UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JORGE LUIZ FRANZON ROSSI

PLATAFORMA DE SOFTWARE PARA GERENCIAR PLACARES EM TORNEIOS AMADORES DE TÊNIS

CAMPO MOURÃO

2021

JORGE LUIZ FRANZON ROSSI

PLATAFORMA DE SOFTWARE PARA GERENCIAR PLACARES EM TORNEIOS AMADORES DE TÊNIS

Software platform for managing scoreboards in amateur tennis tournaments

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Rafael Liberato Roberto

CAMPO MOURÃO

2021

JORGE LUIZ FRANZON ROSSI

PLATAFORMA DE SOFTWARE PARA GERENCIAR PLACARES EM TORNEIOS AMADORES DE TÊNIS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 08/julho/2021

Ivanilton Polato
Doutorado
UTFPR

Lucio Geronimo Valentin
Doutorado
UTFPR

Rafael Liberato Roberto
Doutorado
UTFPR

CAMPO MOURÃO

2021

RESUMO

A prática regular de atividade física é de grande importância para a manutenção de uma boa qualidade de vida, trazendo inúmeros benefícios para a saúde. O esporte é um dos tipos de atividades físicas mais conhecidos, podendo ser praticado de duas formas distintas: o esporte amador e o esporte profissional. No contexto do esporte amador, o tênis tem se destacado pelo considerável aumento da popularidade da prática dessa modalidade como recreação no Brasil, sendo na maior parte das vezes promovido por academias especializadas. Visando manter os praticantes estimulados na prática do esporte, estas academias utilizam-se de diversas estratégias de engajamento, como a organização de torneios, manutenção de *rankings* e a adoção de recursos utilizados no esporte profissional. Tratando-se especificamente da última estratégia, existe uma certa dificuldade em encontrar opções em que a relação custo-benefício seja adequada à realidade das academias, pois muitas vezes as soluções têm um alto custo e fornecem recursos desnecessários ao esporte amador, ou são muito simples, não apresentando características do âmbito profissional. Com base nisso, e também a partir do contato com academias de tênis da cidade de Campo Mourão, este trabalho consiste no desenvolvimento de uma plataforma de software para utilização em torneios promovidos pelas academias entre os seus associados. A plataforma consiste em módulos para controlar o placar das partidas, transmitir a pontuação dos jogos em tempo real, armazenar as estatísticas das partidas e disponibilizar o sistema como um serviço.

Palavras-chaves: Tecnologia no esporte. Sistemas na nuvem.

ABSTRACT

The regular practice of physical activity is of great importance for the maintaince of a good quality of life, bringing countless benefits to health. Sport is one of the best known types of physical activities and can be practiced in two different ways: amateur sport and professional sport. In the context of amateur sport, tennis has been highlighted by the considerable increase in the popularity of this modality as a recreation in Brazil, being most often promoted by specialized academies. Aiming to keep practitioners stimulated in the practice of sport, these academies use various engagement strategies, such as organizing tournaments, maintaining rankings and adopting resources used in professional sports. With regard specifically to the last strategy, there is a certain difficulty in finding options in which the cost-beneffit ratio is adequate to the reality of the academies, as the solutions often have a high cost and provide unnecessary resources for the amateur sport, or are very simple, without features of the professional scope. Based on this, and also from the contact with tennis academies in the city of Campo Mourão, this work consists in the development of a software platform for use in tournaments promoted by the academies among their members. The platform consists of modules to control the scoreboard of the matches, transmit game scores in real time, store match statistics and make the system available as a service.

Keywords: Technology in sport. Cloud systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMQP: *Advanced Message Queuing Protocol.* 13
API: *Application Programming Interface.* 34

CBT: Confederação Brasileira de Tênis. 10
CSS: *Cascading Style Sheets.* 16, 17

HTML: *Hyper Text Markup Language.* 16, 17
HTTP: *Hypertext Transfer Protocol.* 13, 16, 17, 20, 32, 40, 43

IoT: *Internet of Things.* 13, 16, 28, 31, 32
IP: *Internet Protocol.* 13, 20
ITF: *International Tennis Federation.* 10

JSON: *JavaScript Object Notation.* 14, 19

LAN: *Local Area Network.* 22

MQTT: *Message Queuing Telemetry Transport.* 13–16, 31–33, 44

OMS: Organização Mundial da Saúde. 8

QoS: *Quality of Service.* 15

REST: *Representational State Transfer.* 32, 43

SGBD: Sistema de Gerenciamento de Bancos de Dados. 20
SPI: *Serial Peripheral Interface.* 28
SQL: *Structured Query Language.* 18, 33

TCP: *Transmission Control Protocol.* 13

URL: *Uniform Resource Locator.* 17
UTF-8: *8-bit Unicode Transformation Format.* 16

XMPP: *Extensible Messaging and Presence Protocol.* 13

SUMÁRIO

1 INTRODUÇÃO

A importância da prática regular de atividade física é consenso entre todas as organizações de saúde no mundo. Segundo a Organização Mundial da Saúde (OMS), a prática regular de atividades físicas traz benefícios significativos para a saúde, proporcionando um melhor condicionamento muscular e cardiorrespiratório, uma melhor saúde óssea, um menor risco de hipertensão, entre muitos outros aspectos positivos para o corpo (WHO, 2018). A OMS define atividade física como qualquer movimento corporal produzido pelos músculos esqueléticos que requer gasto energético.

O esporte enquadra-se como uma subcategoria de atividade física que inclui aspectos de desempenho e competição. A prática do esporte pode se apresentar em duas modalidades distintas: o esporte amador (ou recreativo) e o esporte profissional (ou de alto rendimento). O esporte profissional visa o alcance do desempenho máximo em busca de vitórias e recordes, enquanto o esporte amador tem como objetivo primordial a promoção dos benefícios da atividade física à saúde dos seus praticantes. No contexto do esporte amador, o tênis destaca-se por ser considerado um esporte completo, pois além dos benefícios ao corpo, também estimula capacidades cerebrais e psicológicas. Além disso, a popularidade desse esporte como recreação tem aumentado consideravelmente no Brasil (EXAME, 2015).

A prática do tênis na modalidade recreativa é promovida usualmente em clubes recreativos, associações desportivas e academias especializadas. Um dos grandes desafios enfrentados por estas organizações, caracteriza-se por manter os praticantes estimulados e engajados na prática do esporte. Diversas estratégias são empregadas nesse sentido, como por exemplo, a organização de torneios, a manutenção de uma classificação ordenada de pontuação (*ranking*) e a adoção de recursos utilizados no esporte profissional. Mais especificamente, na terceira estratégia, existe uma certa dificuldade na utilização de tecnologias da modalidade profissional do tênis entre os atletas amadores. Essa dificuldade ocorre pela falta de opções cuja relação custo-benefício seja adequada à realidade das academias. Por vezes, existe uma dicotomia entre as opções existentes, ou a solução é muito sofisticada com um alto custo e recursos desnecessários ao esporte amador, ou a solução é muito simples na qual as características atrativas do âmbito profissional não estão presentes.

Diante disso, este trabalho faz parte de uma solução tecnológica idealizada a partir de contatos com academias de tênis na cidade de Campo Mourão, com intuito de proporcionar aspectos do esporte profissional aos esportistas amadores. A solução consiste no desenvolvimento de uma plataforma de software para utilização em torneios promovidos pelas academias entre os seus associados. O projeto é composto por um conjunto de módulos para controlar o placar das partidas, transmitir a pontuação em tempo real, armazenar as estatísticas das partidas e disponibilizar o sistema como um serviço. Além disso, a arquitetura do sistema foi concebida de modo a permitir que um placar eletrônico físico possa ser acoplado e manipulado pela plataforma como um dispositivo IoT.

O restante do trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados os conceitos que fundamentam o trabalho. O Capítulo 3 apresenta as soluções existentes que são similares à solução contida neste trabalho. O Capítulo 4 descreve o desenvolvimento do trabalho. Por fim, o Capítulo 5 apresenta as conclusões obtidas a partir do desenvolvimento.

2 FUNDAMENTOS

O mundo esportivo tem incorporado cada vez mais inovações tecnológicas, as quais contribuem para uma melhor experiência e performance dos praticantes, tanto amadores quanto profissionais. Essa tendência pode ser observada nas mais diversas modalidades, incluindo o tênis, no qual a tecnologia está presente em vários aspectos, como nas quadras e raquetes. Um exemplo conhecido no esporte é o Hawk-Eye (FREDERICO, 2017), um sistema que traça visualmente a trajetória da bola, permitindo saber com precisão em qual ponto a quadra foi atingida. Além disso, dispositivos acoplados nas raquetes, como o QLIPP (KELLY, 2017), permitem que diversas informações das jogadas sejam coletadas, como o tipo da batida, velocidade, giro e precisão.

O desenvolvimento de dispositivos, dos mais variados tipos, que coletam e trocam informações não é um conceito limitado ao esporte. A Internet das Coisas se refere a tendência atual de conectar todos os tipos de objetos físicos a internet, como utensílios domésticos, dispositivos vestíveis e cidades inteligentes (REDHAT, 2020).

2.1 Tênis

O tênis se destaca como uma das modalidades que mais tem ganhado popularidade no Brasil. Segundo a *International Tennis Federation* (ITF), órgão máximo do esporte, em 2019 haviam 2.300.000 praticantes no país[1]. Nesta seção serão resumidas algumas das principais regras e fundamentos, disponibilizadas em português pela Confederação Brasileira de Tênis (CBT)[2].

2.1.1 A quadra

Para os jogos simples (um jogador contra o outro), a quadra deve ter 23,77 metros de comprimento por 8,23 de largura, enquanto nos jogos de duplas a largura deve ser de 10,97 metros. É dividida ao meio por uma rede completamente estendida, numa altura de 1,07 metros. A quadra é delimitada por linhas presentes no final, nas laterais e linhas de serviço (paralelas a rede, entre as linhas laterais). Além disso, a área entre as linhas de serviço e a rede é dividida na metade por uma linha central.

2.1.2 Contagem no *game*

A contagem padrão consiste na seguinte ordem:

- **Sem ponto**: Zero
- **Primeiro ponto**: 15
- **Segundo ponto**: 30

[1] http://itf.uberflip.com/i/1169265-itf-global-tennis-report-2019-nations-a-k/59
[2] http://cbt-tenis.com.br/arquivos/seniors/seniors_5a1c3b134e691_27112017_141931.pdf

- **Terceiro ponto**: 40
- **Quarto ponto**: Game

Se ambos os jogadores atingirem 40 pontos, usa-se o termo “Iguais”. Nesse caso, após fazer um ponto, o jogador fica em “Vantagem”, vencendo o *game* se ganhar o próximo ponto. Caso o jogador esteja em “Vantagem” e o oponente vença o ponto, a contagem volta para “Iguais”.

Durante o *tie-break*, a contagem é realizada de forma incremental (“Zero”, “1”, “2”, “3”, etc), até que um dos jogadores alcance 7 pontos, desde que tenha uma margem de 2 pontos sobre o adversário.

2.1.3 Contagem no *set*

Os dois principais métodos de contagem de *set* utilizados são o *set* longo e o *set* com *tie-break*. Qualquer um pode ser utilizado, desde que seja anunciado antes do começo da competição.

- ***Set* longo**: vence o *set* o jogador que alcançar 6 *games* primeiro, desde que tenha uma margem de 2 *games* sobre o oponente. Se for preciso, o *set* deve continuar até essa diferença ser atingida.
- ***Set* com *tie-break***: vence o *set* o jogador que alcançar 6 *games* primeiro, desde que tenha uma margem de 2 *games* sobre o oponente. Se a contagem empatar em 6 *games* iguais, um *tie-break* deve ser realizado.

2.1.4 Contagem no jogo

O jogo pode ser jogado em melhor de 3 *sets* (comum em partidas amadoras), ou melhor de 5 *sets* (comum em campeonatos profissionais).

2.1.5 Troca de lados

O lado inicial dos jogadores é sorteado antes do começo da partida. Logo após o primeiro *game*, os jogadores trocam de lado. Após isso, os jogadores trocam de lado a cada dois *games*.

2.1.6 Saque

O jogador que irá sacar deve se posicionar antes da linha do fundo da quadra, inicialmente do lado direito (alternando a cada ponto). Após posicionar-se e aguardar o adversário estar preparado, o jogador deve lançar a bola para cima e a acertar com a raquete antes que caia no chão. O ponto é ganho se a bola passar a rede, atingir a área de serviço diagonalmente oposta e o adversário não conseguir retornar o saque.

Se o jogador não conseguir atingir a bola ou ela cair fora da área de serviço diagonalmente oposta, pode executar uma segunda tentativa (“Segundo saque”). Se também errar nessa tentativa, o jogador perde o ponto.

2.2 Internet das Coisas

A ideia de interconectar dispositivos inteligentes começou nas décadas de 1980 e 1990, porém os avanços eram lentos devido às limitações tecnológicas da época, tendo em vista que os componentes eletrônicos eram muito grandes e não existia uma forma eficaz de comunicação entre os mesmos. O termo Internet das Coisas foi utilizado pela primeira vez em 1999 por Kevin Ashton, para descrever um sistema no qual a Internet está conectada com o mundo físico por meio de sensores presentes em todos os lugares.

Com a difusão das redes sem fio e a redução dos custos dos computadores, tornou-se possível transformar os mais diversos tipos de objetos do cotidiano em dispositivos pertencentes à Internet das Coisas, desde uma lâmpada que pode ser controlada pelo celular até cidades inteligentes inteiras interconectadas. Atualmente, já existem mais coisas conectadas do que pessoas no mundo, e estimativas apontam que em 2025 existirão cerca de 41.6 bilhões de dispositivos conectados (RANGER, 2020).

Segundo Chebudie et al. (2014), um sistema deve ter o seguinte conjunto de atributos para ser considerado como um sistema da Internet das Coisas:

- ***Interconexão das Coisas:*** O sistema deve lidar com a interconexão dos dispositivos, os quais se referem a qualquer objeto físico relevante segundo a perspectiva do usuário ou da aplicação.
- ***Conexão das Coisas com a Internet:*** Os dispositivos devem estar conectadas a Internet.
- ***Coisas unicamente identificáveis:*** Os componentes do sistema devem ser identificáveis unicamente.
- ***Onipresença:*** O sistema deve estar disponível onde for necessário e sempre que for necessário.
- ***Capacidade de detecção/atuação:*** Existe o envolvimento de sensores e atuadores no sistema.
- ***Inteligência embutida:*** O sistema é composto por objetos dinâmicos e inteligentes que funcionam como ferramentas que estendem o corpo e a mente humana.
- ***Capacidade de comunicação interoperável:*** O sistema se comunica por meio de protocolos padrões e interoperáveis.
- ***Auto-configurável:*** Os dispositivos devem gerenciar a si próprios, em termos de configuração de software e hardware e utilização de recursos computacionais.
- ***Programabilidade:*** Os dispositivos podem ter comportamentos diferentes sem que sejam necessárias modificações físicas.
- ***Cenário de ambiente pequeno:*** Um sistema da Internet das Coisas de baixa complexidade pode ser definido como uma rede que conecta dispositivos unicamente identificáveis à Internet, os quais tem capacidade de detecção/atuação e programabilidade, de forma que o estado de um componente do sistema possa ser atualizado de qualquer lugar, a qualquer momento e por qualquer coisa.

- ***Cenário de ambiente grande:*** Um sistema da Internet das Coisas pode expandir de forma que um grande número de dispositivos estejam interconectados para entregar um serviço complexo e dar suporte a execução de um processo complexo.

