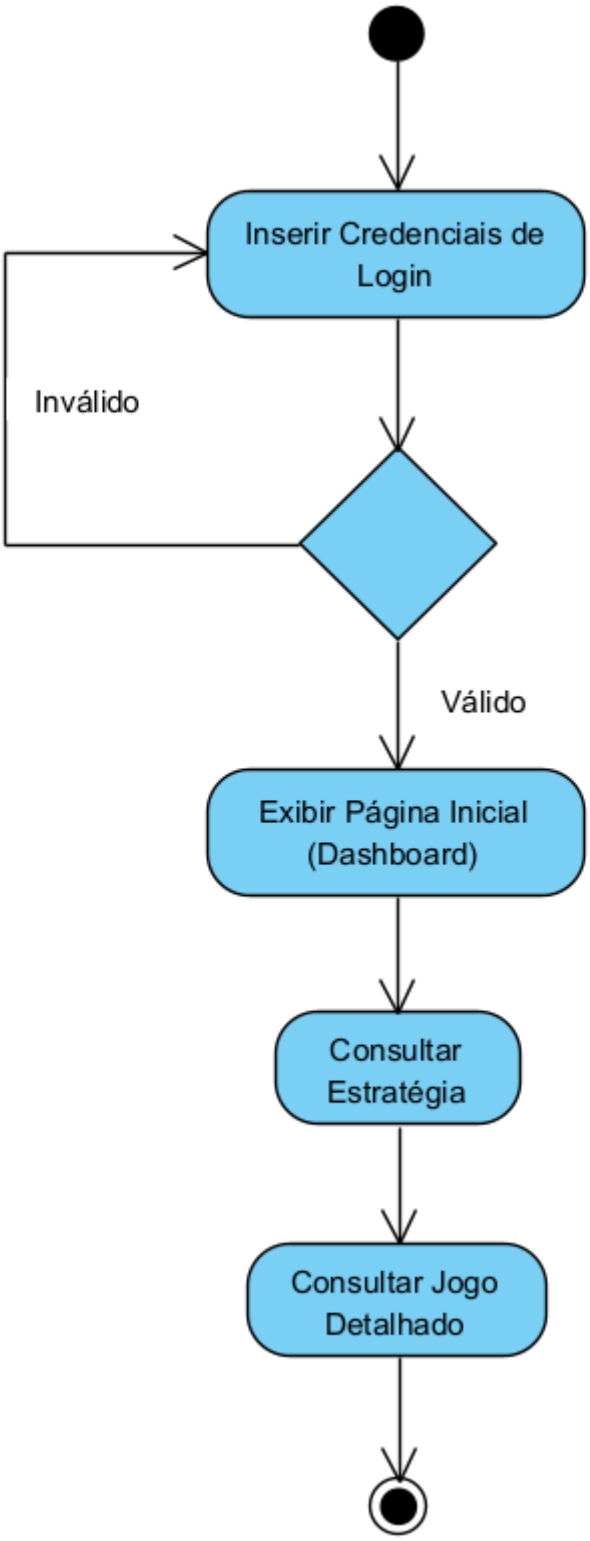


Figura 6 - Diagrama de Atividades Consultar Estratégia

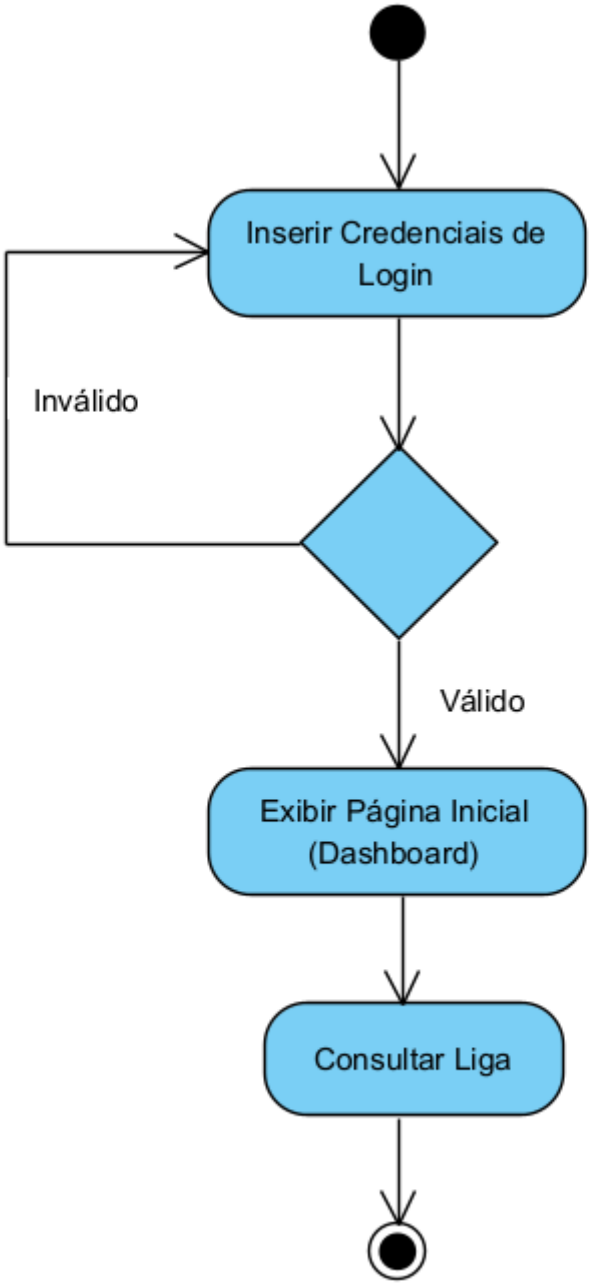


Figura 7 - Diagrama de Atividades Consultar Liga

4.4 Modelos relevantes

Com o objetivo de esclarecer a estrutura e interações fundamentais do sistema, foi criado um Diagrama de Classes. Este diagrama descreve as entidades essenciais, as suas propriedades, métodos e as relações (caso existam). O diagrama de classes destaca a organização lógica e a arquitetura do software. No anexo B está presente a imagem desse mesmo diagrama.

4.5 Estrutura (em árvore, se aplicável)

Foi criado um mapa aplicacional para compreender a estrutura da aplicação e os ecrãs pelos quais os utilizadores podem navegar enquanto utilizam a aplicação. O mapa aplicacional possui três níveis de navegação. No primeiro nível pode-se observar o Painel ou *Dashboard* que é a página principal do utilizador. No segundo nível é possível aceder aos ecrãs Perfil, Ajuda, Suporte, Estratégia em Detalhe, Jogos e Ligas. No terceiro nível pode-se aceder ao ecrã Liga em Detalhe e Jogo em detalhe.

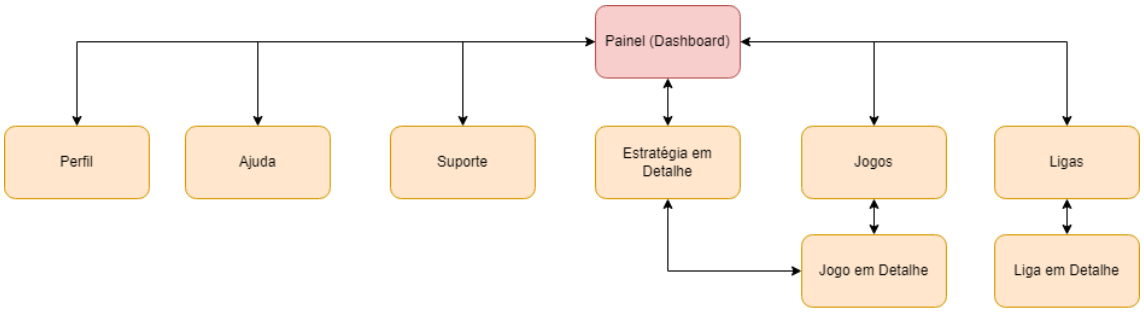


Figura 8 - Mapa Aplicacional

Na página Ajuda o utilizador pode encontrar dicas e "explicações" de como funciona a plataforma FBTTS.

Na página Suporte é possível enviar feedback (através de um formulário), relativamente à aplicação móvel.

Na página Jogos o utilizador pode ver os jogos (das ligas existentes na plataforma), para uma determinada data (quando o utilizador abre esta página, os jogos apresentados são os do próprio dia).

Na página Ligas é possível observar as ligas presentes na plataforma e aceder a informações detalhadas sobre as mesmas.

No Perfil o utilizador pode consultar as suas informações (nome, email, etc).

Na página Ligas em Detalhe é possível ver a tabela de classificação dessa liga, bem como em que jornada a competição se encontra, jornadas futuras e estatísticas de apostas alusivas a essa liga (como estatísticas de golos, como média de golos ao intervalo e por jogo).

Na página Jogo em Detalhe o utilizador consegue observar diferentes estatísticas importantes de ambas as equipas, como os últimos confrontos diretos, os últimos jogos (na sua liga) etc…

4.6 Mockup

O mockup da aplicação móvel integra as cores principais da aplicação web, mantendo as ideias de design do mesmo.