Como visto nos atributos acima, a interconexão entre os dispositivos consiste em um aspecto importante no desenvolvimento de um sistema da IoT. Sendo assim, foram desenvolvidos diversos protocolos de comunicação, em diversas camadas de rede. Tratando-se da camada de aplicação, os principais protocolos são o MQTT, o *Extensible Messaging and Presence Protocol* (XMPP) e o *Advanced Message Queuing Protocol* (AMQP).

Uma questão que pode ser levantada é o motivo de não se utilizar protocolos de rede já existentes em projetos IoT, como o *Hypertext Transfer Protocol* (HTTP), o qual é bastante utilizado em serviços web. Para responder essa questão, serão analisadas algumas características do protocolo HTTP, as quais o limitam no contexto de projetos da Internet das Coisas (YUAN, 2017):

- O HTTP é um protocolo síncrono, no qual o cliente faz a requisição e espera a resposta do servidor. Em projetos IoT com muitos dispositivos conectados à rede, a qual pode apresentar falhas e altos tempos de resposta, a comunicação síncrona pode resultar em uma escalabilidade baixa do sistema.
- O HTTP é unidirecional, significando que somente o cliente inicia a conexão. Como em projetos IoT os dispositivos geralmente são clientes, não é possível que os mesmos recebam comandos sem antes terem realizado uma requisição.
- O HTTP é um protocolo de um para um, ou seja, a comunicação é realizada entre um cliente e um servidor, sendo caro e difícil enviar uma mensagem para vários dispositivos da rede.
- O protocolo HTTP possui muitos cabeçalhos e campos, tornando os pacotes grandes, o que não é adequado para redes limitadas.

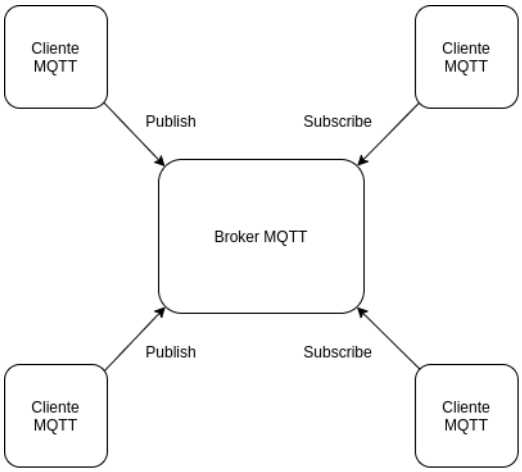
Sendo assim, é comum sistemas IoT utilizarem protocolos assíncronos, que permitam que os clientes recebam mensagens de forma passiva e que possibilitem enviar mensagens para vários dispositivos da rede de forma fácil. Entre os protocolos citados anteriormente, o AMQP é um dos mais utilizados para troca de mensagens em sistemas corporativos, nos quais a confiabilidade, segurança e outros recursos são os aspectos mais importantes, não havendo muita preocupação com a capacidade de computação e a latência da rede. Porém, em sistemas IoT nos quais os recursos computacionais são limitados, é mais comum utilizar protocolos mais leves como o MQTT, o qual será abordado com mais profundidade neste trabalho.

MQTT

O MQTT é um protocolo do tipo *Publish-Subscribe* destinado à troca de mensagens entre dispositivos, geralmente utilizando como base o protocolo *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP). Foi desenvolvido para ser usado em conexões remotas nas quais recursos como memória e largura de banda são limitados (YUAN, 2017), sendo composto por um *Broker* e por um número de

clientes, como na arquitetura ilustrada na Figura 2.1. O *Broker* é um servidor responsável por receber todas as mensagens enviadas pelos clientes e enviá-las para os clientes interessados. Um cliente pode ser qualquer tipo de dispositivo que se comunica com o *Broker*, enviando e recebendo mensagens.

Figura 2.1. Modelo *Publish-Subscribe*



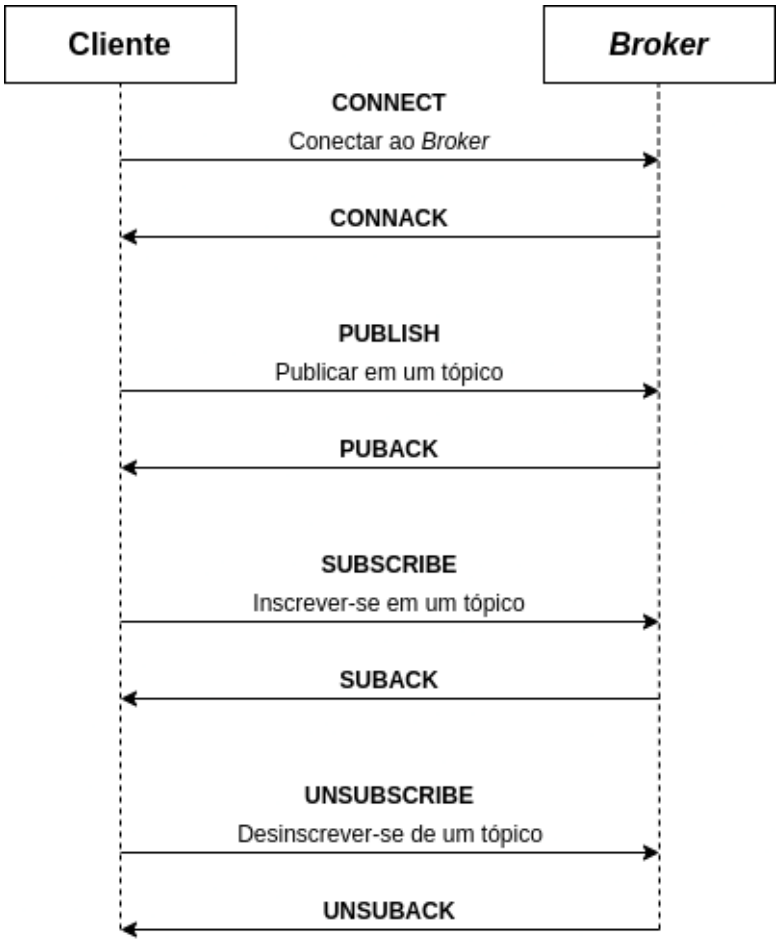
No modelo *Publish-Subscribe*, é comum os clientes receberem apenas parte de todas as mensagens publicadas. Sendo assim, no protocolo MQTT as mensagens são organizadas em tópicos, de forma que os clientes recebam apenas as mensagens publicadas nos tópicos em que tenham interesse. Um exemplo disso seria um sistema no qual vários sensores coletam informações e as publicam, utilizando um tópico diferente para cada sensor. Dessa forma, se um cliente deseja obter atualizações apenas de um sensor em específico, precisa apenas se inscrever no tópico em que o sensor publica as mensagens, não recebendo as mensagens dos demais sensores.

Um cliente pode atuar somente como um *Subscriber*, recebendo apenas as mensagens publicadas, como por exemplo em um circuito conectado à um display que exibe alguma informação. Também pode atuar somente como um *Publisher*, apenas publicando as mensagens, como por exemplo em um controle remoto. Além disso, pode atuar como ambos, o que pode ser exemplificado em um aplicativo de celular que recebe as mensagens e as exibe, além de controlar algum outro dispositivo publicando mensagens.

Cada mensagem contém um cabeçalho definindo o tipo da mensagem, como uma nova conexão ou uma inscrição em um tópico, além do conteúdo útil da mensagem, contendo os dados que desejam ser enviados. Os dados podem ser transferidos como um texto puro simples ou podem ser estruturados utilizando protocolos baseados em texto como o *JavaScript Object Notation* (JSON), desde que os clientes saibam processar corretamente os dados. As principais mensagens do protocolo MQTT, juntamente com as mensagens de confirmação correspondentes, se encontram na Figura 2.2.

Um recurso disponível no MQTT é o de mensagens retidas (*retained messages*). No modo de operação tradicional, caso um cliente não esteja conectado ao *Broker* em determinado momento, perderá todas as mensagens que forem enviadas enquanto estiver desconectado. Se ao enviar uma mensagem o cliente habilitar a *flag* de mensagens retidas, o *Broker* irá armazenar a última mensagem do tipo juntamente com o tópico relacionado. Assim, quando um cliente se conectar e inscrever

Figura 2.2. Principais mensagens do protocolo MQTT



em um tópico que contem uma mensagem retida, irá receber a mensagem imediatamente. Esse mecanismo é útil para que novos clientes recebam os valores atualizados armazenados nos tópicos sem que precisem esperar que alguém publique a mensagem.

Além disso, ao publicar uma mensagem é possível definir qual o nível de *Quality of Service* (QoS) será utilizado durante a conexão. Quanto maior o número, maior será o custo do envio.

- QoS 0 (No máximo uma vez): A mensagem é enviada e o *Broker* não toma nenhuma medida para garantir que a mesma foi entregue.
- QoS 1 (Pelo menos uma vez): A mensagem é enviada até que sua entrega seja confirmada.
- QoS 2 (Exatamente uma vez): A mensagem é enviada exatamente uma vez.

Os recursos de mensagens retidas e QoS foram apresentados pois serão utilizados no desenvolvimento do trabalho, mas existem diversos outros recursos disponíveis no MQTT, os quais podem ser consultados na documentação oficial[3].

A especificação do protocolo MQTT encontra-se atualmente em sua segunda versão, tendo sido a primeira (3.1.1) padronizada em 2014 (OASIS, 2014) e a segunda (5.0) padronizada em 2019 (OASIS, 2019). A última versão trouxe diversas melhorias em relação à versão anterior, dentre as quais estão: melhor comunicação de erros, balanceamento de carga nas inscrições, indicador do formato

[3] https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

do conteúdo da mensagem (*8-bit Unicode Transformation Format* (UTF-8) ou binário), expiração de mensagens e sessões, entre outros.

Existem diversos *Brokers* de mensagens que implementam o MQTT, cada um oferecendo suporte a diferentes linguagens de programação e versões do protocolo. Um dos mais conhecidos é o HiveMQ (HIVEMQ, 2020), desenvolvido em Java, estando disponível em uma versão paga e uma versão gratuita para a comunidade. Outra implementação bastante utilizada é o Eclipse Mosquitto (LIGHT, 2017), desenvolvido em C. Tanto o HiveMQ quanto o Mosquitto são compatíveis com as versões 3.1.1 e 5.0 da especificação do protocolo. Além das implementações convencionais, existem bibliotecas que permitem utilizar o MQTT como um módulo da aplicação. O Aedes (MOSCAJS, 2020) é uma biblioteca para o JavaScript que permite executar um *Broker* a partir de uma aplicação Node.js, sendo compatível com a versão 3.1.1 da especificação.

2.3 Aplicação IoT na nuvem

Os dispositivos conectados à Internet das Coisas geram dados, que geralmente precisam ser processados, armazenados e disponibilizados aos usuários por meio de uma interface, como um aplicativo ou um site. Nesse sentido, a computação em nuvem tem um papel importante no desenvolvimento de projetos IoT, permitindo que os mesmos sejam utilizados por usuários de qualquer lugar do mundo.

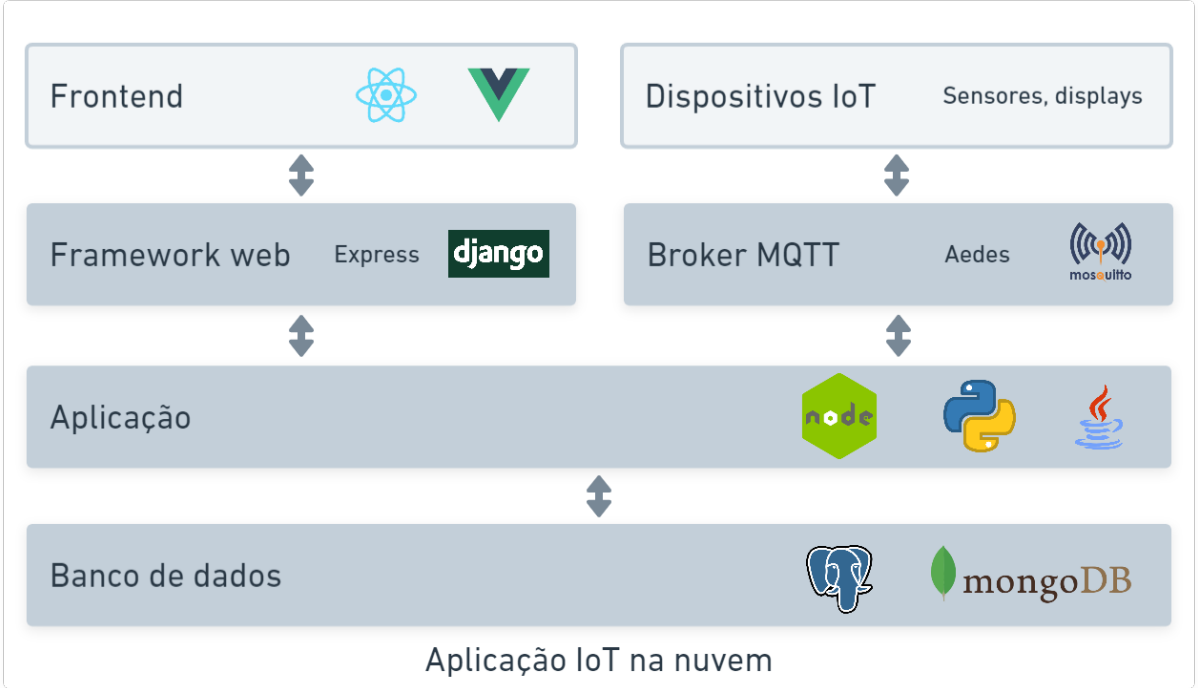
A Figura 2.3 ilustra uma arquitetura comumente utilizada em aplicações IoT na nuvem, sendo composta por uma aplicação que interage com o *frontend* e dispositivos IoT por meio de um *framework* web e um *Broker*, respectivamente, além de um banco de dados. A comunicação entre o *frontend* e o *framework* web geralmente é realizada utilizando o protocolo HTTP, que será abordado com mais detalhes na subseção Arquitetura Cliente-servidor. O *Broker* pode executar tanto como um serviço separado quanto como uma biblioteca integrada na aplicação, como visto na subseção MQTT da seção 2.2. O banco de dados pode ser um banco relacional, não relacional ou de outro tipo, explicados na subseção Banco de Dados.

Arquitetura Cliente-servidor

A Internet é baseada no modelo de computação distribuída Cliente-servidor, no qual existem duas funções que podem ser assumidas pelos processos: o cliente como o usuário de um serviço e o servidor como provedor de um serviço (PUDER et al., 2006), que se comunicam por meio de uma rede de computadores. No desenvolvimento web, é comum utilizar os termos *frontend* para os clientes e *backend* para os servidores.

O *frontend* consiste na parte visual de uma aplicação, com a qual o usuário interage diretamente. As tecnologias e linguagens mais utilizadas no desenvolvimento *frontend* web são o *Hyper Text Markup Language* (HTML), *Cascading Style Sheets* (CSS) e o JavaScript, as quais são executadas nas máquinas dos usuários pelos navegadores de Internet. O HTML é uma linguagem de marcação utilizada para definir a estrutura de uma página, sendo composta por diversas *tags* e

Figura 2.3. Arquitetura de uma aplicação IoT na nuvem



atributos, como botões, links e imagens. O CSS é uma linguagem de estilização utilizada juntamente com o HTML para definir o aspecto visual de uma página, como layout, cores e fontes. Por fim, o JavaScript é uma linguagem de programação destinada a criação de páginas interativas, permitindo adicionar funcionalidades mais complexas às páginas Web. Nos últimos anos, foram desenvolvidos diversos *frameworks* para o JavaScript que fornecem uma estrutura pré-definida para organizar o código fonte. Um dos mais utilizados atualmente é o React, um *framework* declarativo e baseado em componentes para a construção de interfaces (FACEBOOK, 2020).

Por outro lado, o *backend* é responsável pela parte funcional da aplicação, na qual as regras de negócio são implementadas e os dados são manipulados. Ao contrário do *frontend*, que executa no navegador, o *backend* executa em servidores especializados, podendo ser desenvolvido em diversas linguagens como Java, Python ou até mesmo JavaScript, por meio do Node.js[4], um ambiente de execução para código JavaScript fora do navegador. A comunicação com o *frontend* geralmente é realizada por meio do HTTP, um protocolo do tipo requisição-resposta, no qual o navegador inicia as requisições e aguarda as respostas dos servidores (MDN contributors, 2019). As requisições consistem nos seguintes elementos:

- Um método HTTP, indicando qual operação o cliente deseja realizar. A lista dos métodos mais utilizados e suas finalidades se encontram na Tabela 2.1.
- A *Uniform Resource Locator* (URL) do recurso a ser buscado.

[4] https://nodejs.org/

- Versão do protocolo.
- Cabeçalhos opcionais com informações adicionais.
- Corpo de dados, dependendo do método da requisição.

Tabela 2.1. Métodos HTTP mais utilizados

Método	**Significado**
GET	Solicitar um recurso específico
POST	Adicionar um recurso
PUT	Atualizar um recurso
DELETE	Remover um recurso
PATCH	Aplicar modificações parciais em um recurso

As respostas consistem nos seguintes elementos:

- Versão do protocolo.
- Código de status, indicando se a requisição foi corretamente concluída. O significado de cada faixa de código se encontra na Tabela 2.2.
- Mensagem de status, uma descrição simples do código de status.
- Cabeçalhos opcionais com informações adicionais.
- Corpo de dados, dependendo do método da requisição.

Tabela 2.2. Códigos de status de respostas HTTP

Faixa	**Significado**
100–199	Respostas informativas
200–299	Respostas de sucesso
300–399	Redirecionamentos
400–499	Erros do cliente
500–599	Erros do servidor

Bancos de Dados

Muitos sistemas precisam armazenar dados, como usuários cadastrados ou dados obtidos dos sensores. Para este fim, são utilizados os bancos de dados, coleções organizadas de informações, geralmente armazenadas e acessadas por um sistema de computador (ORACLE, 2020). Existem diversos tipos de banco de dados, os quais variam em como as informações são estruturadas. Nos bancos relacionais, os dados são armazenados em um conjunto de tabelas, contendo linhas e colunas, geralmente utilizando *Structured Query Language* (SQL) para a manipulação das informações. Outro tipo de bancos de dados consiste nos não relacionais, os quais permitem que os dados sejam armazenadas de forma não

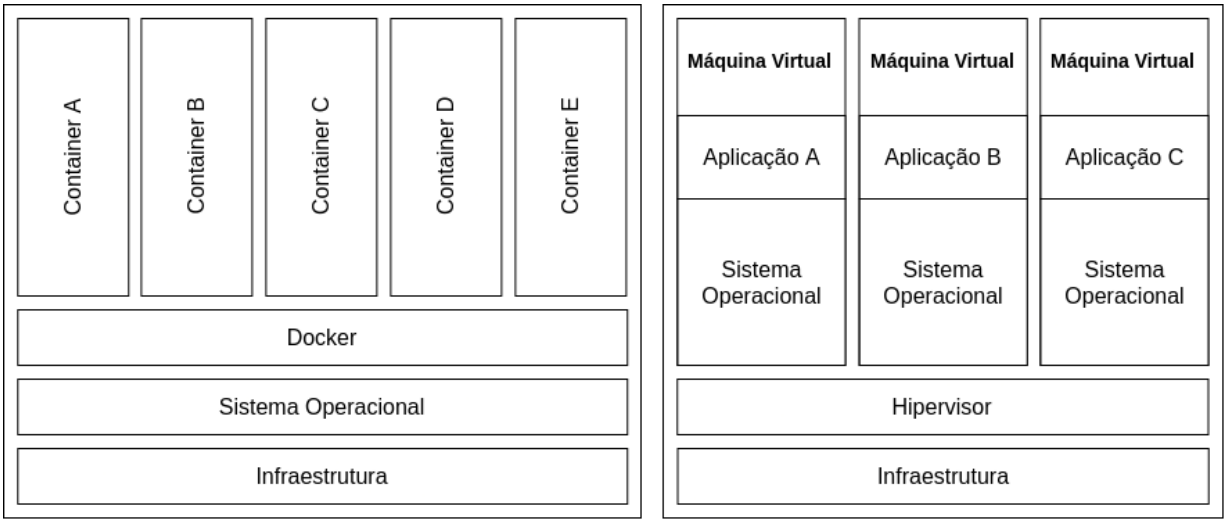
estruturada. Além disso, existem bancos que armazenam as informações em forma de grafos, bem como em documentos JSON.

Docker

O Docker é uma plataforma de código aberto utilizada para criar, implantar e gerenciar aplicações baseadas em *containers* (IBM Cloud Education, 2020). Um *container* é uma unidade de software contendo todo o código, ferramentas, bibliotecas e configurações necessárias para executar uma aplicação, sendo mais leve que uma máquina virtual. Isso permite empacotar os programas em *containers* que podem ser executados de forma segura em qualquer sistema que tenha suporte ao Docker, facilitando o processo de desenvolvimento e distribuição do software.

Os *containers* ocupam menos espaço na memória em relação às maquinas virtuais pois incluem somente os processos e dependências necessárias para executar o código, enquanto uma máquina virtual inclui uma cópia completa do sistema operacional. Essa característica possibilita executar vários *containers* na mesma máquina, além de ser necessário menos recursos computacionais. A Figura 2.4 ilustra a arquitetura do Docker à esquerda e a de máquinas virtuais à direita.

Figura 2.4. Arquitetura do Docker e de máquinas virtuais

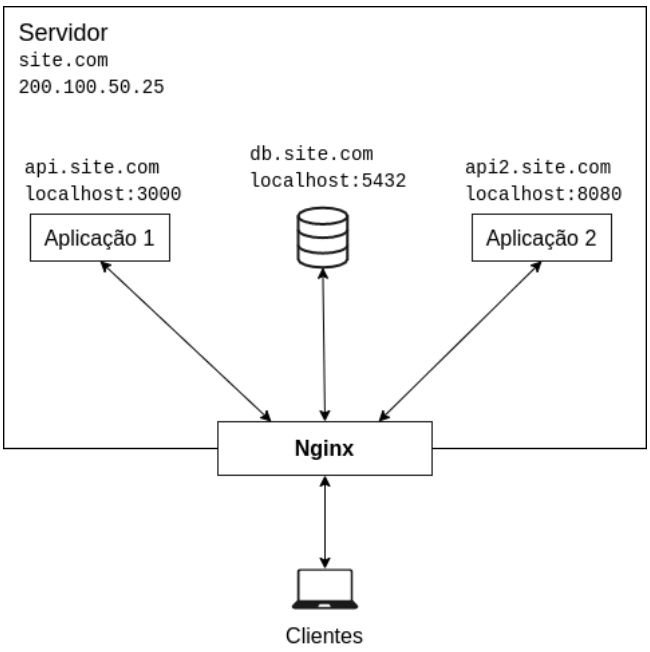


Antes de executar um *container* do Docker, é necessário criar uma imagem que contém todas as informações necessárias para executar a aplicação. A imagem não pode ser executada diretamente, funcionando como um *template* para a criação de novos *containers*. Geralmente é configurada a partir de instruções e comandos em um arquivo de texto chamado Dockerfile, que pode ser compartilhado e utilizado em qualquer ambiente com suporte ao Docker. Uma das maneiras de compartilhar essa configuração é por meio de repositórios de imagens, dentre os quais um dos mais utilizados é o Docker Hub, que contém mais de 100.000 imagens disponíveis atualmente.

Nginx

Como apresentado anteriormente, o Docker permite empacotar as aplicações em *containers* que executam de forma isolada. Sendo assim, é comum ter diversas aplicações executando em um único servidor, como um servidor HTTP, um Sistema de Gerenciamento de Bancos de Dados (SGBD) e um banco de dados em memória como o Redis. Dessa forma, é importante ter um meio para se comunicar com uma aplicação específica no servidor, sendo necessário analisar cada requisição para que a mesma seja processada pelo serviço correto. Uma das formas de permitir que várias aplicações sejam acessadas a partir do mesmo endereço IP é utilizando um *proxy* reverso, que recebe as requisições dos clientes, repassa para o processo destino e retorna a resposta para o cliente, como ilustrado na Figura 2.5. Um dos softwares mais utilizados para este fim é o Nginx, um servidor web de código aberto que além de atuar como *proxy* reverso também pode ser utilizado para cache, balanceamento de carga, *streaming*, além de outras funções.

Figura 2.5. *Proxy* reverso



Ainda na Figura 2.5, os domínios fictícios `api.site.com`, `db.site.com` e `api2.site.com` foram configurados para apontar para o endereço do servidor. Assim, por meio das configurações do Nginx, especifica-se que, quando chegar uma requisição destinada ao endereço `api.site.com`, a mesma deve ser redirecionada para o serviço executando na porta 3000 do servidor. Também é especificado que ao chegar uma requisição destinada ao endereço `db.site.com`, esta deve ser enviada ao processo executando na porta 5432, e assim por diante.

3 TRABALHOS RELACIONADOS

Este capítulo tem por objetivo analisar alguns projetos já existentes de placares eletrônicos semelhantes ao proposto neste trabalho, comparando as principais diferenças, funcionalidades e preço de cada solução. Na primeira seção serão abordados placares eletrônicos específicos para o tênis, enquanto na segunda seção serão vistas algumas iniciativas de placares eletrônicos para outras modalidades esportivas.

3.1 Placares eletrônicos no tênis

Não foram encontradas iniciativas de baixo custo de placares eletrônicos para o tênis durante a pesquisa. Além de uma busca geral na Internet, foram realizadas consultas nos sites Instructables[1] e Tinkercad[2], comunidades destinadas ao compartilhamento de projetos e circuitos dos mais variados tipos. Sendo assim, no decorrer dessa seção serão abordadas iniciativas comerciais de placares eletrônicos para o tênis.

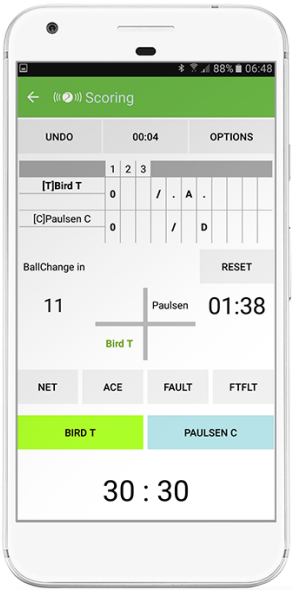
Foram encontradas poucas opções de placares comercializados no Brasil, entre os quais estão os produtos da empresa Plakr[3] e da empresa SportDesign[4]. O placar vendido pela Plakr possui uma lógica interna que calcula os pontos automaticamente de acordo com as regras do tênis, mostrando a pontuação do *game*, o número de *games* do *set* atual, a quantidade de *sets* ganhos por jogador e quem está sacando. As mesmas informações podem ser visualizadas no placar da empresa SportDesign, com a diferença de que a pontuação de cada *set* é mostrada de forma separada. Além disso, outros diferenciais são a exibição dos nomes dos jogadores e da duração atual da partida. Porém, diferentemente da solução desenvolvida neste trabalho, ambos os placares são controlados por meio de um controle remoto dedicado. Dessa forma, nesta seção serão exploradas com mais detalhes iniciativas nas quais o controle pode ser realizado por meio do celular.

No exterior é possível encontrar produtos mais avançados, como o SCORLI[5], apresentado na Figura 3.1. O placar é compatível com diversos esportes que utilizam raquetes, como o tênis, tênis de mesa e squash, podendo ser controlado por um relógio ou por um aplicativo no celular. A conexão entre os dispositivos é feita via *Bluetooth*, porém também é possível conectar um módulo *Wi-Fi* opcional ao placar, permitindo a exibição de imagens e vídeos, como anúncios de parceiros. Além disso, utilizando o aplicativo é possível visualizar as estatísticas gerais da partida ou de cada *set*, como número de pontos, *aces*, *winners*, duplas-faltas e erros não-forçados.

[1] https://www.instructables.com/
[2] https://www.tinkercad.com/
[3] https://www.plakr.com.br/plakr-p100/
[4] http://sportdesign.com.br/placar-t%C3%AAnis-de-campo
[5] http://scorli.com.hk/index.php

Figura 3.1. Placar, aplicativo e relógio do SCORLI

Figura 3.2. Placar e aplicativo Tennis Ticker



Outro exemplo de placar que pode ser encontrado fora do país é o Tennis Ticker[6], apresentado na Figura 3.2. Pode ser controlado tanto por uma pessoa de fora da quadra, por meio de um aplicativo no celular, como pelos próprios jogadores, utilizando um placar interativo ou um aplicativo para *smartwatchs*. Os dados coletados pelos dispositivos são enviados via *Wi-Fi* ou *Local Area Network* (LAN) para um servidor, permitindo que as estatísticas das partidas sejam acompanhadas em tempo real por outras pessoas a partir de uma interface Web ou por um aplicativo específico para este fim. Diferentemente das outras soluções citadas, nas quais a pontuação é realizada somente ponto a ponto, o Tennis Ticker também permite que a pontuação também seja feita *game* a *game*.

Para se ter uma ideia geral das diferenças entre as soluções descritas nesta seção, na Tabela 3.1 é apresentada uma síntese dos principais aspectos dos placares, comparando os seguintes pontos:

- **Tipo de controle (TC)**: Dispositivo e meio de comunicação utilizados no controle do placar.
- **Transmissão online (TO)**: Indica se os dados do placar são transmitidos pela Internet

[6] https://tennis-ticker.biz/live-scoring/App

- **Estratégia de gerenciamento de controle (EC)**: Método utilizado para gerenciar quem pode controlar o placar em determinado momento.
- **Características relevantes (CR)**: Diferenciais relevantes em relação aos demais placares.

Tabela 3.1. Comparativo dos placares

Nome	TC	TO	EC	CR
Plakr	Controle remoto (RF)	Não	Sem estratégia	-
SportDesign	Controle remoto (RF)	Não	Sem estratégia	-
SCORLI	App (Bluetooth)	Não	Sem estagégia, acesso ao placar após digitar uma senha no app	Visualização das estatísticas durante a partida
Tennis Ticker	App (Wi-Fi)	Sim	Não informado	App para *smartwatch* e armazenamento das estatísticas das partidas

Analisando com mais detalhes a coluna Estratégia de gerenciamento de controle (EC), é possível notar que nos placares controlados pelo celular não existe ou não é informado como é realizado o gerenciamento de quem pode manipular a pontuação em determinado momento do tempo, nem se é possível alternar quem está controlando o placar. Sendo assim, uma funcionalidade interessante e que não está presente nos placares seria a possibilidade de alternar de forma segura o celular com o qual é realizado o controle da pontuação no decorrer da partida, o que permitiria uma maior flexibilidade durante as competições.

Por fim, para que seja possível ter uma referência dos preços dos placares analisados nesta seção, são listados na Tabela 3.2 os valores atuais encontrados na Internet, sem realizar conversões entre as moedas. Não foi possível encontrar o preço do placar Tennis Ticker no site oficial ou em outras lojas.

Tabela 3.2. Preços dos placares

Nome	Preço
Plakr	R$ 6.875,40
SportDesign	R$ 14.800,00
SCORLI	€ 1,995.00
Tennis Ticker	Não informado

3.2 Placares eletrônicos em outros esportes

Com o objetivo de verificar se a mesma situação de existirem poucas opções de placares eletrônicos controlados pelo celular se repete em outros esportes além do tênis, foram realizadas buscas de soluções parecidas para modalidades esportivas considerados mais comuns, como o futebol e o vôlei, as quais serão apresentadas nesta seção.