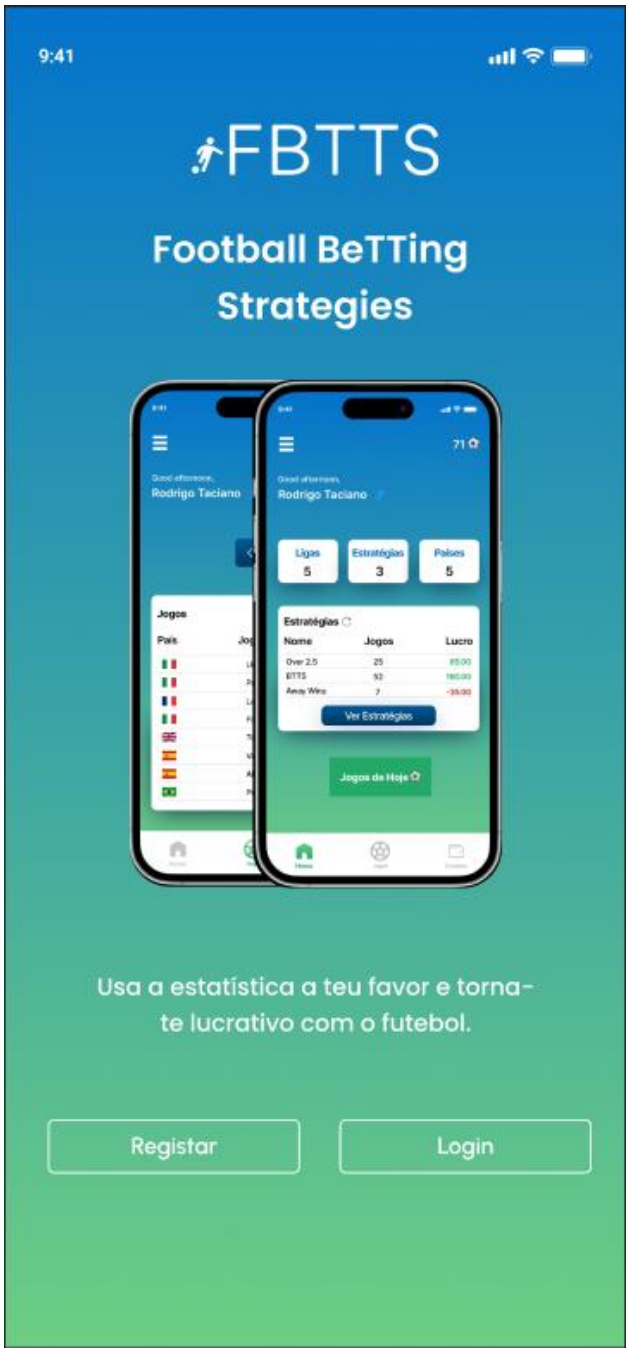


Figura 9 - *Splash screen* da FBTTS

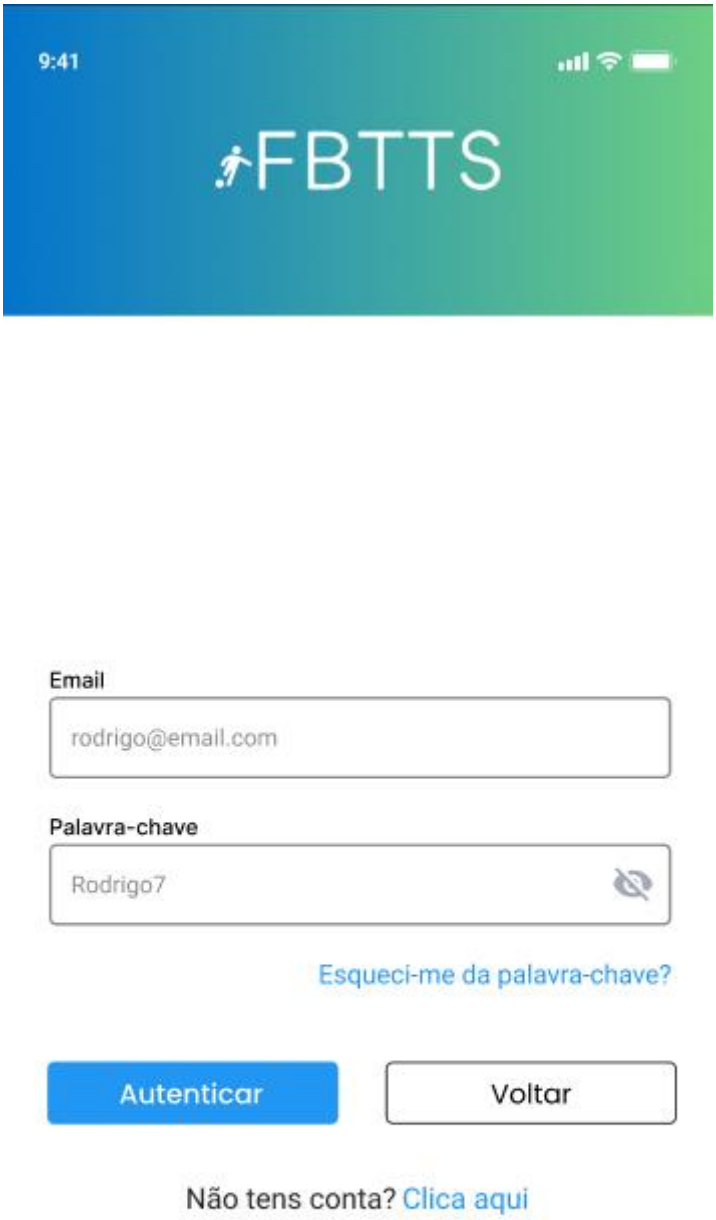


Figura 10 - Ecrã de Login

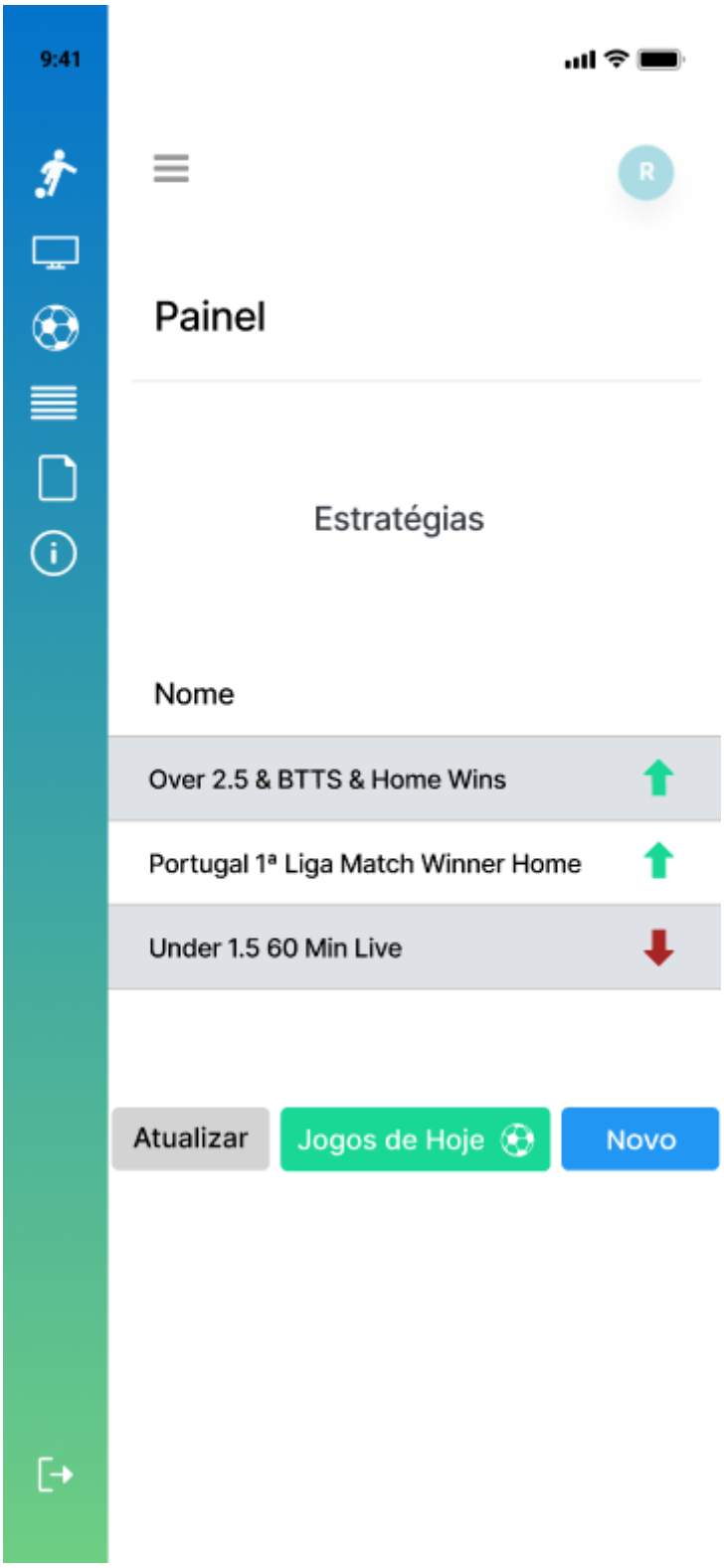


Figura 11 - Página principal (Dashboard)

5 Solução Proposta

5.1 Introdução

A aplicação móvel está implementada para que seja utilizada em ambos os sistemas Operativos Android e iOS. O desenvolvimento da aplicação móvel será realizado através da biblioteca React Native, que permitire a criação de aplicações móveis para ambos os Sistemas Operativos. Para o desenvolvimento *back-end*, a escolha recaiu sobre o Spring Boot, uma *framework* Java que oferece uma base sólida para a construção de serviços RESTful e APIs. Como base de dados, MongoDB, uma base de dados NoSQL altamente flexível e escalável. O vídeo de demonstração da aplicação pode ser visto no seguinte link: https://youtu.be/cnS7ytb2EMws.

Link do Github (Front-end).

Link do Github (Back-end).

5.2 Arquitetura

A arquitetura Java SpringBoot adotada para este TFC consiste numa estrutura baseada em microsserviços. Essa abordagem modularizada permite a divisão de tarefas nos diferentes componentes, permitindo uma maior flexibilidade e escalabilidade no desenvolvimento e manutenção do sistema.

A arquitetura segue um padrão de camadas, a camada "*Controller*", que gere os pedidos HTTP, a camada de serviço responsável pela lógica de negócios e acesso aos dados, "*Service Layer*", e a camada "*Model*", que se conecta à base de dados (MongoDB). Essa estrutura favorece a separação de responsabilidades e a reutilização de código.

No contexto do Spring Boot, a arquitetura segue um modelo altamente orientado por convenções, que permite uma configuração mais simples através de anotações e diminuindo a necessidade de definições detalhadas. A modularidade promovida pela adoção de microsserviços permite separar funcionalidades específicas, o que facilita a manutenção independente de cada componente.

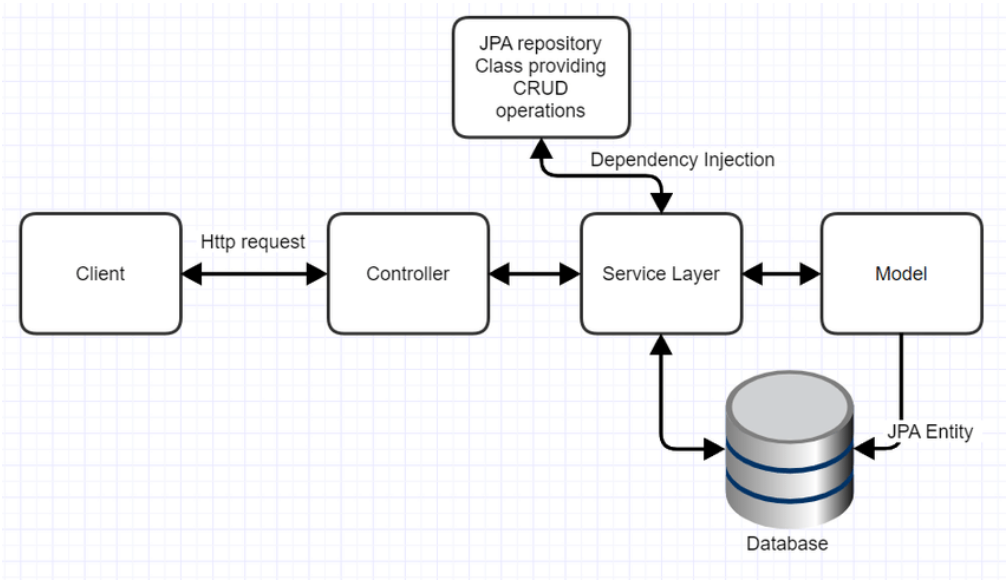


Figura 12 - Arquitetura Java Spring Boot (fonte: ResearchGate)