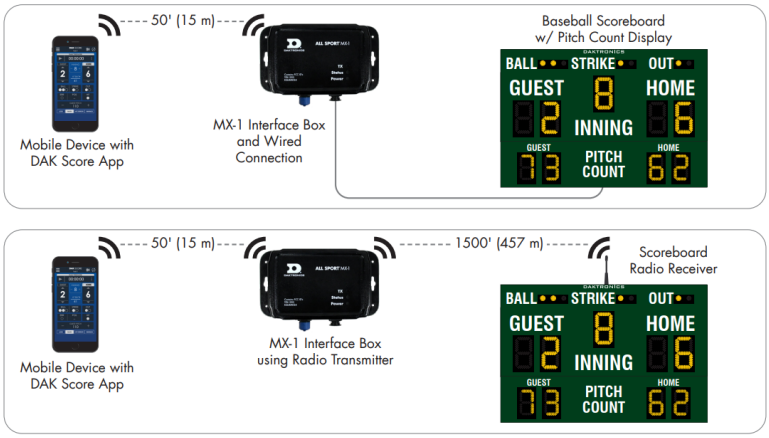
Tratando-se do futebol, foi possível encontrar um número maior de opções em relação ao tênis, todas sendo comercializadas por empresas de fora do país. Entre os produtos, se destacam os da empresa DakTronics[7], que possui um diverso catálogo de placares de diferentes tamanhos e especificações, controlados por um aplicativo, apresentado na Figura 3.4. Como ilustrado na Figura 3.5, o celular se comunica com uma interface de comando via *Bluetooth*, a qual pode ser conectada ao placar tanto por um cabo quanto por um transmissor de rádio.

Figura 3.4. Aplicativo DakTronics

Existem outras soluções semelhantes de placares eletrônicos para o futebol, as quais serão apenas citadas para fins de referência. Entre estas, se encontram as comercializadas pelas empresas LēDerbord[8], Major Display[9] e Score Vision[10].

[7] https://www.daktronics.com/en-us/products/sports/soccer
[8] https://www.lederbord.com/
[9] http://www.majordisplay.com/mobile-controls-scb3000/
[10] https://scorevision.com/scoreboards/outdoor/soccer/

Figura 3.5. Comunicação entre o aplicativo DakTronics e o placar



Para o vôlei, foram encontradas menos opções em relação ao futebol, sendo que estas também são comercializadas por empresas como a DakTronics e a Score Vision, as quais fornecem placares para diversas modalidades esportivas. Não foi encontrada uma opção de placar eletrônico controlado por aplicativo específico para o vôlei.

4 PLATAFORMA PARA GERENCIAR PLACARES DE TORNEIOS AMADORES DE TÊNIS

Este trabalho faz parte de uma solução tecnológica idealizada a partir do contato com academias que atuam no segmento da prática do tênis na cidade de Campo Mourão. A solução tecnológica se enquadra em uma estratégia muito utilizada para o engajamento de atletas amadores: a adoção de recursos do esporte profissional na prática amadora. Nessa perspectiva, a solução consiste no desenvolvimento de uma plataforma de software para gerenciar os placares de partidas de tênis em torneios amadores promovidos pelas academias aos seus associados. O projeto foi concebido para atender os requisitos do esporte amador e as restrições impostas pela realidade econômica e de infraestrutura das academias. Embora o placar eletrônico não esteja presente neste trabalho, ressalta-se que a plataforma desenvolvida provê suporte para um futuro acoplamento deste dispositivo. Na arquitetura desenvolvida, o placar eletrônico deve ser um dispositivo IoT e a comunicação deverá ocorrer por meio do protocolo MQTT. O protocolo de comunicação foi implementado e validado por meio de um protótipo eletrônico que simula o placar.

A Subseção 4.1 descreve as funcionalidades implementadas na plataforma. A Subseção 4.2 apresenta o protótipo do placar físico que já foi desenvolvido. Por fim, a Subseção 4.3 apresenta a metodologia utilizada no desenvolvimento deste trabalho.

4.1 Principais funcionalidades

A plataforma visa proporcionar uma experiência mais próxima à encontrada em âmbito profissional do tênis, trazendo recursos que não são comuns no dia a dia das academias. Destaca-se que as funcionalidades desenvolvidas na plataforma produzem benefícios para diferentes personagens envolvidos no torneio amador: a academia, o organizador do torneio, o atleta amador e o espectador.

Academia

Para a academia, o principal benefício proporcionado pela plataforma é o estímulo ao engajamento dos praticantes ao proporcionar uma solução tecnológica similar à de torneios profissionais.

A plataforma também disponibiliza um recurso para o acompanhamento da pontuação em tempo real pela Internet, permitindo que pessoas que não estejam assistindo ao jogo presencialmente possam acompanhar as partidas. As pontuações dos jogos em andamento podem ser visualizadas em uma interface web disponibilizada em um domínio específico para a academia. Com esse recurso, a academia pode utilizar a transmissão da pontuação em tempo real como forma de divulgação do torneio, da academia e do esporte. Uma interface web específica para visualização em telas maiores, tal como uma *smart TV*, foi desenvolvida para que a academia possa transmitir a pontuação das partidas em andamento em ambientes de convivência. Essa interface conta com elementos mais

destacados e fontes maiores, para que seja possível visualizar as informações a partir de uma maior distância.

Organizador do torneio

Uma das principais funcionalidades da plataforma é o controle do placar dos jogos por meio de uma interface web simples. Caso o placar eletrônico seja acoplado à plataforma, a interface web também controlará o dispositivo. Considerando que grande parte das academias não contam com uma equipe específica para organização dos torneios e nem recursos financeiros para contratação, a plataforma permite que o coordenador transfira o controle do placar para outra pessoa a qualquer instante. A transferência é realizada por meio da geração de um *link* com um *token* de autenticação embutido. Ao acessar o *link* enviado pelo coordenador, o usuário acessa uma interface autenticada e assume o controle do placar. Essa estratégia foi definida para suprir a ausência de uma equipe fixa de organização, para minimizar a burocracia do gerenciamento de usuários autenticados e, ainda assim, garantir um nível razoável de segurança no controle do placar.

A maioria dos torneios amadores contam com o recrutamento de voluntários transitórios para auxiliar o coordenador. Dessa forma, em virtude da facilidade da transferência do controle via *link* autenticado, o coordenador pode recrutar voluntários durante o andamento das partidas. O coordenador pode, a qualquer momento, retomar o controle do placar e/ou transferi-lo para outra pessoa. O sistema garante que somente uma única pessoa manipule o placar ao mesmo tempo.

Portanto, o sistema disponibiliza uma interface para o organizador criar as partidas e realizar a transferência do controle do placar e outra interface para o voluntário que gerenciará os pontos da partida. A interface do voluntário foi concebida de forma que o usuário não necessite conhecer as regras de pontuação do tênis. O voluntário somente precisa indicar quem fez o ponto. Toda a lógica da pontuação foi implementada na interface.

Por fim, considerando que os torneios são destinados ao esporte amador, o sistema permite que, na criação da partida, o coordenador possa realizar algumas customizações de regras da prática profissional, que são comuns na prática amadora. Por exemplo, habilitar ou desabilitar a regra da vantagem e definir se o desempate será um *Set* regular ou um super *Tie-Break*.

Atleta amador e Espectador

Com o controle dos pontos sendo realizado por um voluntário, os atletas não terão mais a responsabilidade gerenciar os pontos da partida. A prática de controlar os próprios pontos traz confusões durante a partida, principalmente, para os atletas iniciantes. Outro benefício proporcionado pela plataforma é a coleta e armazenamento dos dados dos jogos e atletas. Embora a plataforma não ofereça nenhuma interface para visualização de estatística, ressalta-se que os dados estão sendo coletados e armazenados para ser explorados em trabalhos futuros.

O recurso de acompanhamento da pontuação em tempo real das partidas, já mencionado anteriormente, também caracteriza-se como benefício para o atleta amador e, obviamente, para

o espectador. O atleta pode explorar esse recurso para divulgar seu jogo em suas redes sociais e o espectador pode acompanhar os detalhes da partida mesmo que não possa estar presente na arquibancada. Além da pontuação, por meio de uma linha do tempo visual, são exibidos os principais eventos, como qual jogador conquistou o ponto, qual o tipo do ponto, quando terminou o *game*, quando terminou o *set* e quando terminou o jogo. A partir da coleta de dados, em um trabalho futuro, também será possível visualizar algumas estatísticas dos jogadores durante e após as partidas, como porcentagem de pontos ganhos, porcentagem de *aces*, entre outras.

Por fim, como última funcionalidade a se destacar, a plataforma foi desenvolvida sob uma arquitetura para disponibilizar sua utilização na forma de um serviço. Com isso, foi criada uma área administrativa para gerenciar a criação das academias. De forma resumida, as principais funcionalidades disponibilizadas pela plataforma são:

- Controle do placar eletrônico por meio de uma aplicação web;
- Transferência do controle do placar para outras pessoas por meio do compartilhamento de *link* autenticado;
- Transmissão da pontuação em tempo real;
- Transmissão da pontuação em tempo real para grandes telas;
- Armazenamento de estatísticas;
- Disponibilização do sistema como um serviço;

A seguir, serão apresentadas algumas características do protótipo eletrônico utilizado para validar a comunicação entre a plataforma e um possível placar eletrônico.

4.2 Protótipo eletrônico

O protótipo eletrônico foi construído por um aluno do curso de Engenharia Eletrônica com intuito de viabilizar o desenvolvimento da camada do protocolo de comunicação com um placar eletrônico que será incorporado como trabalho futuro.

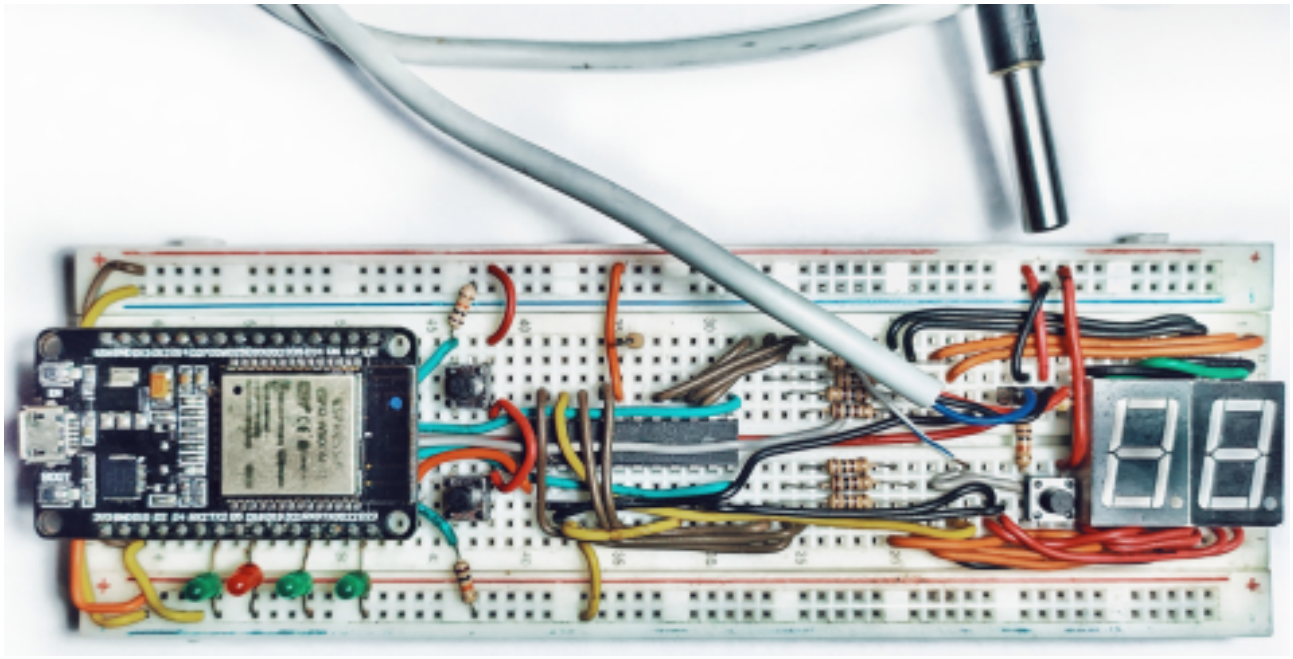
No protótipo foi considerado um placar com um display para a pontuação atual do *game* para cada jogador, além de um display para 3 *sets*. A decisão de mostrar essa quantidade de *sets* foi embasada em uma característica comum nos torneios amadores, os jogos dificilmente possuem mais do que 3 *sets* devido ao condicionamento físico exigido. Além disso, ao invés de mostrar os nomes dos jogadores em um *display*, os mesmos são identificados de forma fixa como "Jogador 1" e "Jogador 2". O protótipo foi construído com base no conceito de dispositivos IoT, tendo em vista a interconexão com a Internet e com outros sistemas, por meio da qual os dados são enviados e recebidos.

O protótipo do placar foi desenvolvido com o microcontrolador ESP32-WROOM-32D da Espressif® de 32-bits, que contém periféricos de comunicação *Wi-Fi* e *Bluetooth*. Para controlar os displays foi utilizado o circuito integrado MAX7219 da Sparkfun®, controlado por meio de um barramento *Serial Peripheral Interface* (SPI). Além disso, o protótipo possui um sensor DS18B20 da Maxim Integrated®, responsável por fornecer os dados da temperatura ambiente.

Foram adicionados quatro *LEDs* para indicar o funcionamento do dispositivo. O primeiro indica se o sistema de arquivos foi iniciado com sucesso, o segundo indica se o protótipo está tentando se conectar a rede, o terceiro indica se a conexão foi bem sucedida e o quarto indica se o placar está conectado com o *Broker*.

Por fim, a Figura 4.1 mostra uma foto de uma visão de cima do protótipo. Por falta de espaço na *protoboard* e disponibilidade de alguns componentes, a imagem mostra apenas um circuito de controle e dois *displays*. Porém, o projeto contem três circuitos de controle, cada um suportando até oito *displays*, não sendo necessário modificar o código ou os demais componentes para a inclusão dos mesmos.

Figura 4.1. Protótipo do placar eletrônico



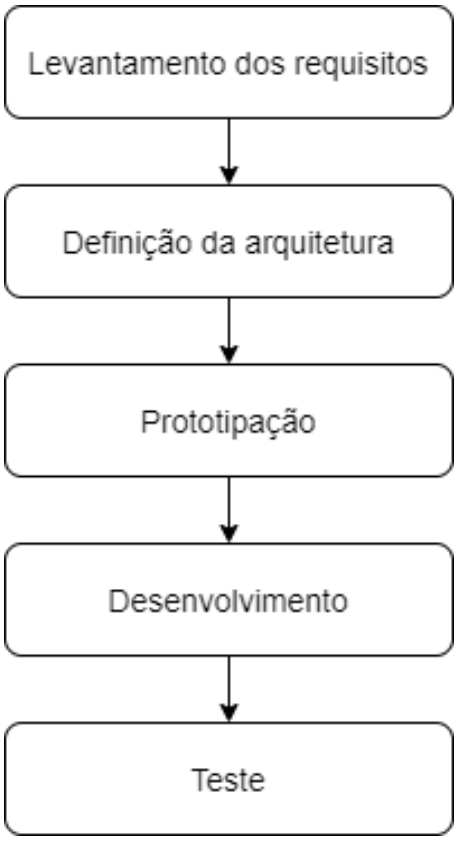
4.3 Metodologia

O desenvolvimento deste trabalho foi dividido em 5 etapas, as quais estão ilustradas na Figura 4.2.