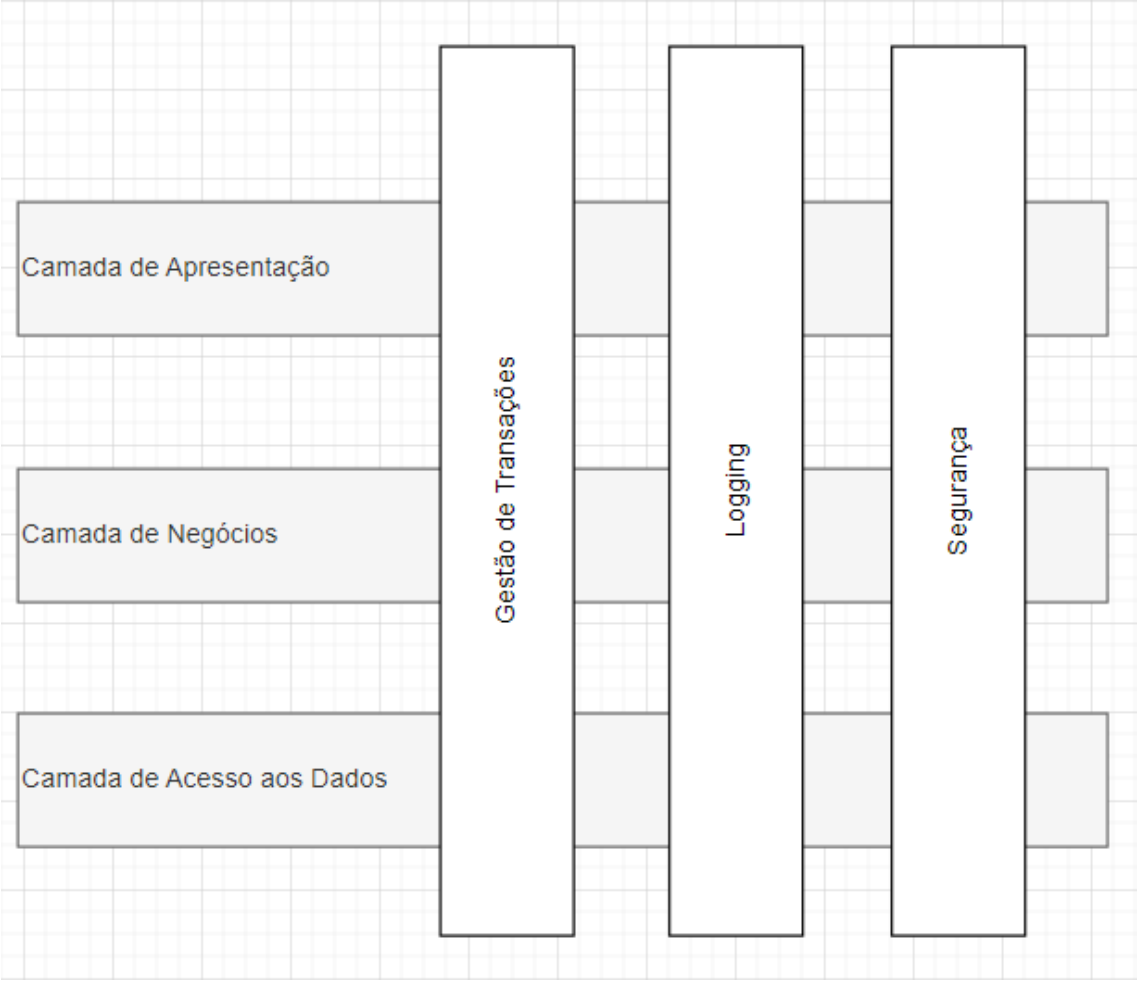


Figura 13 - Camadas do Spring Boot

Camada de Apresentação: A camada de apresentação é responsável pela interação direta com os utilizadores. Aqui, são tratados os pedidos HTTP, gestão de sessões, e apresentação de resultados. O Spring MVC, por exemplo, pode ser utilizado para criar *Controllers* que lidam com *WebServices* e pedidos à API.

Camada de Negócios: Nesta camada, as operações de negócios são implementadas, garantindo que a aplicação executa as suas funcionalidades conforme as necessidades do domínio. O SpringBoot fornece suporte para a criação de serviços e componentes de negócios eficientes.

Camada de Acesso de Dados: Responsável pela comunicação com a base de dados, a camada de acesso de dados utiliza tecnologias como Spring Data JPA para fornecer uma abstração simplificada do acesso aos dados. As operações de leitura e gravação na base de dados são encapsuladas nesta camada, permitindo uma comunicação eficiente e segura.

***Logging*:** A camada de *Logging* aborda a necessidade de procurar e registar atividades no sistema. Utilizando bibliotecas de *logging* do Spring, é possível criar *logs* detalhados que ajudam na gestão e resolução de problemas.

Segurança: A camada de Segurança do Spring Boot disponibiliza recursos robustos para autenticação e autorização (como JWT). Configurações como Spring Security podem ser implementadas para garantir a integridade e a confidencialidade dos dados.

Gestão de Transações: Na camada de Gestão de Transações, o Spring Boot facilita o controlo transacional, garantindo que operações críticas são atomicamente executadas na base de dados. O uso de anotações, como **@Transactional**, simplifica a gestão de transações em ambientes complexos.

5.3 Tecnologias e Ferramentas Utilizadas

5.3.1 Maven

A escolha do Maven como ferramenta de gestão de projeto é fundamentada em diversas vantagens que esta tecnologia oferece, contribuindo significativamente para o desenvolvimento eficiente e organizado de software. Algumas das razões para optar pelo Maven são:

- Gestão de Dependências Simplificada: Maven simplifica a gestão de dependências, facilita a inclusão e atualização de bibliotecas externas ao projeto. O que reduz a complexidade associada à gestão manual de bibliotecas e garante que as versões corretas sejam as utilizadas.
- Repositório Central: Maven utiliza o Repositório Central como um repositório centralizado para bibliotecas e plugins, o que facilita a partilha de conteúdo e promove a reutilização de código.
- Plugins Abundantes: Maven possui uma ampla variedade de plugins que automatizam tarefas comuns de desenvolvimento, como a integração de testes e criação de relatórios, o que aumenta a produtividade e a consistência no desenvolvimento.
- Integração com Ambientes de Desenvolvimento: Maven integra-se facilmente com os ambientes de desenvolvimento mais utilizados, como Eclipse e IntelliJ IDEA.

5.3.2 React Native

A escolha de React Native para o desenvolvimento da aplicação é baseada nas inúmeras vantagens que esta *framework* oferece, tais como:

- Desenvolvimento *Multiplataform*: React Native permite que o mesmo código seja utilizado para ambos os Sistemas Operativos, iOS e Android, o que simplifica o processo de criação de uma aplicação para os dois Sistemas Operativos.

- Eficiência e Agilidade: A agilidade é uma das principais vantagens do React Native. Os *developers* podem criar aplicações num curto espaço de tempo comparativamente com o desenvolvimento nativo, economizando tempo e recursos.
- *Performance*: React Native oferece uma *user experience* fluida e responsiva, exige mais do computador no seu desenvolvimento, no entanto permite um melhor desempenho final.
- Informação Disponível: React Native tem uma comunidade bastante grande, o que permite aos *developers* partilharem conhecimento de forma mais fácil e intuitiva, de várias formas como vídeos ou no StackOverflow por exemplo.

React Native é framework de JavaScript mantida pelo Facebook que possibilita o desenvolvimento de aplicações para ambos os Sistemas Operativos (iOS e Android). Utilizando JavaScript e React, o React Native permite que os *developers* criem aplicações móveis de alto desempenho.

5.3.3 Spring Boot

Spring Boot é uma *framework* Java amplamente reconhecida que oferece uma base sólida para a construção de serviços RESTful e APIs.

Uma das principais vantagens do Spring Boot é a sua facilidade de configuração. O Spring Boot simplifica o processo de configuração e inicialização do projeto, permitindo que os *developers* se concentrem no desenvolvimento de funcionalidades em vez de o gastar a desenvolver o ambiente, o que acelera o processo e reduz a complexidade do projeto.

Outra vantagem é a facilidade de comunicação com a base de dados: O Spring Boot oferece um suporte integrado para a criação de aplicações conectadas a bases de dados de uma forma eficaz.

5.3.4 MongoDB

Para base de dados, a escolha recaiu sobre o MongoDB, um sistema de base de dados NoSQL (“*NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.”)* que se destaca pela sua alta flexibilidade e escalabilidade. MongoDB é uma escolha estratégica para nosso projeto, pois a quantidade de informação a ser processada é gigante.

Podemos utilizar um exemplo da aplicação FBTTS:

São analisados cerca de 300 jogos para cada liga num determinado ano, além da consideração de todas as jornadas da competição numa época completa, o que dá um total de cerca de 34/38 jornadas, o que torna a escolha do MongoDB benéfica.

Dentro deste contexto de um único documento MongoDB (formatado em JSON), é possível consolidar um conjunto considerável de informações, abrangendo detalhes de jogos, estatísticas de desempenho das equipas, e até mesmo informações sobre outras equipas na competição a cada jornada. Essa abordagem elimina a necessidade de realizar múltiplos JOINs, uma vez que as informações relacionadas estão encapsuladas dentro do documento, simplificando assim as consultas e proporcionando maior eficiência no acesso aos dados.

Além disso, a capacidade do MongoDB de suportar transações é essencial para garantir a consistência dos dados, especialmente quando lidamos com análises detalhadas que envolvem diversas interações e atualizações simultâneas. Em resumo, o MongoDB oferece uma solução robusta e eficiente para a gestão de dados complexos em projetos como o FBTTS, destacando-se pela flexibilidade, escalabilidade e desempenho nas análises desportivas.

5.3.5 Postman

Para a realização de testes de APIs, foi utilizado o Postman, uma ferramenta de desenvolvimento amplamente reconhecida e utilizada pela sua flexibilidade e escalabilidade.