O desenvolvimento do trabalho iniciou-se com a etapa de **Levantamento dos requisitos**, na qual diferentes personagens envolvidos na prática do tênis foram ouvidos para entender a perspectiva de cada um.

Em seguida, na etapa de **Definição da arquitetura**, com as informações levantadas, planejou-se a arquitetura adequada para atender as funcionalidades identificadas. Em resumo, a arquitetura planejada está dividida em duas grandes partes: uma com os aspectos relacionados ao hardware e outra com aspectos relacionados ao software. Ressalta-se que este trabalho contempla somente os aspectos relacionados ao software.

Figura 4.2. Etapas do desenvolvimento



A partir das funcionalidades identificadas e da arquitetura projetada, na etapa de **Prototipação**, foram desenhados os protótipos das principais telas que compõem o sistema, as quais serviram como base durante o desenvolvimento do trabalho.

Com base nos protótipos desenhados, na etapa de **Desenvolvimento** foram implementadas todas as funcionalidades descritas.

Por fim, após desenvolvido o sistema, na etapa de **Teste** seriam realizados experimentos com os usuários, a fim de saber em como a solução foi benéfica durante as partidas, bem como os pontos que poderiam ser melhorados. Porém, devido as medidas restritivas decorrente à pandemia durante o desenvolvimento do trabalho, essa etapa não pode ser realizada. Para validar se os casos de uso da aplicação estavam funcionando conforme a responsabilidade definida, foram implementados testes unitários para as classes mais importantes.

4.3.1 Levantamento dos requisitos

Para o levantamento de requisitos foram realizadas reuniões com um proprietário de academia, um professor de tênis e jogadores amadores.

As entrevistas com o proprietário de academia e o professor de tênis resultaram na identificação das seguintes funcionalidades:

- **Disponibilização do sistema como um serviço**: essa funcionalidade foi sugerida devido ao fato das academias não possuírem uma infraestrutura compatível com um sistema deste porte.

Com essa funcionalidade, o único requisito para a academia seria uma conexão com a internet e sua distribuição;

- **Controle do placar eletrônico por meio de uma aplicação web**: essa é a funcionalidade central da solução. Segundo os entrevistados, esse recurso contribuirá para o engajamento e participação dos associados nos torneios organizados.
- **Transferência do controle do placar para outras pessoas**: essa funcionalidade foi proposta com intuito de auxiliar na organização dos torneios. Os entrevistados relataram que o objetivo principal dos torneios é o incentivo dos tenistas à prática do esporte. Dessa forma, os torneios não dispõem de recursos financeiros para contratação de colaboradores. Com isso, os organizadores contam com o auxílio de voluntários na organização. Portanto, considerando essa particularidade, deveria haver uma estratégia descomplicada para transmissão do controle do placar entre os voluntários. A solução encontrada foi o controle por meio de *tokens*. O *token* é gerado pelo coordenador da academia e compartilhado pelas redes sociais por meio de um *link*. Ao acessar o *link*, o voluntário acessa a interface de controle do placar eletrônico.

As entrevistas com os jogadores de tênis resultaram na identificação das seguintes funcionalidades:

- **Transmissão da pontuação em tempo real**: essa funcionalidade foi identificada de acordo com uma prática comum entre os jogadores de tênis. Nos jogos mais importantes dos torneios, os espectadores ficam alimentando as redes sociais com informações sobre o placar do jogo.
- **Armazenamento de estatísticas**: essa funcionalidade foi identificada devido ao desejo dos jogadores de se ter um relatório com estatísticas relacionadas ao seu desempenho. Como essa funcionalidade necessitaria de uma carga de trabalho adicional para manter um controle autenticado dos tenistas, optou-se por somente manter uma base de dados com as informações das partidas, para que, posteriormente, essa funcionalidade seja implementada.

4.3.2 Definição da arquitetura

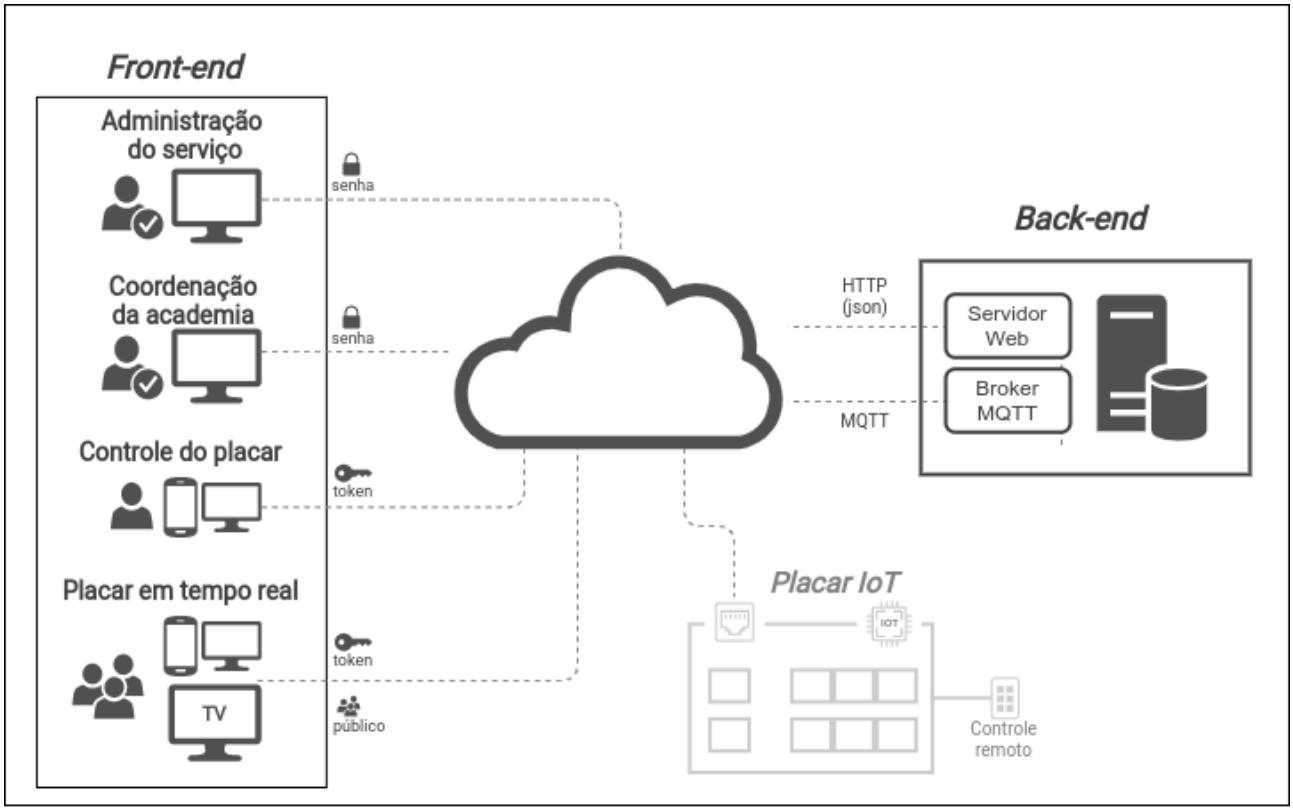
A partir das funcionalidades descritas, é possível identificar dois principais aspectos a serem levados em consideração na definição da arquitetura do projeto:

- Comunicação entre a aplicação web e o protótipo eletrônico simulando o placar físico para o controle e acompanhamento da pontuação.
- Processamento, armazenamento e disponibilização dos dados das academias, placares, partidas e jogadores.

Para a comunicação entre a aplicação web e o protótipo foi utilizado o MQTT, um protocolo do tipo *Publish-Subscribe* destinado a troca de mensagens entre dispositivos de forma assíncrona, sendo um dos mais utilizados em projetos IoT, como visto na fundamentação teórica. No aspecto do processamento dos dados gerados pelo sistema, foi utilizada a arquitetura cliente-servidor, a mais

comum em sistema web. Dessa forma, na Figura 4.3 é possível ter uma visão geral dos componentes do sistema.

Figura 4.3. Visão geral dos componentes do sistema



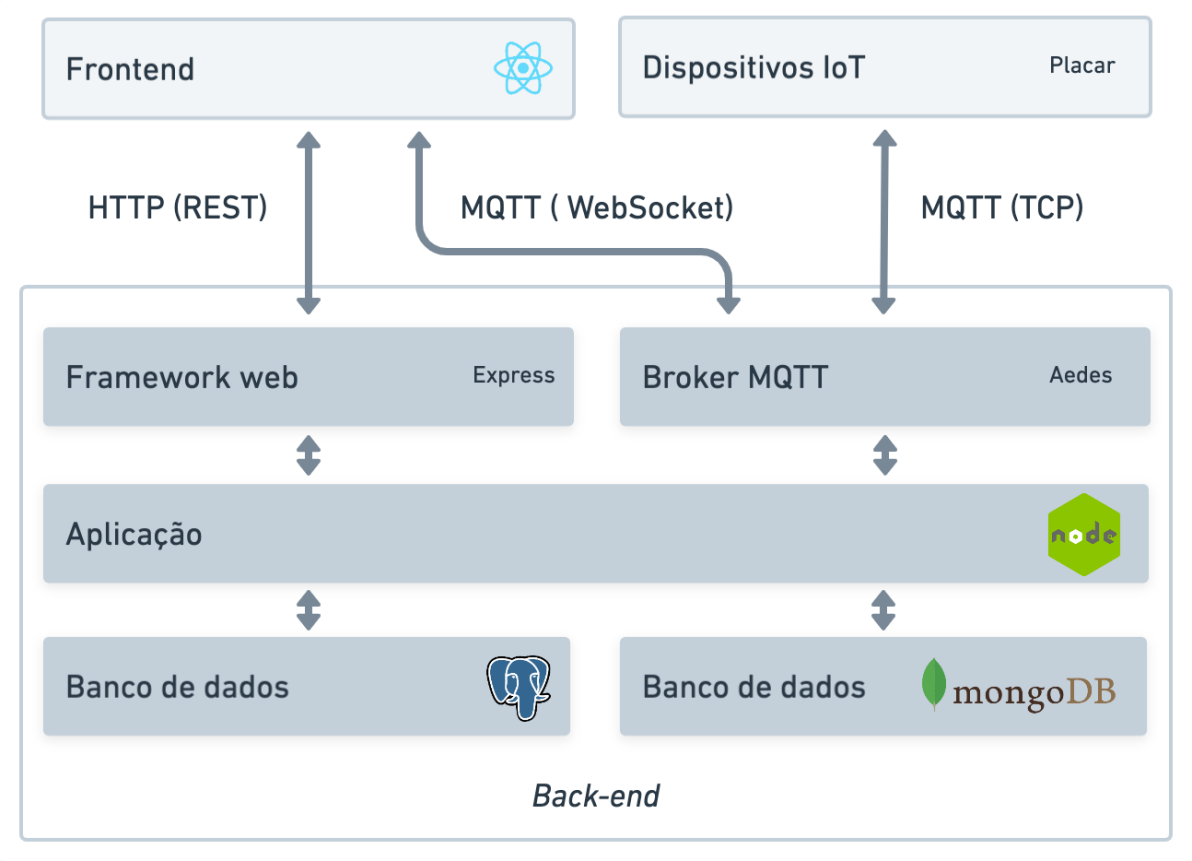
Com base nessas informações, para atender aos requisitos do projeto foi necessário pensar em uma arquitetura na qual fosse possível unir os conceitos de Internet das Coisas e de sistemas web na nuvem, de forma que o *Broker* MQTT trabalhasse em conjunto com o servidor de aplicação. Sendo assim, a arquitetura utilizada para o desenvolvimento do trabalho é comumente utilizada em aplicações IoT na nuvem (ilustrada na Figura 2.3 da Seção 2.3.

A Figura 4.4 é uma adaptação da imagem utilizada para ilustrar a arquitetura, porém acrescentando mais informações, tais como quais tecnologias foram utilizadas em cada camada, além dos principais protocolos de comunicação. No decorrer desta subseção, cada parte relacionada ao software da arquitetura será abordada com mais detalhes, iniciando pelo *backend* e finalizando pelo *frontend* da aplicação.

Backend

Como é possível visualizar na Figura 4.4, o *backend* é composto por uma aplicação, a qual interage com o *frontend* e dispositivos IoT por meio de um *framework* web e um *Broker* MQTT, respectivamente, além dos bancos de dados. O protocolo utilizado na comunicação entre o *frontend* e o *backend* foi o HTTP, seguindo o padrão *Representational State Transfer* (REST). Além disso, o *frontend* também se comunica com o *Broker* MQTT, porém utilizando o protocolo MQTT sobre *WebSockets*.

Figura 4.4. Arquitetura



A aplicação foi desenvolvida utilizando o Node.js, juntamente com o Express[1] como *framework* web, o Knex[2] para a construção de consultas SQL e o Mongoose[3] para a comunicação com o MongoDB. Como *Broker*, foi utilizado o Aedes[4], uma biblioteca que permite executar um *Broker* MQTT a partir do Node.js. A decisão de executar o *Broker* como um módulo da aplicação e não como um serviço separado surgiu da necessidade de se ter um controle mais fino de quais mensagens podem ser publicadas, recurso utilizado na estratégia do gerenciamento de controle do placar físico, o qual será abordado a seguir.

Entre as funcionalidades do sistema, foi apresentado a possibilidade de transferir o controle da pontuação para outra pessoa por meio de um link compartilhado. Para tornar esse processo menos burocrático, não é necessário ter um cadastro das pessoas que podem controlar o placar. Porém, isso impõe a necessidade de pensar em uma estratégia alternativa para garantir a segurança desse sistema, como não permitir que mais de uma pessoa possa manipular a pontuação ao mesmo tempo. Sendo assim, foi pensado um sistema baseado em *tokens*, no qual cada placar tem um *token* de publicação, chamado de ***publish_token*** e um *token* de *refresh*, chamado de ***refresh_token***, os quais são gerados

[1] https://expressjs.com/
[2] http://knexjs.org/
[3] https://mongoosejs.com/
[4] https://github.com/moscajs/aedes

aleatoriamente e salvos no banco de dados no momento da criação da partida. Dessa forma, ao publicar no *Broker*, deverá ser enviado juntamente com a carga útil da mensagem o *publish_token*, que será verificado quando a mensagem for recebida pela aplicação. Se o *token* enviado for diferente do salvo no banco de dados, a mensagem será descartada e não será recebida pelos demais *subscribers*, como a interface de acompanhamento da partida e o placar eletrônico.

Com base nisso, quando for necessário alternar quem está controlando o placar, será enviado para a pessoa um link contendo o *refresh_token*, que ao ser acessado irá requisitar uma rota da *Application Programming Interface* (API) responsável por gerar novos *tokens* aleatórios para o placar. Dessa forma, quem possuía os *tokens* antigos não terá mais permissão para publicar no *Broker*, nem para transferir o controle para outra pessoa, pois os *tokens* estarão diferentes dos salvos no banco de dados. Para transferir o controle está sendo utilizado um *token* diferente do utilizado para a verificação do controle da pontuação, pois se fosse utilizado somente o *publish_token* para ambas as tarefas, ao receber o link de controle, duas pessoas teriam acesso ao *token* de publicação, o que permitiria que ambas controlassem o placar.

Uma outra funcionalidade do sistema é a transmissão da pontuação em tempo real, que permite que as pessoas que não possam estar presencialmente na quadra possam assistir aos jogos de qualquer lugar. Porém, para atender os jogadores que não querem que suas informações fiquem públicas, foi implementado um sistema de proteção por PIN, que pode ser definido no momento da criação da partida. Como não é possível enviar dados adicionais no momento da inscrição em um tópico, a abordagem escolhida para a implementação dessa funcionalidade foi tornar os tópicos do *Broker* aleatórios a cada partida. Sendo assim, o tópico é apenas retornado para o usuário após validar se o PIN está correto, o que garante que somente os usuários autorizados possam se inscrever no *Broker*. Porém, essa abordagem impõe dois novos desafios:

- Uso de *wildcards*: é possível se inscrever em um tópico do *Broker* utilizando *wildcards*. Por exemplo, se inscrever no tópico # pode ser utilizado para recebar as mensagens enviadas para todos os tópicos do *Broker*. Essa possibilidade permitiria burlar a proteção do PIN, pois mesmo que um usuário não conhecesse o tópico da partida, ainda sim conseguiria se inscrever. Para contornar isso, o servidor intercepta a tentativa de inscrição em um tópico. Se o tópico no qual o usuário deseja inscrever-se contém algum *wildcard* indevido, o mesmo é impedido de se inscrever. Isso força o cliente a conhecer o tópico completo para conseguir se inscrever no *Broker*.
- Placar físico: atualmente, o protótipo do placar físico não tem um mecanismo para conhecer esse tópico aleatório da partida, apenas se comunicando por meio de um tópico fixo, que é gerado a partir de informações do hardware. Dessa forma, é necessário que seja estabelecida uma forma de comunicação entre o tópico aleatório utilizado pela partida e o tópico fixo gravado no placar. Sendo assim, quando uma mensagem é publicada no tópico do placar, o servidor à redireciona para o tópico aleatório da partida. Da mesma forma, quando uma mensagem é publicada no tópico da partida, o servidor à redireciona para o tópico do placar.

Finalizando a explicação do *backend*, os dados gerados pela aplicação são armazenados utilizando o PostgreSQL[5], em conjunto com o MongoDB[6] para o armazenamento dos dados gerados durante as partidas. A escolha de se utilizar um banco de dados não relacional para estes dados se deve ao fato de que os mesmos não necessitam do nível de integridade que um banco de dados relacional proporciona, além de consistirem em registros que não precisam seguir uma estrutura pré-definida. Sendo assim, a Figura 4.5 apresenta o diagrame entidade relacionamento do projeto, enquanto a Figura 4.6 apresenta os esquemas que são armazenados no MongoDB.

Frontend

O *frontend* da aplicação foi desenvolvido utilizando o React[7], sendo composto por quatro partes bem definidas, como ilustrado na Figura 4.3, as quais serão explicadas no decorrer desta subseção.

Como descrito nas funcionalidades, um dos recursos implementados foi o uso do sistema como um serviço. Sendo assim, na parte da **Administração do serviço** foram desenvolvidas interfaces destinadas ao gerenciamento das academias cadastradas, inclusão de novos placares nas academias e gerenciamento de coordenadores das academias. Por ser uma área administrativa, o acesso é realizado por meio de um usuário e senha.

O gerenciamento das partidas, a inclusão de novos coordenadores e o gerenciamento dos jogadores é responsabilidade da **Coordenação da academia**, uma área que é acessível a partir de um subdomínio específico para cada academia, que é criada pela Administração do serviço no momento do cadastro. O acesso também é realizado por meio de um usuário e senha.

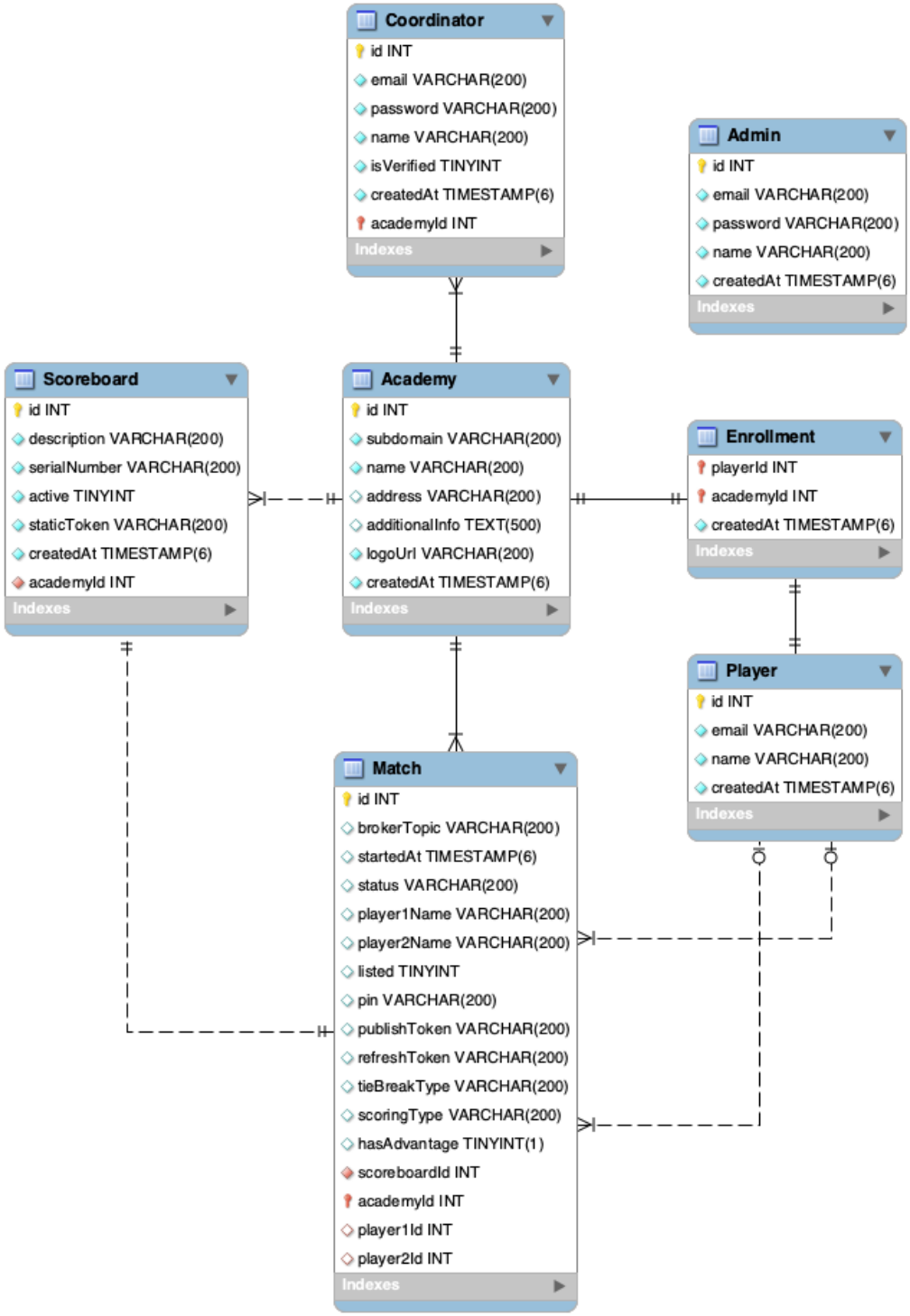
O controle da pontuação e envio dos links de acesso é realizado pelo **Controle do placar**. Para facilitar o trabalho de quem será responsável pela partida, a interface de controle do placar possui dois botões, um para cada jogador. Ao pressionar os botões, o sistema calcula automaticamente a pontuação com base nas regras escolhidas no momento da criação da partida, informando qual jogador deve sacar, bem como quando finalizou um *game*, *set* ou o jogo. A autenticação é realizada utilizando o sistema de *tokens* explicado anteriormente.

A partir do **Placar em tempo real**, pessoas que não possam acompanhar presencialmente as partidas podem visualizar a pontuação em tempo real, acessando um link enviado por quem está controlando o placar. Caso os jogadores prefiram, é possível definir um PIN de 4 dígitos para proteger o acesso aos dados da partida.

Por fim, a partir do **Placar em tempo real**, também é possível acompanhar a pontuação em tempo real da mesma forma que na interface descrita acima, mas de forma mais adaptada para a visualização em TVs e monitores presentes na academia.

[5] https://www.postgresql.org/
[6] https://www.mongodb.com/
[7] https://reactjs.org/

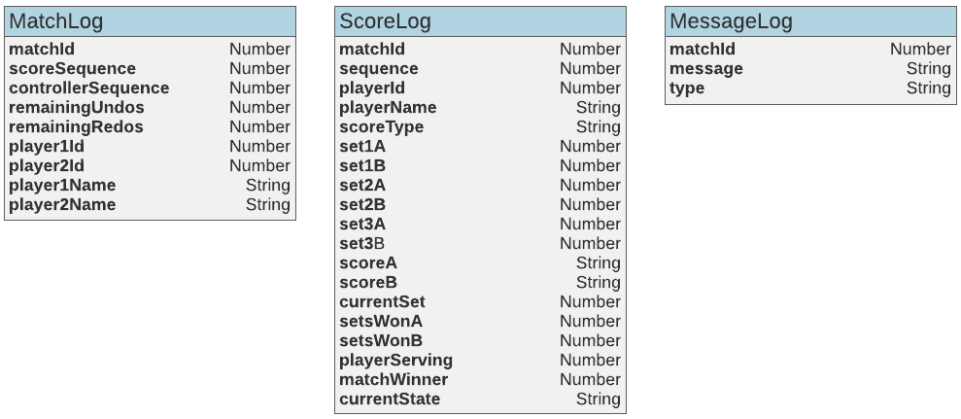
Figura 4.5. Diagrama entidade relacionamento



4.3.3 Prototipação

Nesta subseção serão apresentados os protótipos das principais telas da aplicação web, ilustradas na Figura 4.7. A parte **A** mostra a tela com a lista dos placares, por meio da qual é possível acompanhar as pontuações em tempo real, bem como saber quais placares estão com partidas em andamento e quais estão disponíveis.

Figura 4.6. Esquemas do MongoDB



Ao clicar em um placar disponível, o usuário é redirecionado para a tela ilustrada na parte **B**, por meio da qual será realizada a configuração da partida, como o nome dos jogadores, tipo do *tie-break* e tipo da pontuação.

Após iniciar a partida ou ao clicar em um placar com um jogo em andamento, é mostrado ao usuário a tela principal do sistema, na qual estão disponíveis os controles de pontuação, juntamente com uma linha do tempo visual com os últimos acontecimentos, como pode ser visto na parte **C**.

A partir da tela principal é possível acessar uma tela de ações, ilustrada na parte **D**, por meio da qual o usuário pode finalizar a partida e desfazer ou refazer as últimas jogadas.

Também é possível acessar uma tela contendo o link para transferir o controle do placar e um link para acompanhar a partida em tempo real, como ilustrado na parte **E**.

Por fim, ao acessar o link para acompanhar a partida em tempo real, é apresentado ao usuário uma tela parecida com a de controle do placar, porém sem os botões de pontuação, como é possível visualizar na parte **F**.

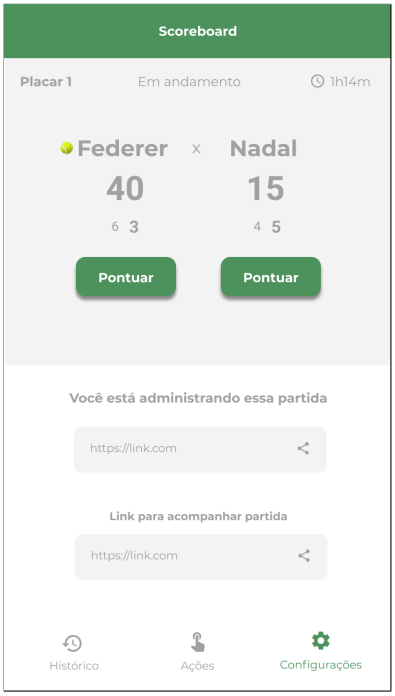
Figura 4.7. Protótipos

(A) Lista de placares

(B) Configurações da partida

(c) Tela principal

(D) Ações da partida



(E) Links da partida

(F) Acompanhar partida

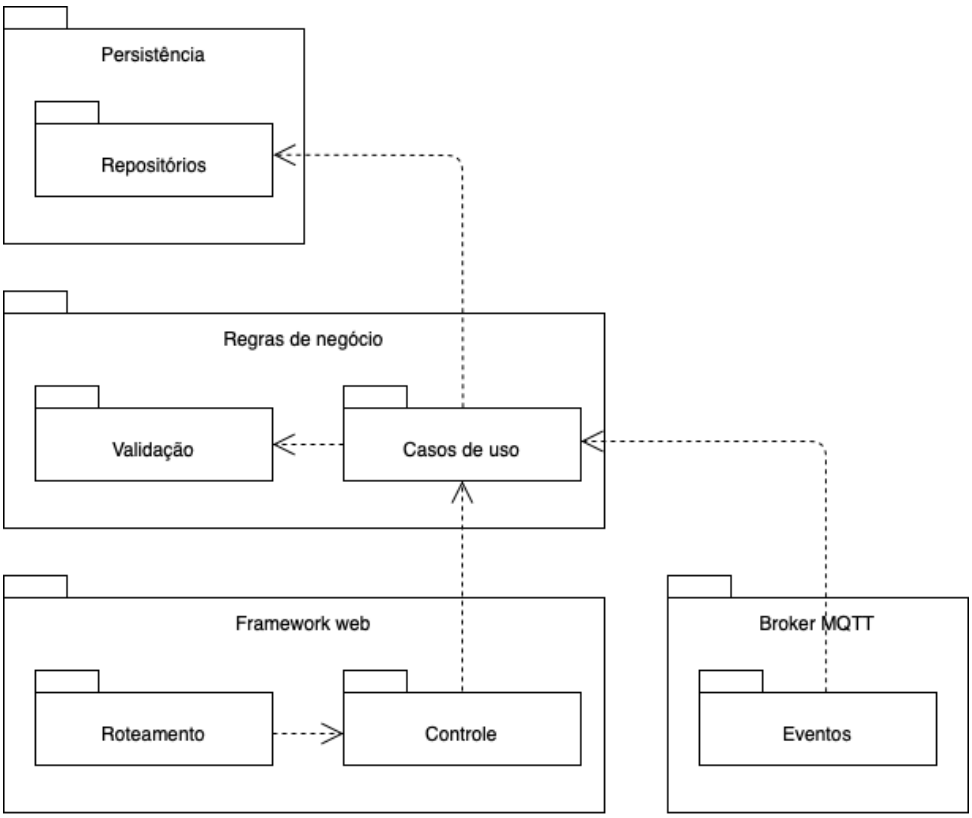
4.3.4 Desenvolvimento

A partir da arquitetura definida na Subseção 4.3.2, foram desenvolvidos todos os módulos descritos no Levantamento dos Requisitos. Sendo assim, esta subseção irá explorar com mais detalhes a arquitetura utilizada, com foco nos aspectos relacionados à organização do código-fonte do projeto. A primeira parte explora o desenvolvimento do *backend*, seguido pela segunda parte, que explora o desenvolvimento do *frontend*.

Backend

O código-fonte do *backend* foi separado em camadas com finalidades bem definidas, ilustradas na Figura 4.8. Foram utilizados conceitos propostos pela Arquitetura Limpa (MARTIN; COPLIEN, 2009), os quais visam reduzir o acoplamento do código e a dependência de detalhes de implementação, como o banco de dados. Entre os benefícios obtidos com a adoção desta arquitetura estão um código mais organizado, testável e manutenível.

Figura 4.8. Camadas do *backend*



Sendo assim, as camadas estão representadas pelos pacotes, sendo estas a **Camada de roteamento**, **Camada de controle**, **Camada de eventos**, **Camada de casos de uso**, **Camada de validação** e **Camada de repositórios**. Algumas camadas podem ainda ser agrupadas em conceitos mais amplos, delimitados pelos pacotes mais externos. Por exemplo, a parte relacionada ao *framework* Web engloba as camadas de roteamento e de controle.