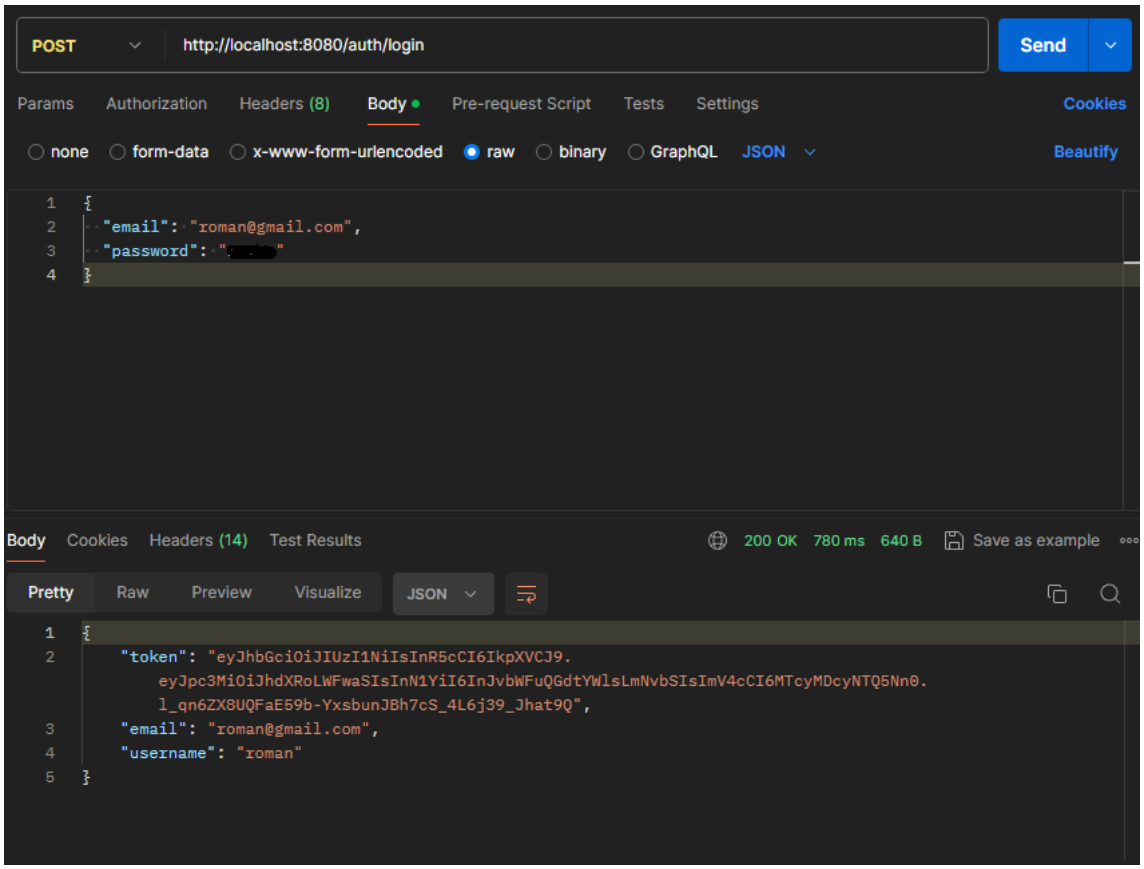


Figura 14 - Teste Autenticação no Postman

5.4 Implementação

No projeto realizado em React Native (Front-end), as seguintes pastas suportam a estrutura do projeto: assets, components, JSON, context, routes, screen e services.

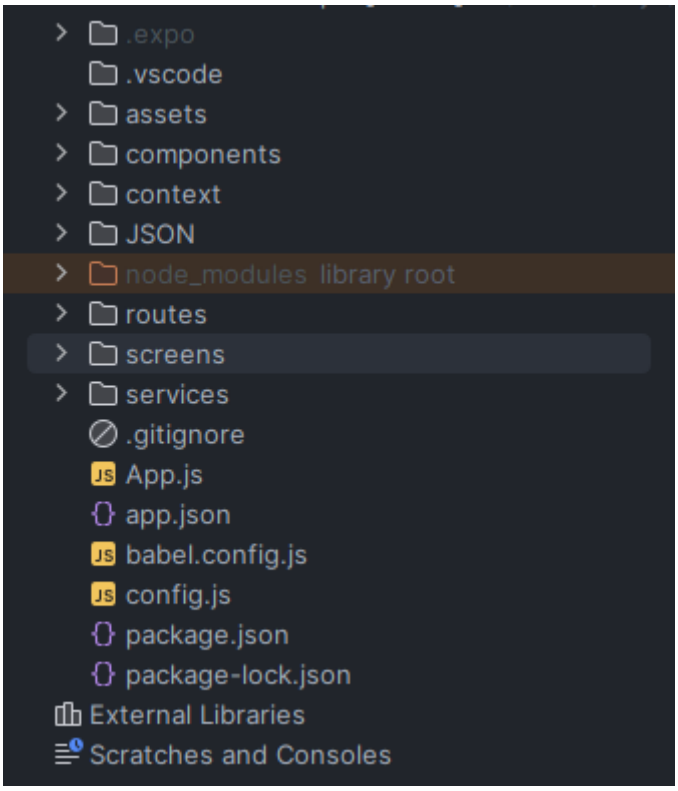


Figura 15 - Estrutura do Projeto (Front-End)

Esta estrutura permite o funcionamento da autenticação, navegação da aplicação, consulta de estratégias, ligas, jogos e informações da conta.

Quando o utilizador abre a aplicação é remetido para a *splashscreen* da mesma, clicando no botão “começar” navega para a página de autenticação.

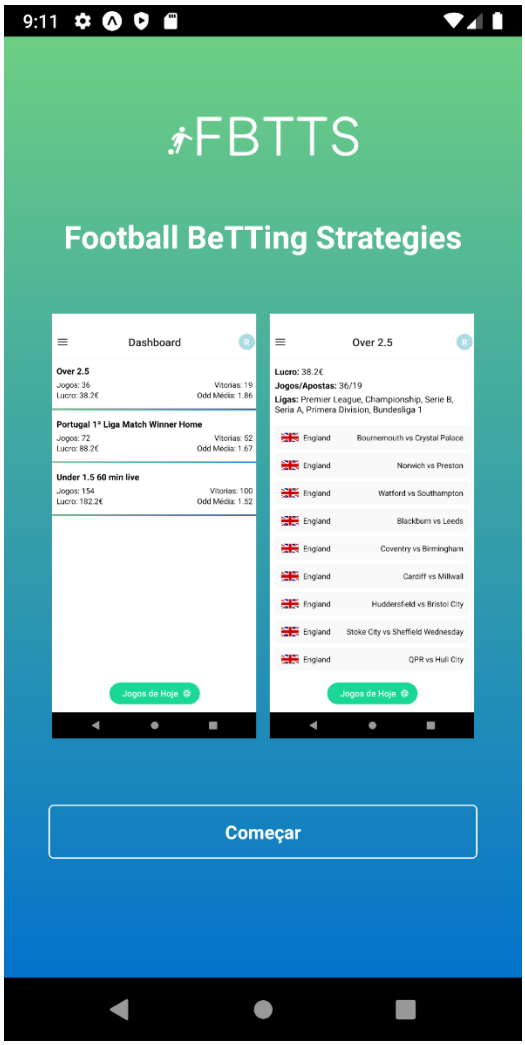


Figura 16 – *Splashscreen*

A aplicação possibilita um login inicial (como se pode observar na Figura 17 - Página de Login), onde a validação das credenciais é feita através de um pedido à API, dependendo da resposta é feito o redirecionamento para o painel (dashboard), caso exista algum erro, o acesso é negado.

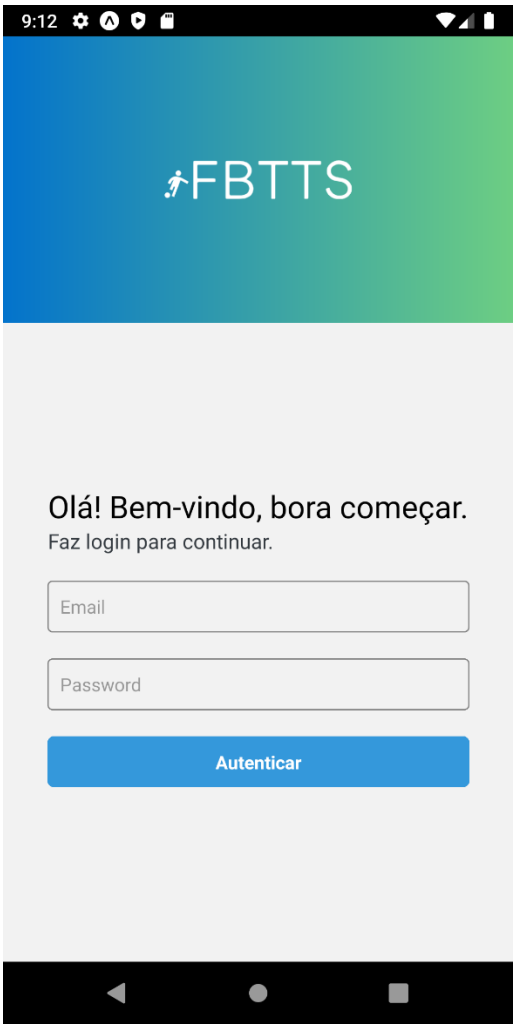


Figura 17 - Página de Login

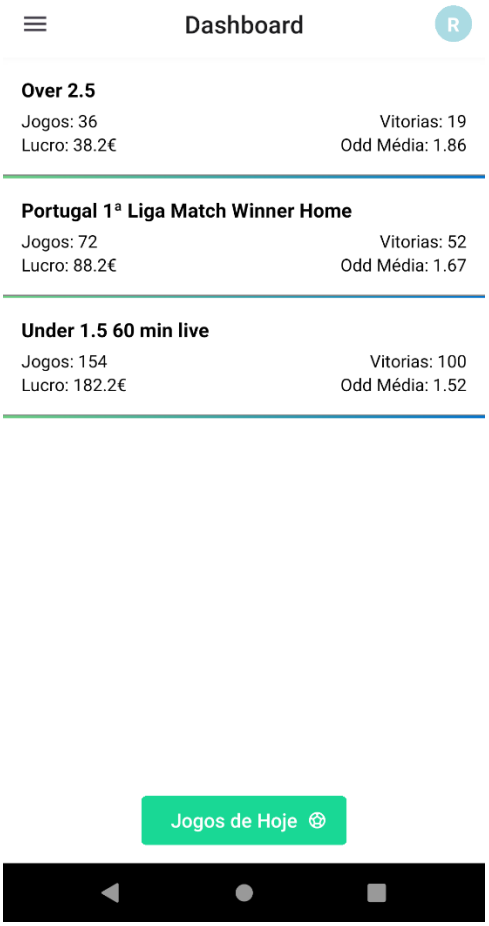


Figura 18 - Dashboard da Aplicação

A consulta de uma estratégia possui as seguintes informações: lucro obtido, número de jogos (da estratégia)/vitórias e as ligas dessa determinada estratégia. Para além dessa informação existe uma lista de jogos (jogos das ligas selecionadas na estratégia).

Figura 19 - Página de Estratégia em Detalhe

O utilizador pode ainda consultar um determinado jogo, que caso já tenha ocorrido tem o seguinte formato (Figura 20 - Página de jogo (eventos).