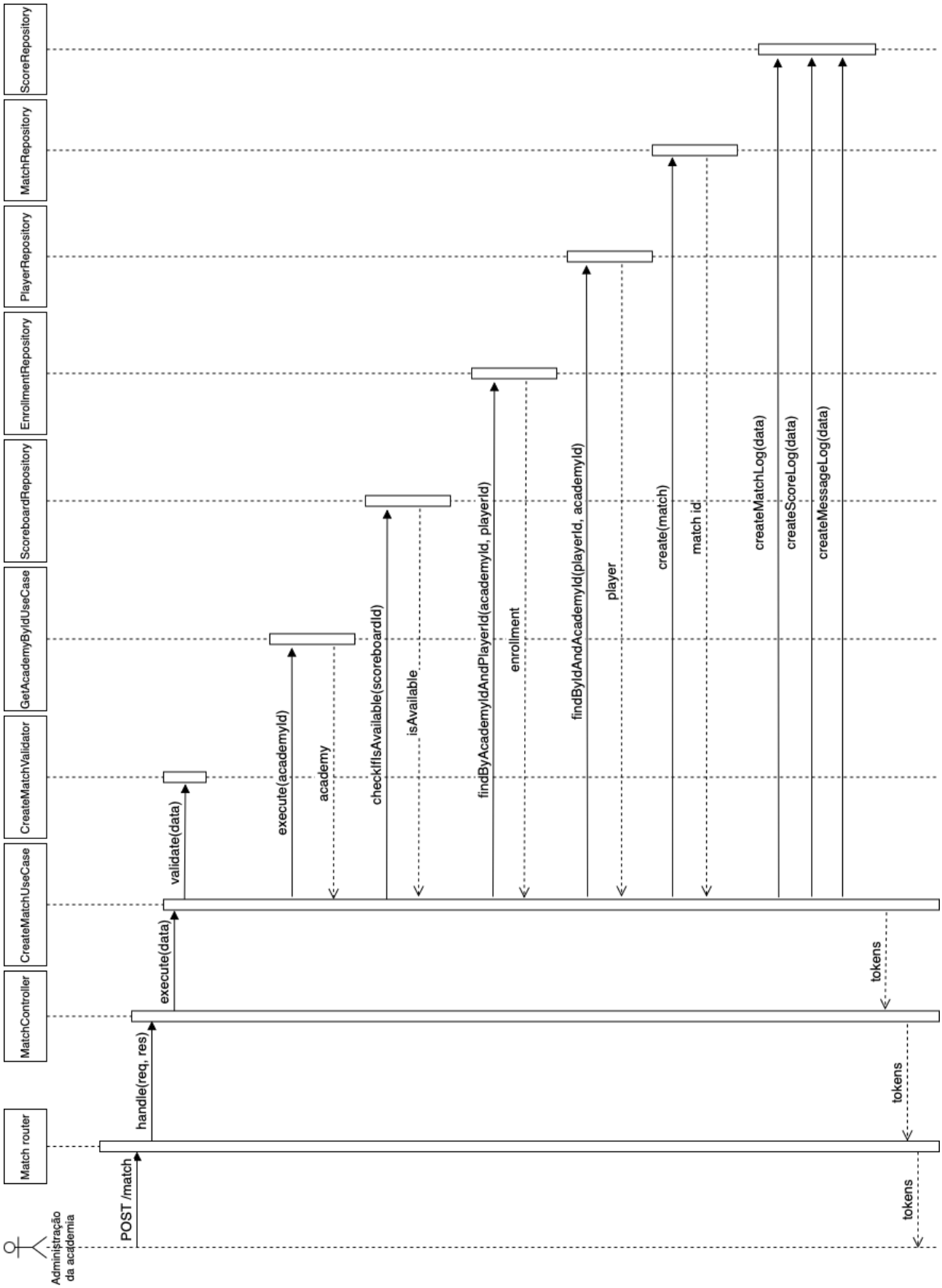
Sendo assim, a seguir são exploradas as responsabilidades de cada camada:

- **Roteamento**: nesta camada são especificadas as funções que serão executadas quando uma requisição HTTP chegar no servidor. Além disso, aqui também são aplicadas as restrições relacionadas à autenticação, por exemplo, garantindo que determinado recurso somente possa ser acessado se o usuário autenticado for um administrador do serviço. O corpo das funções dessa camada é bem pequeno, consistindo apenas em uma chamada para a camada de controle.
- **Controle**: nesta camada, as informações são extraídas do cabeçalho, corpo ou parâmetros da requisição. Com estas informações, o controlador chama o caso de uso correspondente à ação que o usuário deseja executar. Por fim, dependendo do resultado retornado pelo caso de uso, o controlador retorna um status HTTP indicando sucesso ou erro, além dos dados de resposta da requisição.
- **Eventos**: nesta camada são especificadas algumas funções que serão executadas quando acontecer algum evento relacionado ao Broker, como o recebimento de uma nova mensagem ou a inscrição de um cliente em um tópico específico.
- **Casos de uso**: esta é a camada principal da arquitetura, onde todas as regras de negócio são implementadas. Não está acoplada a nenhum *framework* ou biblioteca, o que permite que as dependências do projeto possam ser alteradas sem que sejam necessárias modificações na lógica central da aplicação (desde que as dependências mantenham a mesma interface). Os casos de uso foram desenvolvidos pensando em funcionalidades pequenas e específicas, de forma que possam ser reaproveitados e utilizados por outros casos de uso quando necessário.
- **Validação**: esta camada é responsável por validar os dados que chegam para os casos de uso, como garantir que todos os campos necessários para executar determinada operação foram preenchidos. Para auxiliar na validação dos dados, foi utilizada a biblioteca Joi[8], que conta com várias regras pré-definidas, como checar se um e-mail é válido ou se um campo tem um número mínimo de caracteres.
- **Repositórios**: a responsabilidade dos repositórios é abstrair o acesso aos dados, sejam estes dados presentes no PostgreSQL ou no MongoDB. Dessa forma, caso seja necessário modificar algo relacionado à manipulação dos dados, somente esta camada será afetada.

Para exemplificar o funcionamento destas camadas, a Figura 4.9 representa o fluxo completo que é executado em um dos principais casos de uso do sistema, a criação de partidas. Com exceção da camada de roteamento, que não faz o uso de classes, o restante das camadas são identificadas com os nomes que foram utilizados no projeto.

[8] https://github.com/sideway/joi

Figura 4.9. Diagrama de sequência para o caso de uso de criação de uma partida



1. Quando é enviada uma requisição do tipo POST para o caminho `/match`, uma função presente no roteador de partidas é executada. Então, esta função chama o método `handle` da classe `CreateMatchController`, informando os parâmetros `request` e `response`, que permitem acessar o conteúdo e manipular a resposta da requisição.
2. No método `handle` do `CreateMatchController`, o `id` da academia vinculada ao usuário logado é extraído dos dados de autenticação. Além disso, os dados que serão utilizados para a criação da partida são extraídos do corpo da requisição. Com esses dados, o controlador executa o método `execute` da classe `CreateMatchUseCase`. Por fim, caso a partida tenha sido criada com sucesso, os dados retornados pelo caso de uso são enviados para o usuário na resposta da requisição.
3. Para garantir que todos os dados necessários para a criação da partida foram enviados corretamente, a primeira ação executada pelo método `execute` da classe `CreateMatchUseCase` é chamar o método `validate` da classe `CreateMatchValidator`. Após validados os dados, o caso de uso pode prosseguir com a criação da partida. Para isso, são utilizados outros casos de uso auxiliares (passo 5), além de chamadas para os repositórios (passos 6 a 10).
4. A classe `CreateMatchValidator` verifica se todas as informações necessárias para criar a partida foram enviadas na requisição, como os jogadores e as regras escolhidas para o jogo. Caso tenha alguma informação inválida ou incompleta, é gerada uma exceção, o que faz com que seja retornada uma mensagem de erro para o usuário.
5. O caso de uso `GetAcademyByIdUseCase` é executado para verificar se a academia na qual a partida será criada realmente existe na base de dados. Se a academia não for encontrada, é retornado um erro para o usuário.
6. Se a partida for transmitida em um placar eletrônico físico, é chamado um método do `ScoreboardRepository` para verificar se o placar existe e se está disponível. Se alguma dessas condições não for satisfeita, é retornado um erro para o usuário.
7. A última verificação realizada pelo caso de uso é se os jogadores informados existem na base de dados e estão matriculados na academia. Para isso, é chamado um método do `EnrollmentRepository`, repositório responsável por lidar com o vínculo entre os jogadores e as academias.
8. O nome dos jogadores está sendo armazenado juntamente com os dados coletados durante a partida. Para obter o nome de um jogador a partir do `id`, está sendo utilizado um método presente no `PlayerRepository`.
9. Após todos os requisitos necessários para criar a partida forem atendidos, é chamado um método do `MatchRepository`, responsável por criar a partida no banco de dados.
10. Por fim, o caso de uso chama um método do `ScoreRepository`, para salvar a pontuação inicial da partida no MongoDB, bem como os dados necessários para o funcionamento da funcionalidade de refazer/desfazer as jogadas.

Para construir os objetos das classes das camadas, foi utilizada uma técnica da Engenharia de Software conhecida como Injeção de Dependências. Desta forma, a responsabilidade de instanciar as dependências não é mais do objeto em si, passando a ser responsabilidade de um outro mecanismo autorizado (MARTIN; COPLIEN, 2009). Esta mudança de responsabilidade permite desenvolver um código mais reusável, testável e manutenível (GOOGLE; SPRINGSOURCE, 2009). Como exemplo, o `CreateMatchController` não instancia um objeto do tipo `CreateMatchUseCase`, mas recebe uma instância do mesmo pelo método construtor. Da mesma forma, os casos de uso não instanciam os repositórios ou validadores que necessitam, mas também recebem todas as instâncias por meio do construtor. Porém, devido ao número de camadas e de classes, construir essa árvore de dependências manualmente seria uma tarefa repetitiva e trabalhosa. Portanto, foi utilizado um `container` de injeção de dependências chamado Awilix[9] para auxiliar neste processo.

A Figura 4.10 apresenta o diagrama de classes relacionadas ao caso de uso de criar a partida. Para simplificar a imagem, alguns métodos e dependências foram omitidos. O restante das classes das outras funcionalidades do sistema seguem o mesmo padrão, contendo os mesmos métodos destacados em negrito. É possível visualizar que o *controller* e o caso de uso recebem as dependências por meio do método construtor, seguindo o conceito de Injeção de Dependências. Uma ressalva em relação ao diagrama é que os modificadores de acesso apenas representam conceitualmente o nível de encapsulamento dos métodos e atributos, tendo em vista que não é possível definir membros privados no JavaScript.

Finalizando a explicação sobre o *backend*, a Figura 4.11 ilustra o diagrama de casos de uso que representa o sistema, contendo todas as funcionalidades desenvolvidas.

Frontend

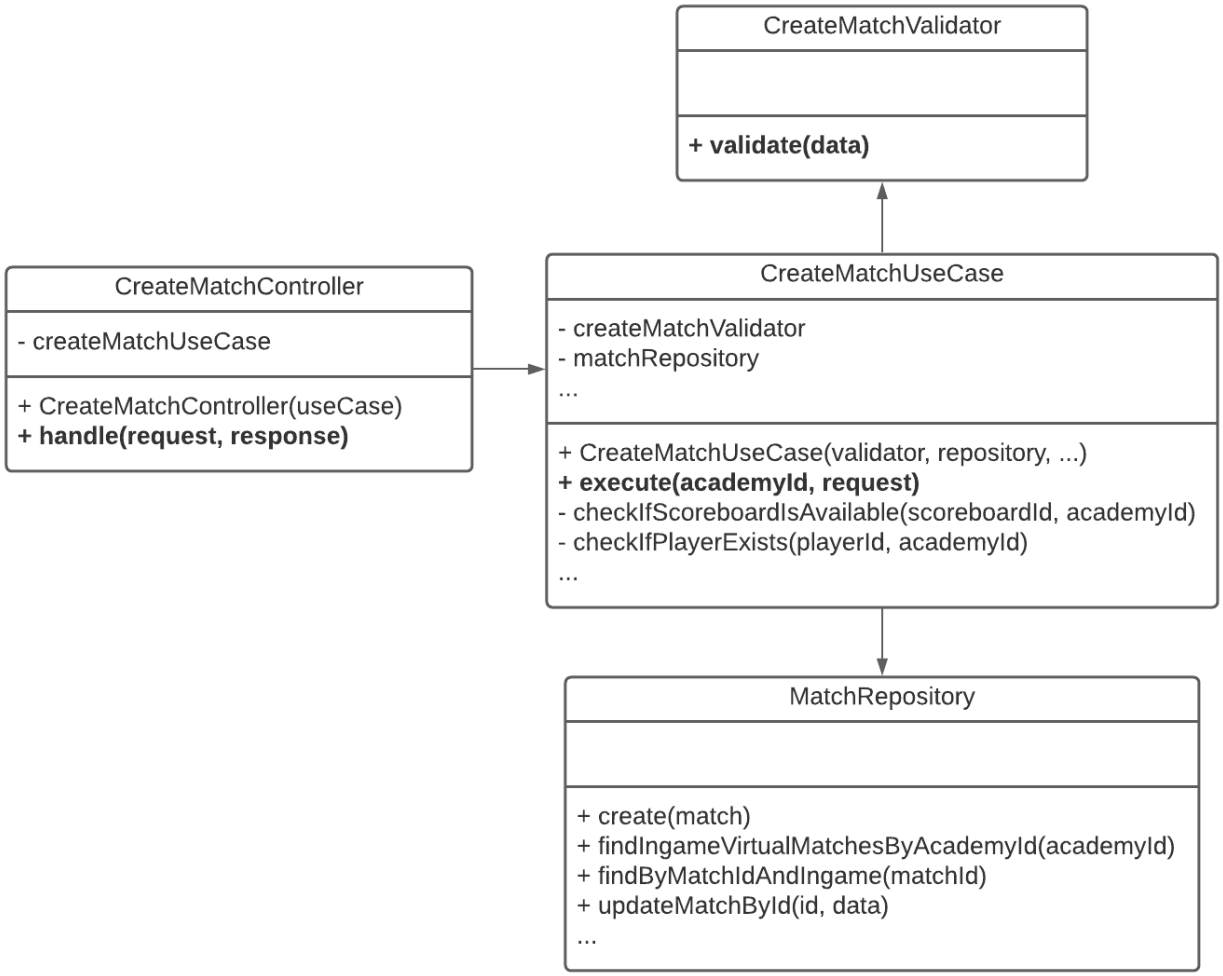
Como mencionado na subseção 4.3.2, o *frontend* da aplicação foi desenvolvido utilizando o React. Para uma melhor separação de conceitos, o código foi dividido em dois projetos: um para administração do serviço e outro para coordenação da academia. Para facilitar o entendimento, considere que a plataforma foi hospedada no domínio scoreboardapp.tech.

- Administração do serviço: atende somente a interface da Administração do serviço, acessível no endereço `https://admin.scoreboardapp.tech`.
- Coordenação da academia: atende as interfaces da Coordenação da academia, Controle do placar, Placar em tempo real e Placar em tempo real para monitores. A aplicação pode ser acessada pelo endereço: `https://[subdominio da academia].scoreboardapp.tech`.

A comunicação entre o *frontend* e o *backend* é realizada por meio de requisições HTTP, seguindo o padrão REST. Para isto, foi utilizada a biblioteca Axios[10], um cliente HTTP popular entre

[9] https://github.com/jeffijoe/awilix
[10] https://axios-http.com/

Figura 4.10. Diagrama de classes do caso de uso de criação da partida



a comunidade Node.js. Para se comunicar com o *Broker*, foi utilizada a biblioteca MQTT.js[11], um cliente para o protocolo MQTT que pode ser executado tanto no navegador quanto no Node.js.