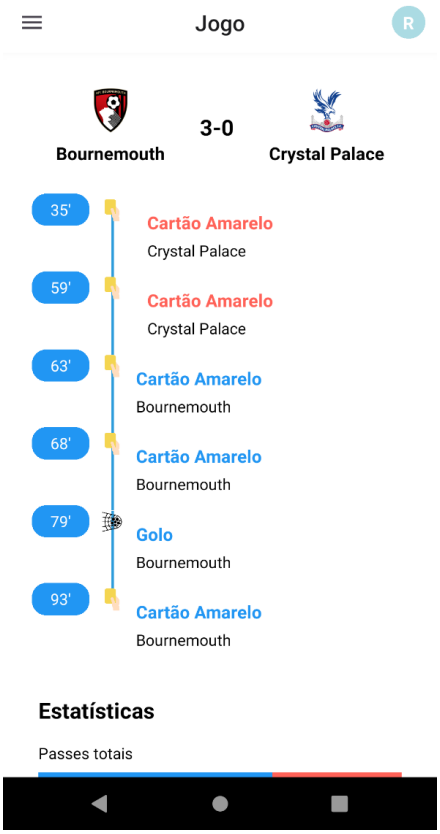


Figura 20 - Página de jogo (eventos)

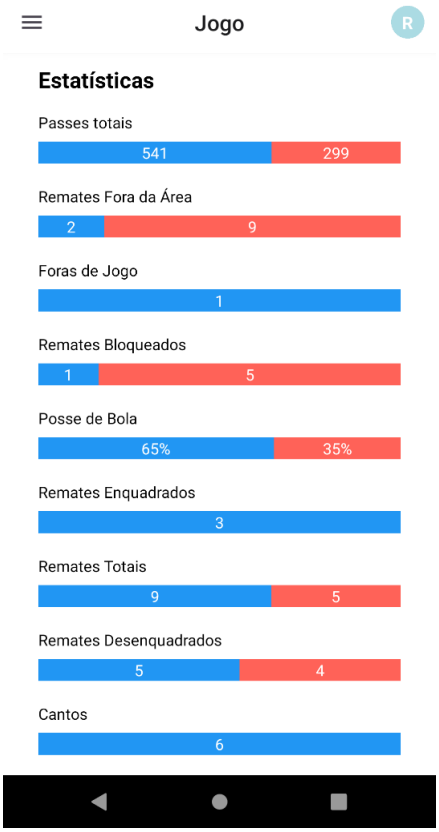


Figura 21 - Página de jogo (estatísticas)

Figura 22 - Página de jogo (últimos jogos)

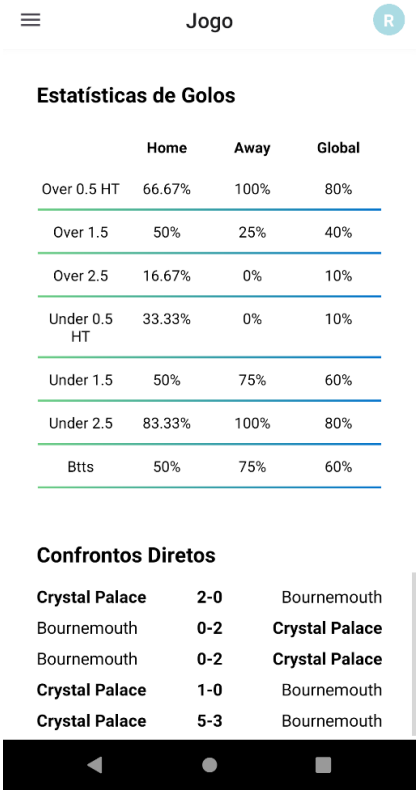


Figura 23 - Confronto Direto entre ambas as equipas

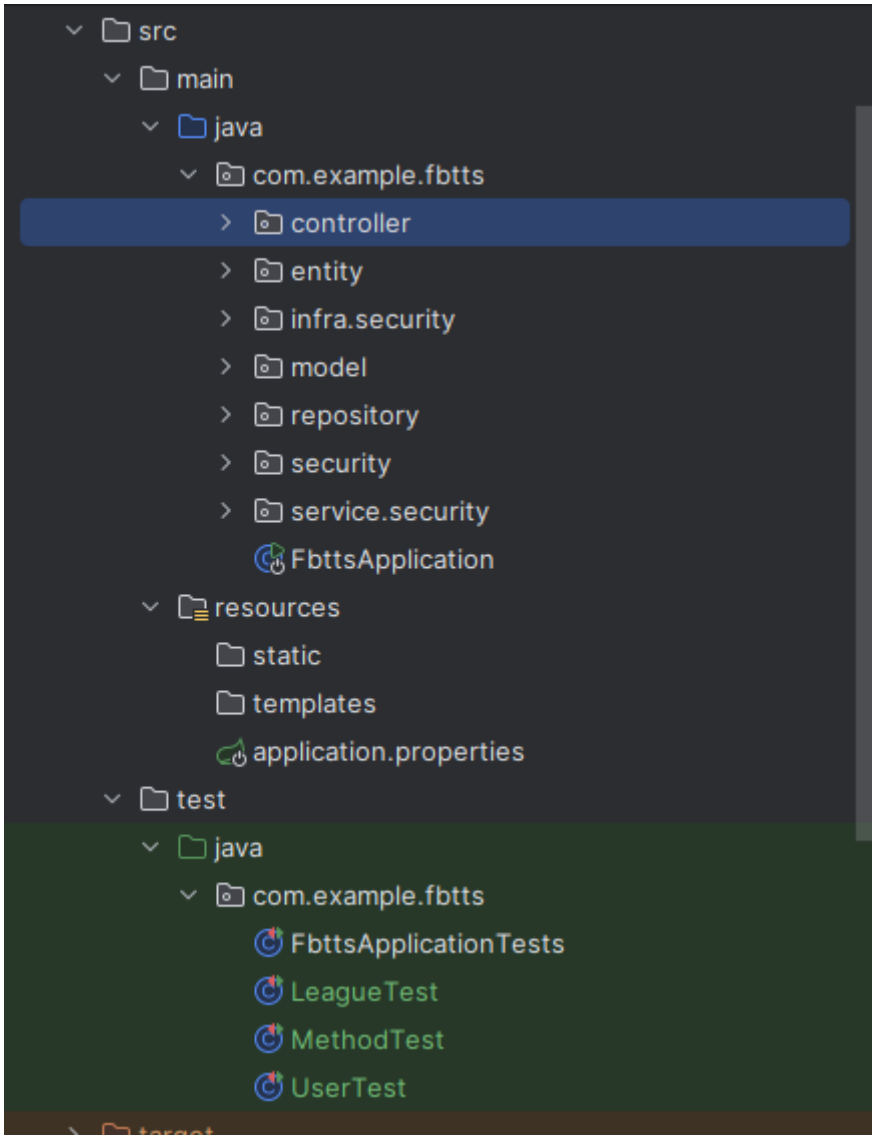


Figura 24 - Estrutura do Projeto Back-end

A estrutura do *back-end* (Spring Boot) está organizada de forma a garantir a separação de responsabilidades da aplicação. A camada *controller* é responsável por lidar com os pedidos da API e *WebServices*, direcionando-as para os serviços apropriados e retornando os dados esperados. Na camada *entity*, são definidas as classes que representam as entidades do modelo de domínio. A camada *model* abrange as classes de transferência de dados (DTOs) e outras estruturas que facilitam a comunicação entre as camadas da aplicação. A camada *repository* trata da persistência dos dados. A segurança é tratada na camada de security, onde está configurada a autenticação e autorização utilizando o Spring Security, de forma a garantir a proteção dos dados da aplicação. Ainda possui testes unitários, com o objetivo de testar os pedidos feitos à API (de forma a complementar os testes feitos manualmente através do Postman), nos vários *controllers*.

5.4.1 Ambiente Produtivo da Solução

Relativamente ao ambiente produtivo da solução, é necessário que os utilizadores finais possuam dispositivos móveis com sistemas operativos Android API 16 (Android 4.1) e acima ou, no caso de iPhones, iOS 11 e acima, com pelo menos 2GB de memória RAM. Para além disso, é necessário que esses dispositivos possuam conexões de rede estáveis.

Em termos de recursos computacionais, os utilizadores finais devem possuir dispositivos com pelo menos 2GHz de processamento e pelo menos 1 GB de espaço em disco livre para efetuar o *download* e instalar a aplicação.

5.5 Abrangência

Neste projeto serão utilizados conhecimentos aprendidos nas Unidades Curriculares de Computação Móvel e Linguagens de Programação II, no que toca ao desenvolvimento móvel e programação. Serão utilizados conhecimentos obtidos na Unidade Curricular de Interação Humano-Máquina para criação de protótipos. Para além disso, também serão aplicados conhecimentos adquiridos nas unidades curriculares de Engenharia de Software, Engenharia de Requisitos e Testes, Computação Distribuída e Segurança Informática.

6 Plano de testes e validação

Para validar o devido funcionamento da solução desenvolvida e para que esta cumpra todos os requisitos e objetivos definidos previamente, foram realizados testes aos requisitos e com utilizadores.

6.1 Testes com utilizadores

Após a fase de desenvolvimento e de testes, a aplicação encontra-se numa versão estável e capaz de ser utilizada pelo público-alvo.

Os requisitos funcionais foram testados no final da fase de desenvolvimento, antes da realização dos testes com utilizadores: os resultados completos estão na Tabela 4 - Tabela de Testes aos Requisitos. Como é mostrado, os principais requisitos, assinalados como must have, foram cumpridos.

Para verificar o cumprimento destes padrões foram realizados testes com utilizadores com a participação de 4 utilizadores.

Utilizador 1:

Perfil: Pessoa perto dos 20 anos, com experiência no que diz respeito à utilização de smartphone, com bastante conhecimento em apostas desportivas.

Experiência: Este utilizador realizou todas as tarefas propostas sem problemas. Sugeriu adicionar uma forma alternativa de navegar até ao perfil, pois pode não ser intuitivo para utilizadores com pouca experiência em utilizar aplicações móveis.

Utilizador 2:

Perfil: Pessoa com cerca de 50 anos, com alguma experiência em usar smartphones e com algum conhecimento sobre apostas desportivas.

Experiência: Este utilizador realizou as tarefas propostas pelo aluno, sem precisar de ajuda significativa. Mencionou não ter problemas em relação ao tamanho do texto da aplicação, e afirmou que as cores utilizadas na mesma eram bastante apelativas. Referiu que a aplicação era útil, contudo demorou um pouco a perceber a técnica de navegação utilizada na aplicação (menu hamburger).

Utilizador 3:

Perfil: Pessoa perto dos 20 anos, com experiência em usar smartphones, não costuma fazer análise para apostar (aposta para se divertir).

Experiência: Este utilizador conseguiu realizar as tarefas propostas com alguma ajuda. Sugeriu colocar painéis para distinguir secções em páginas que utilizam scroll. Achou o método de navegação intuitivo e bem estruturado, sugeriu também a implementação de dark mode.

Utilizador 4:

Perfil: Pessoa com cerca de 30 anos com bastante experiência no que toca à utilização de aplicações móveis. Pouca experiência relativamente a apostas desportivas.

Experiência: Este utilizador realizou as tarefas propostas sem dificuldades, contudo afirmou que pode ser confuso para pessoas com nenhum/pouco conhecimento acerca de apostas desportivas (que era o seu caso).

6.2 Testes Unitários

O principal objetivo dos testes unitários é garantir a integridade das diferentes funcionalidades implementadas nos *controllers* da aplicação, como por exemplo no "LeagueController". Estes testes têm como objetivo:

- Verificar se os endpoints respondem corretamente às requisições HTTP.
- Validar se os dados retornados estão no formato JSON esperado.
- Assegurar que o comportamento da API está de acordo com os requisitos especificados.

```
    @Test  Rodrigo-Taciano-a22204447 *
    @WithMockUser(username = "roman", roles = {"USER"})
    public void whenGetLeagues_thenReturnJson() throws Exception {
        String filePath = BASE_PATH + "leagues.json";
        File file = new File(filePath);
        byte[] jsonData = Files.readAllBytes(file.toPath());
        String expectedContent = new String(jsonData, StandardCharsets.UTF_8);

        mockMvc.perform(get( urlTemplate: "/leagues")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(expectedContent));
    }

    @Test  Rodrigo-Taciano-a22204447 *
    @WithMockUser(username = "roman", roles = {"USER"})
    public void whenGetLeagueDetailed_thenReturnJson() throws Exception {
        String filePath = BASE_PATH + "bundesliga1.json";
        File file = new File(filePath);
        byte[] jsonData = Files.readAllBytes(file.toPath());
        String expectedContent = new String(jsonData, StandardCharsets.UTF_8);

        mockMvc.perform(get( urlTemplate: "/leaguedetailed")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(expectedContent));
    }
}
```

Figura 25 - Testes Unitários à classe LeagueController

```java
    @Test  Rodrigo-Taciano-a22204447 *
    @WithMockUser(username = "roman", roles = {"USER"})
    public void whenGetMethods_thenReturnJson() throws Exception {
        String filePath = BASE_PATH + "methods.json";
        File file = new File(filePath);
        byte[] jsonData = Files.readAllBytes(file.toPath());
        String expectedContent = new String(jsonData, StandardCharsets.UTF_8);

        mockMvc.perform(get( urlTemplate: "/methods")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(expectedContent));
    }

    @Test  Rodrigo-Taciano-a22204447 *
    @WithMockUser(username = "roman", roles = {"USER"})
    public void whenGetMatches_thenReturnJson() throws Exception {
        String filePath = BASE_PATH + "matches.json";
        File file = new File(filePath);
        byte[] jsonData = Files.readAllBytes(file.toPath());
        String expectedContent = new String(jsonData, StandardCharsets.UTF_8);

        mockMvc.perform(get( urlTemplate: "/methods/matches")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(expectedContent));
    }
}
```

Figura 26 - Testes Unitários à classe MethodController

7 Método e Planeamento

O planeamento inicial apresentado na 1ª entrega intercalar dividia o desenvolvimento deste TFC está dividido em três partes:

- Fase de Análise/Levantamento de Requisitos;
- Fase de Implementação;
- Fase de Testes.

As etapas propostas apontavam para as quatro entregas obrigatórias do TFC, com o objetivo de fasear o desenvolvimento em secções que atendessem às entregas. O planeamento do inicial do TFC pode ser visto na Figura 27 – Planeamento Inicial.

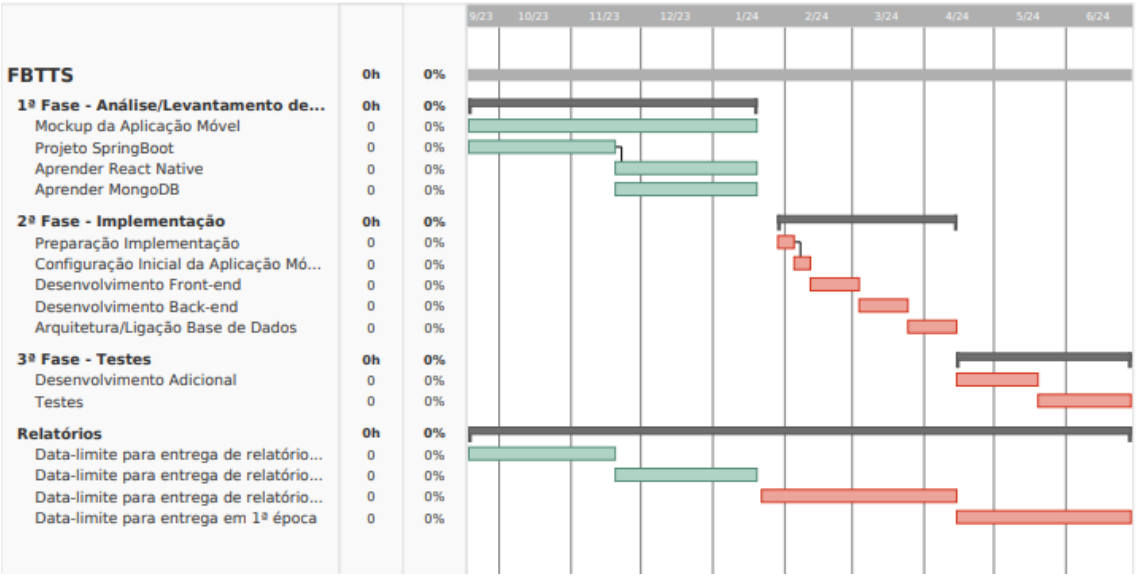


Figura 27 – Planeamento Inicial

Com a existência de atrasos e situações inesperadas nas 2ª e 3ª fase, houve várias mudanças nas diferentes fases do desenvolvimento. Essas alterações deram origem ao calendário de desenvolvimento final Figura 28 - Calendário Final (imagem maior no anexo).

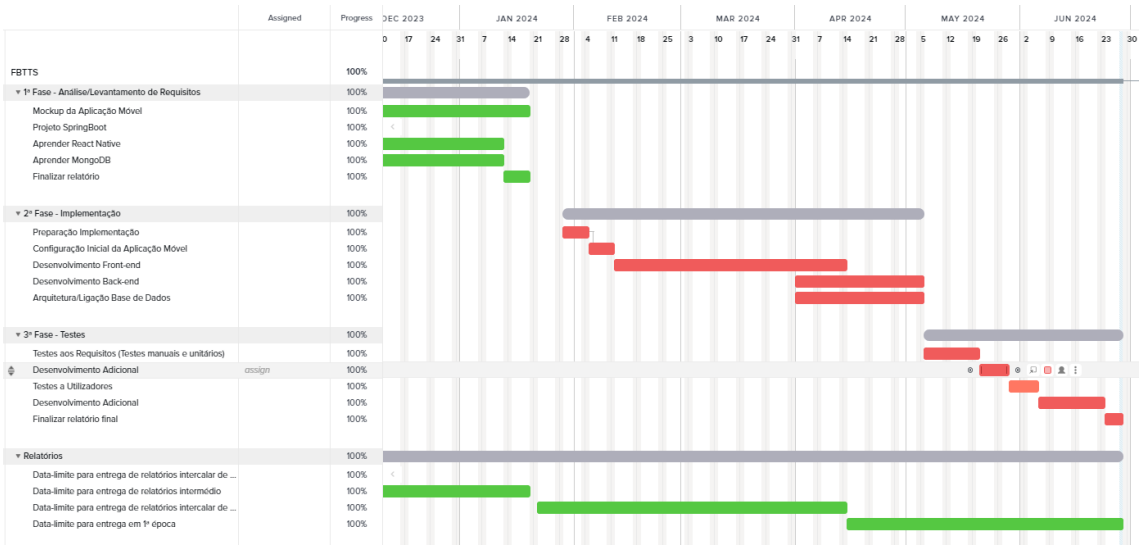


Figura 28 - Calendário Final

Bibliografia

[1] MongoDB, www.mongodb.com, abril 2024

[2] FBTTS, www.fbtts.pt, junho 2024

[3] SpringBoot, www.spring.io/projects/spring-boot, abril 2024

[4] "Why Spring Boot?", *https://spring.io/why-spring/*, abril 2024

[5] What is MoSCoW Prioritization https://airfocus.com/glossary/what-is-moscow-prioritization/, abril 2024

[6] Practical Fibonacci, *https://www.scrum.org/resources/blog/practical-fibonacci-beginners-guide-relative-sizing*, abril 2024

[7] React Native, *https://reactnative.dev/*, abril 2024

[8] Amazon Web Services, *https://aws.amazon.com/*, dezembro 2023

[9] Maven, *https://maven.apache.org/*, dezembro 2023

[10] Apache Tomcat, *https://tomcat.apache.org/*, dezembro 2023

[11] Why Spring, https://spring.io/why-spring, janeiro 2024

[12] RESTful Web Services – A Question of Standards https://ieeexplore.ieee.org/document/8540763, jan. 2024

[13] StackOverflow, https://stackoverflow.com, abril 2024

[14] Testing in SpringBoot, https://www.baeldung.com/spring-boot-testing, maio 2024

[15] Statista, https://www.statista.com/outlook/amo/online-gambling/online-sports-betting/worldwide, junho 2024

[16] TeamGantt, https://www.teamgantt.com, junho 2024

[17] Playlist de tutoriais de react native, https://www.youtube.com/playlist?list=PL4cUxeGkcC9ixPU-QkScoRBVxtPPzVjrQ, abril 2024

Anexo A – Progresso do Trabalho

Na imagem, podemos ver as tarefas concluídas a verde e as tarefas por concluir a vermelho. Este calendário inclui a entrega deste Relatório Intermédio como tarefa concluída.

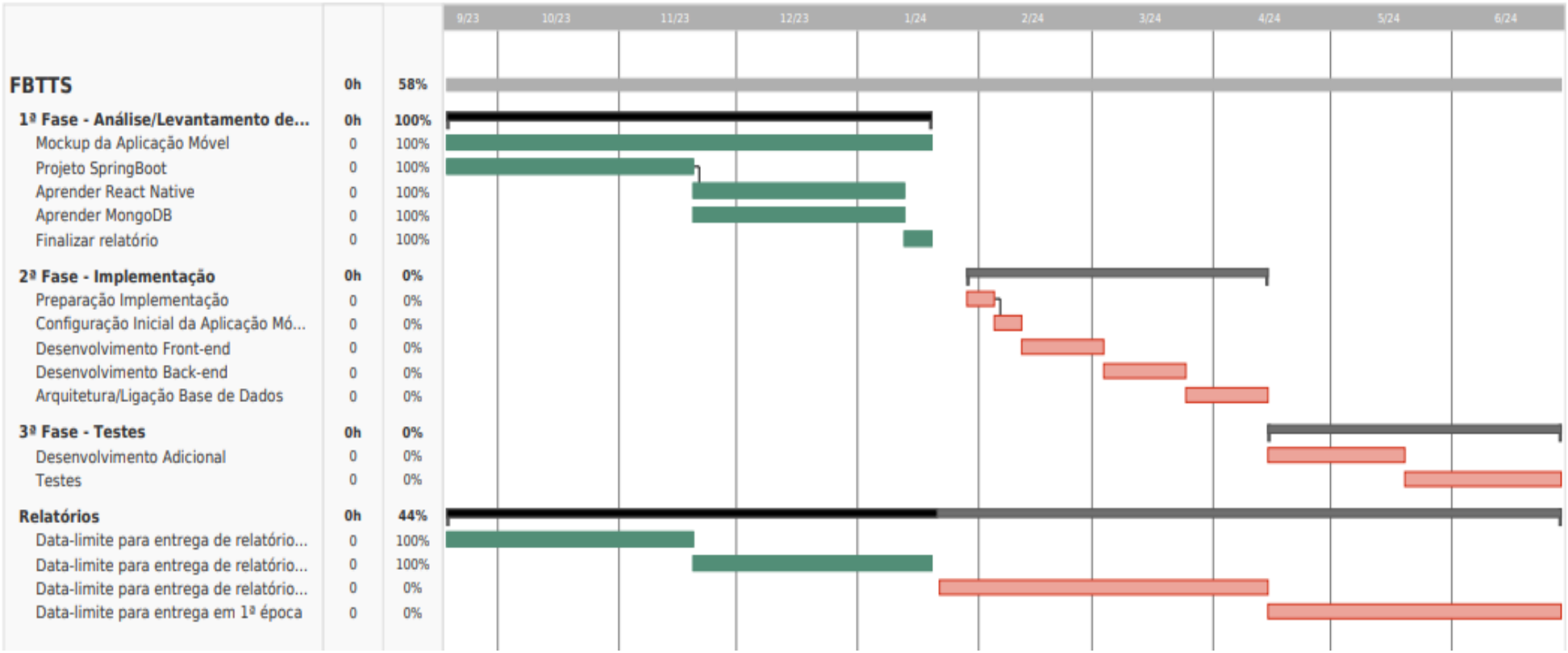


Figura 29 - Gannt planeamento de projeto com tarefas concluídas

Anexo B – Diagrama de Classes FBTTS

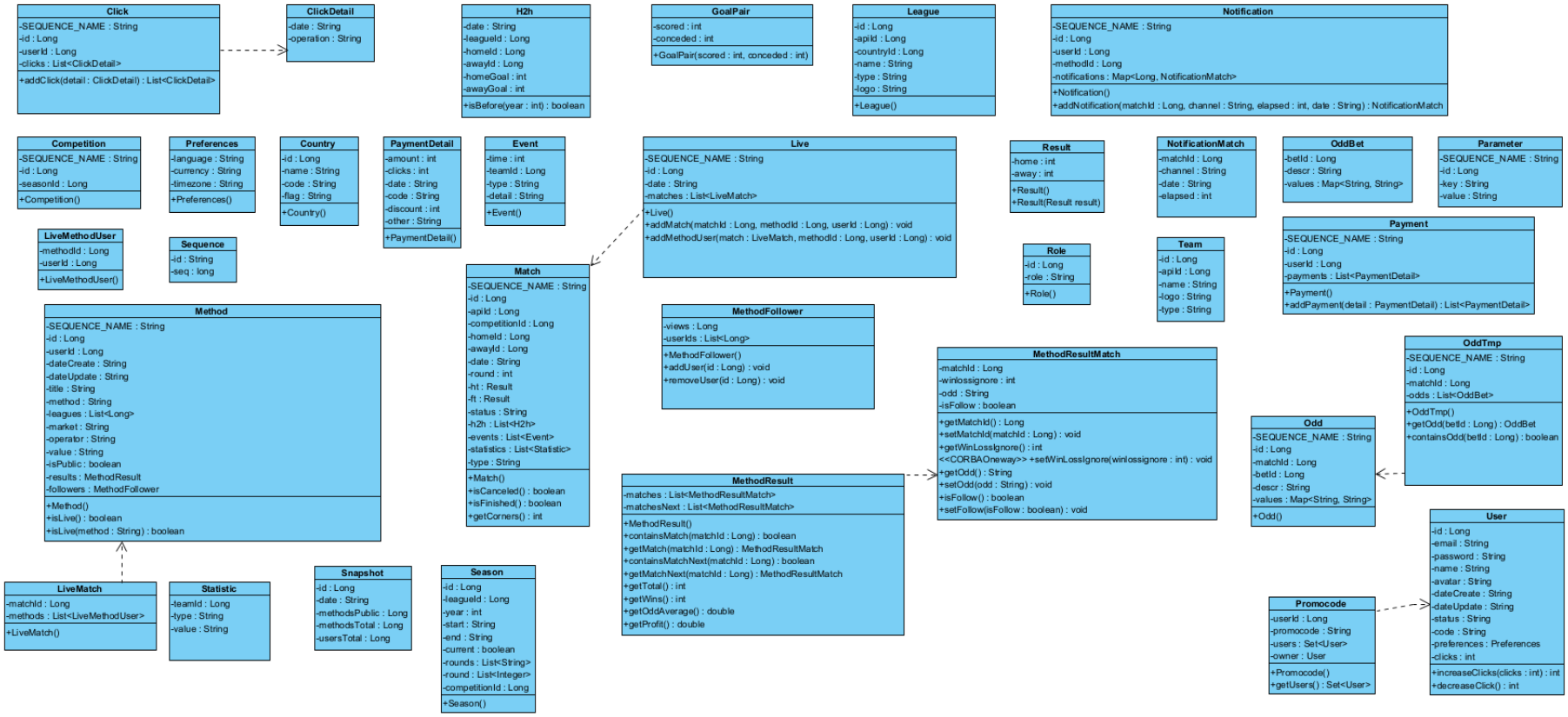


Figura 30 - Diagrama de Classes FBTTS

Anexo C – Guião de Tarefas

Introdução

Este guião contém 5 tarefas e possui um tempo estimado de 10 minutos.

Os dados recolhidos nesta sessão serão avaliados de forma anónima, além disso pode desistir do teste a qualquer momento.

Tarefa 1

Contexto:

É um utilizador da FBTTS e deseja consultar uma estratégia.

Descrição:

Selecione uma estratégia e indique ao observador dos testes que ligas estão associadas à estratégia.

Tarefa 2

Contexto:

Deseja consultar um determinado jogo da estratégia.

Descrição:

Clique num determinado jogo e indique ao observador dos testes que equipas estão nesse jogo.

Tarefa 3

Contexto:

Deseja consultar uma determinada liga.

Descrição:

Navegue até às ligas e consulte uma.

Tarefa 4

Contexto:

Pretende consultar informação sobre o perfil.

Descrição:

Navegue até à página do perfil..

Tarefa 5

Contexto:

Deseja observar os jogos do dia

Descrição:

Indique ao observador de testes que jogos irão decorrer nesse dia.

Anexo D – Imagens de Código Desenvolvido

```java
@RestController  Rodrigo-Taciano-a22204447 *
public class UserController {

    @Autowired
    UserRepository userRepository;

    @PostMapping("/addUser")  Rodrigo-Taciano-a22204447
    public void addUser(@RequestBody User user){
        userRepository.save(user);
    }

    @GetMapping("/{email}/cliques")  new *
    public ResponseEntity<Integer> getUserCliques(@PathVariable String email) {
        User user = userRepository.findByEmail(email);
        if (user != null) {
            return ResponseEntity.ok(user.getClicks());
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

    @PostMapping("/{email}/decrement-clicks")  new *
    public ResponseEntity<?> decrementUserClicks(@PathVariable String email) {
        User user = userRepository.findByEmail(email);
        if (user != null) {
            if (user.getClicks() > 0) {
                user.setClicks(user.getClicks() - 1);
                userRepository.save(user);
                return ResponseEntity.ok(user.getClicks());
            } else {
                return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("No more clicks left");
            }
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("User not found");
        }
    }

}
```

Figura 31 - UserController

```java
@Getter  24 usages  Rodrigo-Taciano-a22204447 *
@Setter
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(of = "id")
@Document(collection = "users")
public class User implements UserDetails {

    @Transient
    public static final String SEQUENCE_NAME = "user_sequence";

    @Id
    private Long id;

    private String type = "BetStrategy";
    private String email;
    private String username;
    private String password;
    private UserRole role;

    private int clicks;
    @DBRef
    private List<Method> methods = new ArrayList<>();
    private boolean hasPayments;

    private String language;
    private String currency;

    public User(String username, String email, String password, UserRole role) {
        this.username = username;
        this.email = email;
        this.password = password;
        this.role = role;
        this.clicks = 75;
    }
```

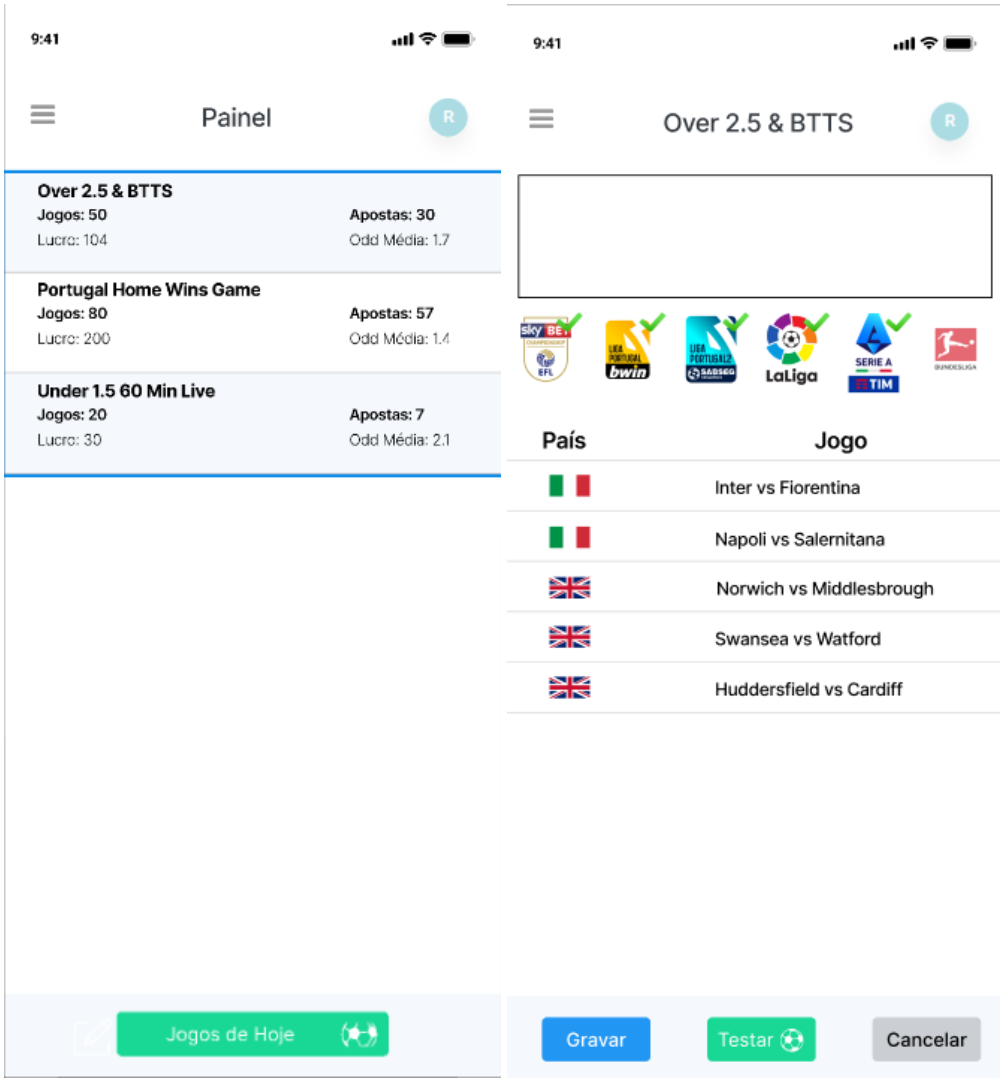
Figura 32 - User

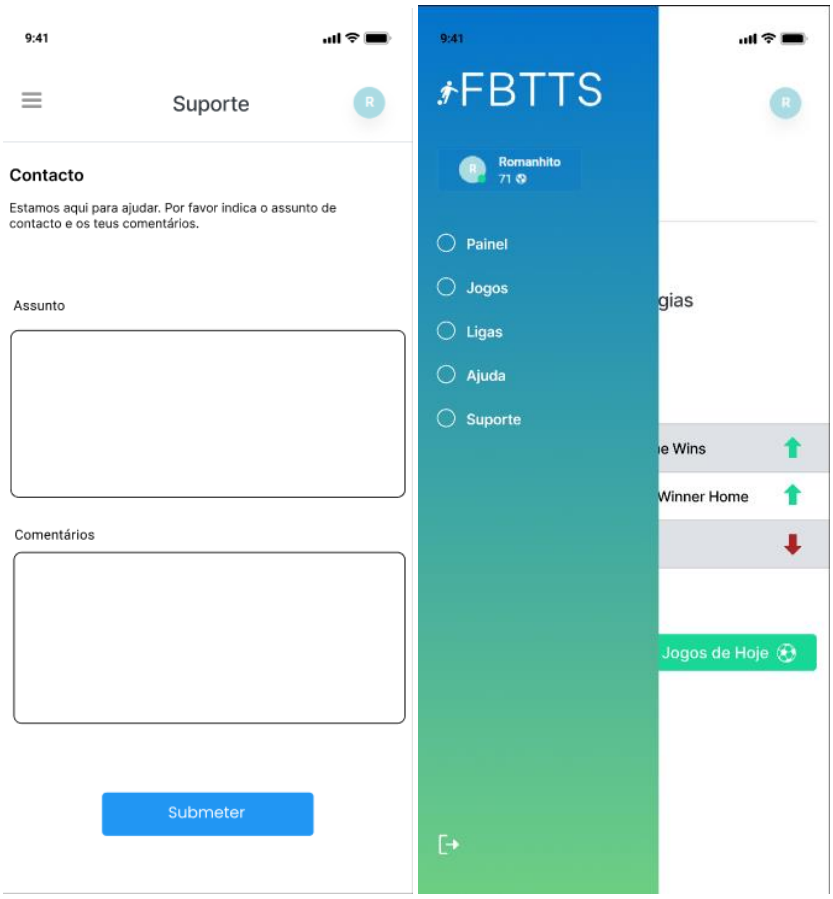
```jsx
export const AuthProvider = ({ children }) => {
    const [isLoading : boolean , setIsLoading] = useState( initialState: false);
    const [userToken, setUserToken] = useState( initialState: null);
    const [userInfo, setUserInfo] = useState( initialState: null);

    1+ usages  Rodrigo-Taciano-a22204447
    const login = (email, password) : void  => {
        setIsLoading( value: true);
        axios.post( url: `${BASE_URL}/login`,  data: {
            email,
            password
        }) Promise<AxiosResponse<...>>
            .then(res : AxiosResponse<any>  => {
                let userInfo = res.data;
                setUserInfo(userInfo);
                setUserToken(userInfo.token);
                AsyncStorage.setItem( key: 'userInfo', JSON.stringify(userInfo));
                AsyncStorage.setItem( key: 'userToken', userInfo.token);
            }) Promise<void>
            .catch(e => {
                console.log(`Login error ${e}`);
            });
        setIsLoading( value: false);
    };
```

Figura 33 - Login Front-end

Anexo E – Mockup




9:41
Suporte
R
Contacto
Estamos aqui para ajudar. Por favor indica o assunto de contacto e os teus comentários.
Assunto
Comentários
Submeter
9:41
FBTTS
Romanhito
71
Painel
Jogos
Ligas
Ajuda
Suporte
gias
e Wins
Winner Home
Jogos de Hoje


9:41
Jogos
R
2023-10-24
País
Jogo
Pisa vs Lecco
Brescia vs Modena
Norwich vs Middlesbrough
Swansea vs Watford
Huddersfield vs Cardiff

Anexo F – Testes aos Requisitos

Os testes nesta secção têm como objetivo validar o funcionamento correto dos requisitos funcionais anteriormente definidos no capítulo 4.1. Os testes na tabela abaixo englobam as principais funcionalidades que a aplicação móvel FBTTS deve oferecer aos seus utilizadores.

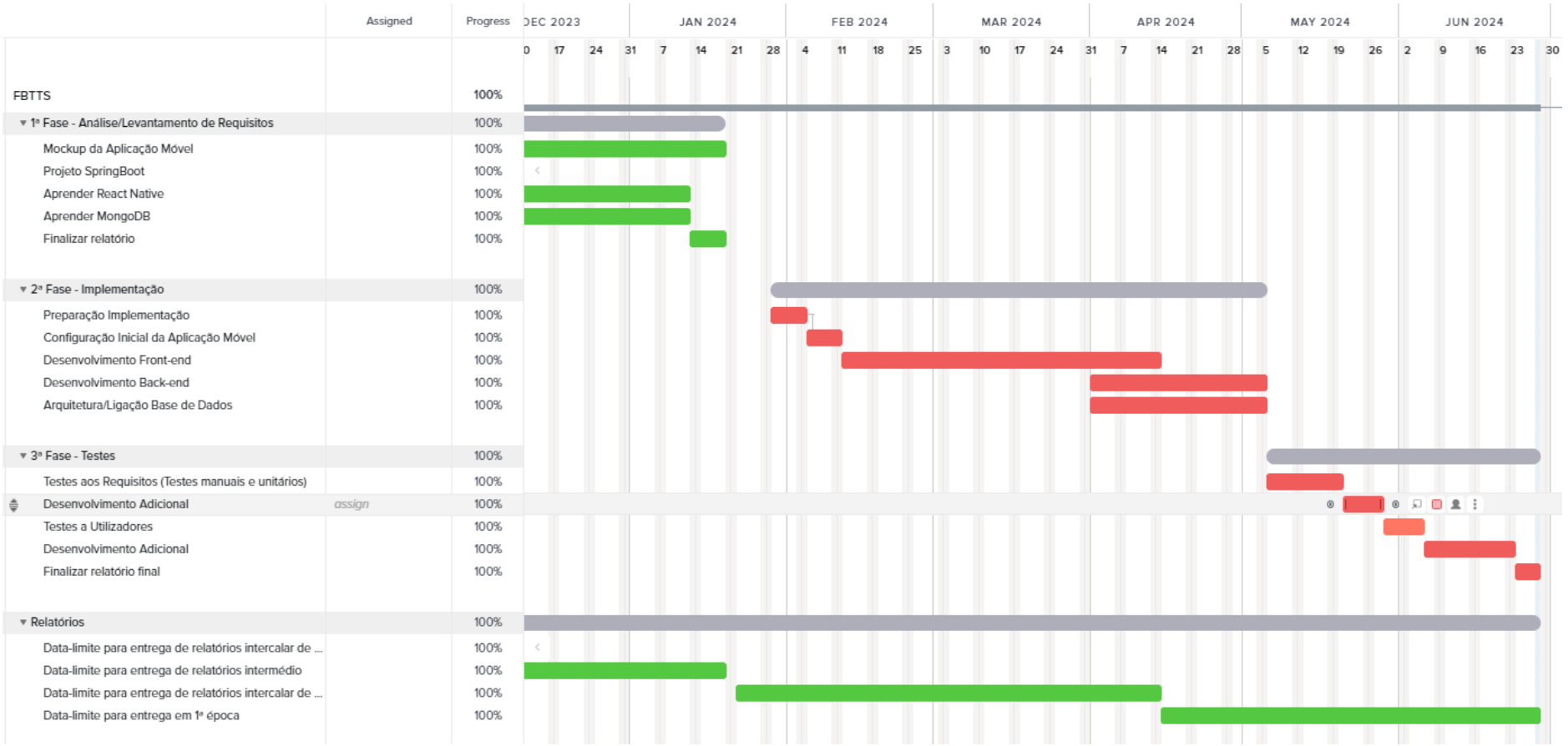
Dada por concluída esta realização de testes, caso tudo se encontre em condições de avançar serão feitos testes com utilizadores finais.

Tabela 4 - Tabela de Testes aos Requisitos

Título	Descrição	Requisitos	Resultado Esperado	Obtido
Autenticação	O utilizador acede à aplicação com as credenciais válidas	R01	O utilizador acede à dashboard caso introduza credenciais válidas. Caso introduza cradenitis inválidas, o acesso à aplicação não é autorizado.	Log in efetuado com sucesso e utilizador redirecionado para a página inicial. Credenciais inválidas não permitem acesso
Consultar Estratégias a partir da dashboard	Na Dashboard (página inicial) são exibidas as estratégias do utilizador	R04	Na página principal (Dashboard) é apresentado o nome das estratégias do utilizador.	É exibida a estratégia e os seus detalhes
Consultar Estratégia em detalhe	O utilizador consulta detalhadamente a sua estratégia.	R06	Na página Ver Estratégia é apresentada mais informação sobre uma estratégia, bem como os jogos associados à mesma.	Foi possível consultar a estratégia em detalhe
Consultar Jogo	O utilizador consulta um jogo em detalhe	R14	Na página Jogo são apresentados não só detalhes do jogo, mas também	É exibido o jogo e os seus detalhes (como o resultado e as estatísticas

			detalhes dos desempenhos das equipas ao longo dos últimos jogos.	associadas ao mesmo)
Consultar Menu	O utilizador consulta o menu hamburger.	R16	O menu é mostrado.	Após clicar no botão do menu, o mesmo é exibido e o utilizador pode navegar para as várias páginas da aplicação.

Anexo G – Calendário Final


Assigned
Progress
DEC 2023
JAN 2024
FEB 2024
MAR 2024
APR 2024
MAY 2024
JUN 2024
0 17 24 31 7 14 21 28 4 11 18 25 3 10 17 24 31 7 14 21 28 5 12 19 26 2 9 16 23 30
FBTTS 100%
1ª Fase - Análise/Levantamento de Requisitos 100%
Mockup da Aplicação Móvel 100%
Projeto SpringBoot 100%
Aprender React Native 100%
Aprender MongoDB 100%
Finalizar relatório 100%
2ª Fase - Implementação 100%
Preparação Implementação 100%
Configuração Inicial da Aplicação Móvel 100%
Desenvolvimento Front-end 100%
Desenvolvimento Back-end 100%
Arquitetura/Ligação Base de Dados 100%
3ª Fase - Testes 100%
Testes aos Requisitos (Testes manuais e unitários) 100%
Desenvolvimento Adicional assign 100%
Testes a Utilizadores 100%
Desenvolvimento Adicional 100%
Finalizar relatório final 100%
Relatórios 100%
Data-limite para entrega de relatórios intercalar de ... 100%
Data-limite para entrega de relatórios intermédio 100%
Data-limite para entrega de relatórios intercalar de ... 100%
Data-limite para entrega em 1ª época 100%

Glossário

LEI	Licenciatura em Engenharia Informática
AWS	Amazon Web Services
TFC	Trabalho Final de Curso
FBTTS	Football BeTTing Strategies
PWA	Progressive Web Apps