Ambos os projetos foram hospedados utilizando o Vercel[12]. O serviço permite publicar de forma fácil e gratuita sites baseados em arquivos estáticos, como é o caso desta aplicação. Além disso, tem integração com o Github[13], atualizando automaticamente os serviços quando ocorrem alterações no código-fonte.

Por fim, a seguir são apresentadas as principais telas do sistema, iniciando pela Administração do Serviço. A Figura 4.12 mostra uma listagem de todas as academias que utilizam o serviço. Por meio desta tela é possível adicionar, pesquisar, editar e excluir as academias.

[11] https://github.com/mqttjs/MQTT.js
[12] https://vercel.com/
[13] https://github.com/

Figura 4.11. Diagrama de casos de uso

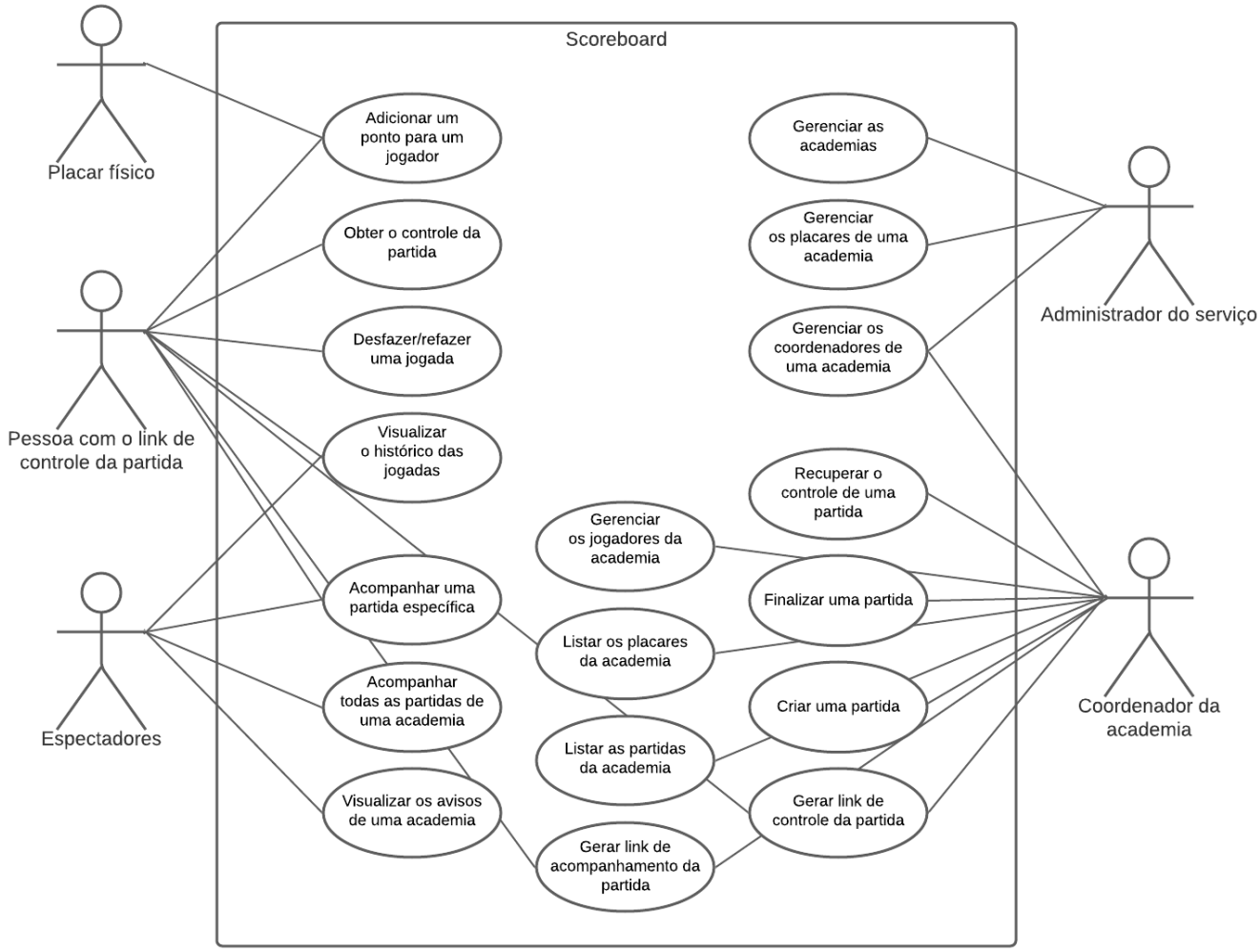
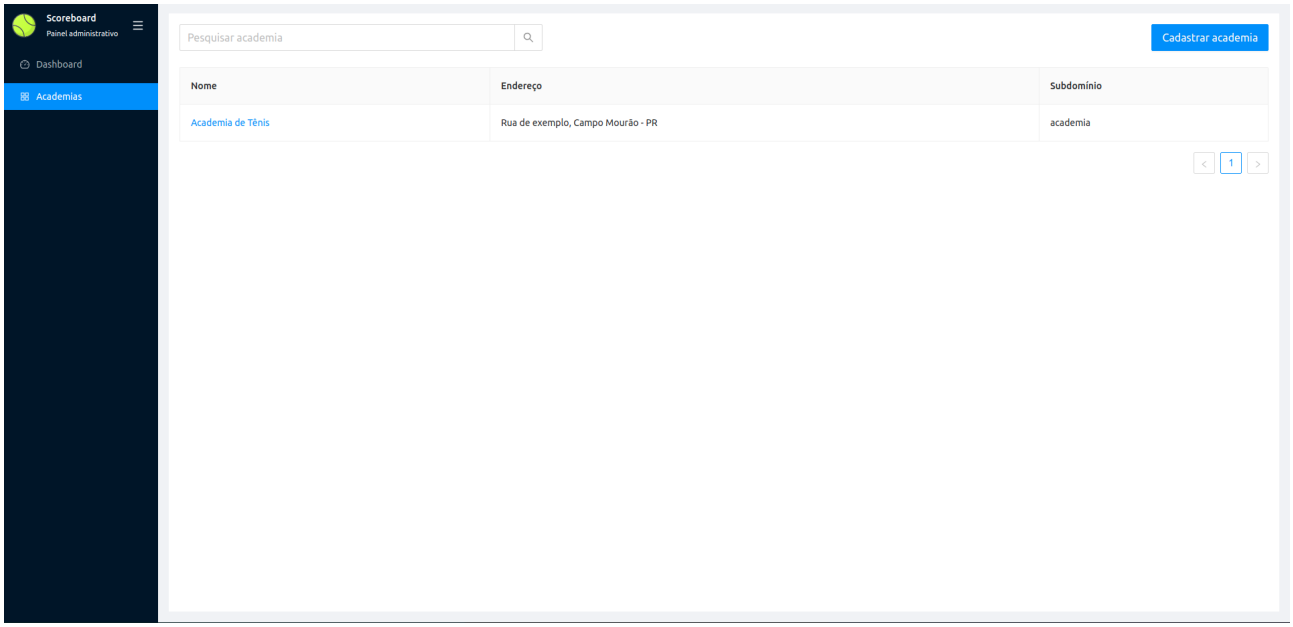
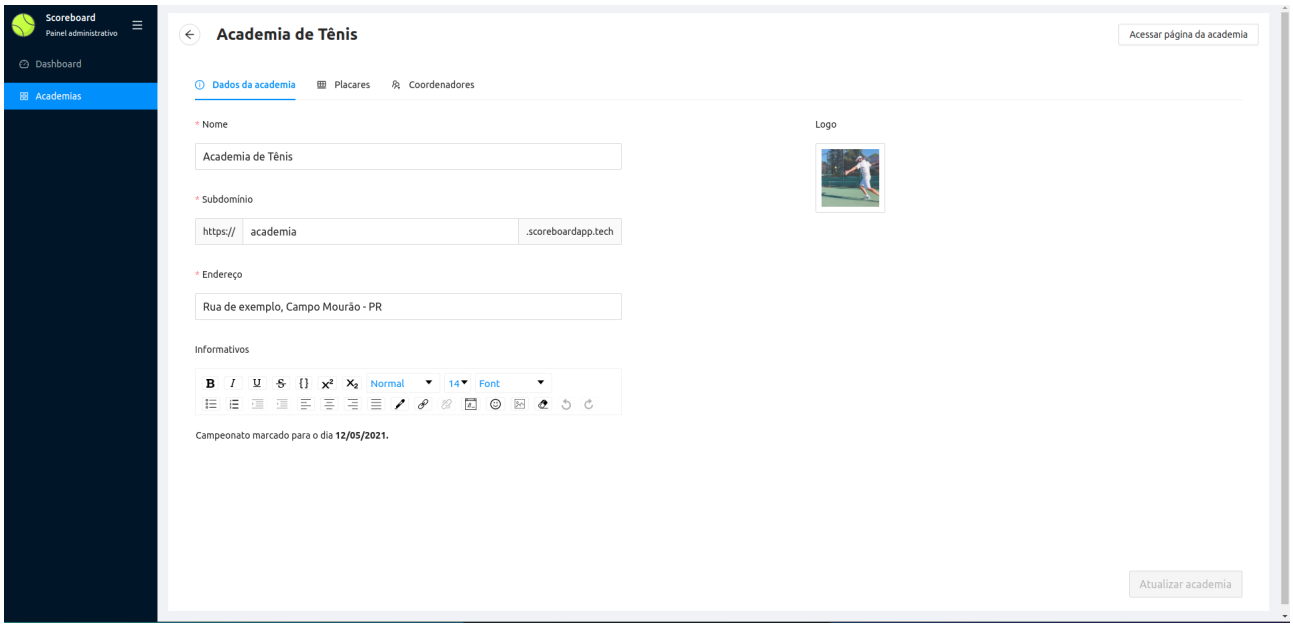


Figura 4.12. Administração do Serviço - Academias



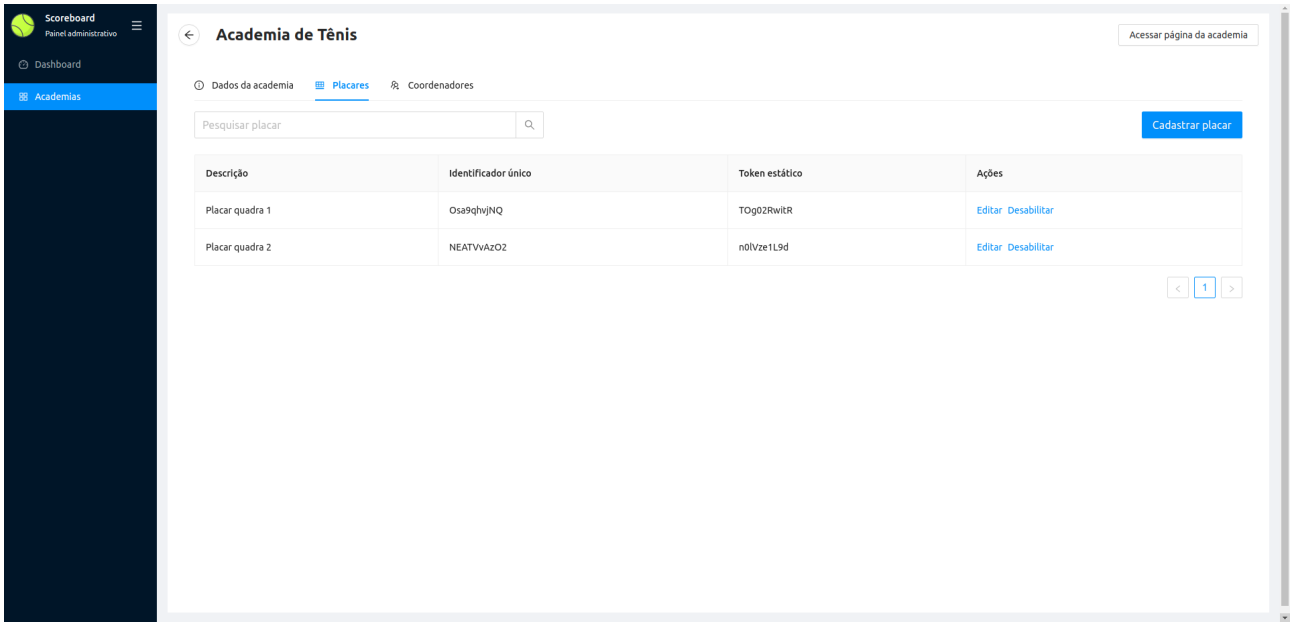
Ao clicar em uma linha da tabela, o usuário é redirecionado para a tela ilustrada na Figura 4.13, onde é possível atualizar os dados da academia, como o nome, subdomínio e informativos.

Figura 4.13. Administração do Serviço - Dados da academia



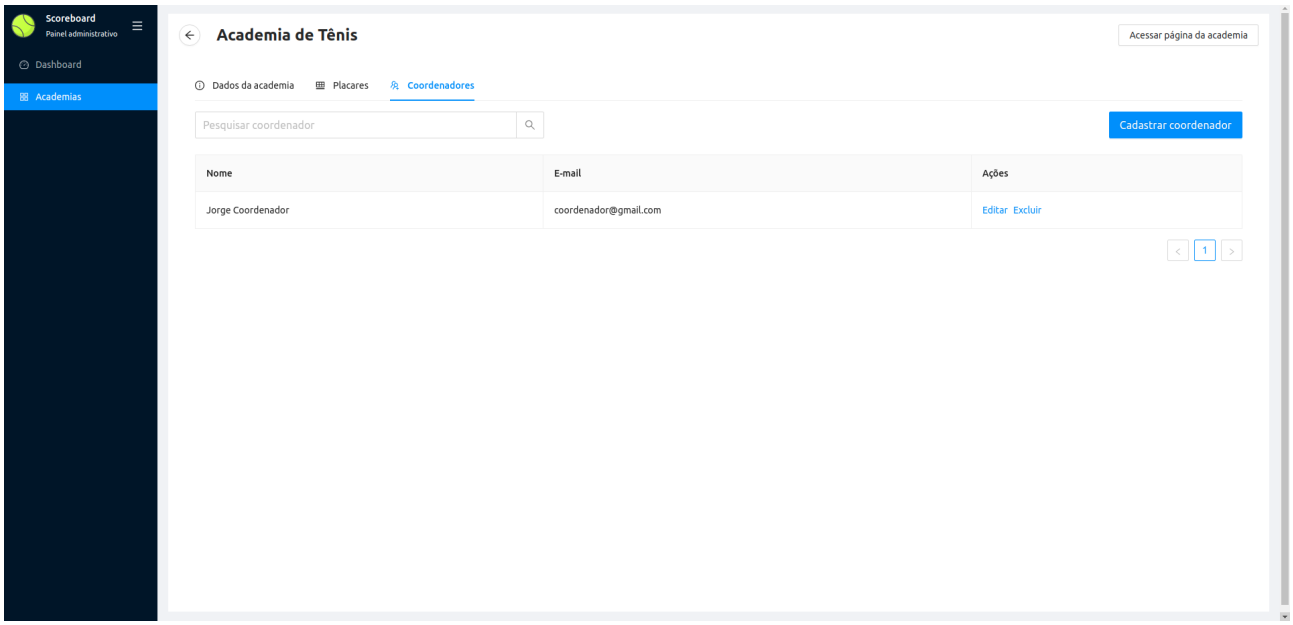
Na mesma tela, está disponível uma aba “Placares”, que permite gerenciar os placares utilizados pela academia, como é possível visualizar na Figura 4.14.

Figura 4.14. Administração do Serviço - Placares da academia



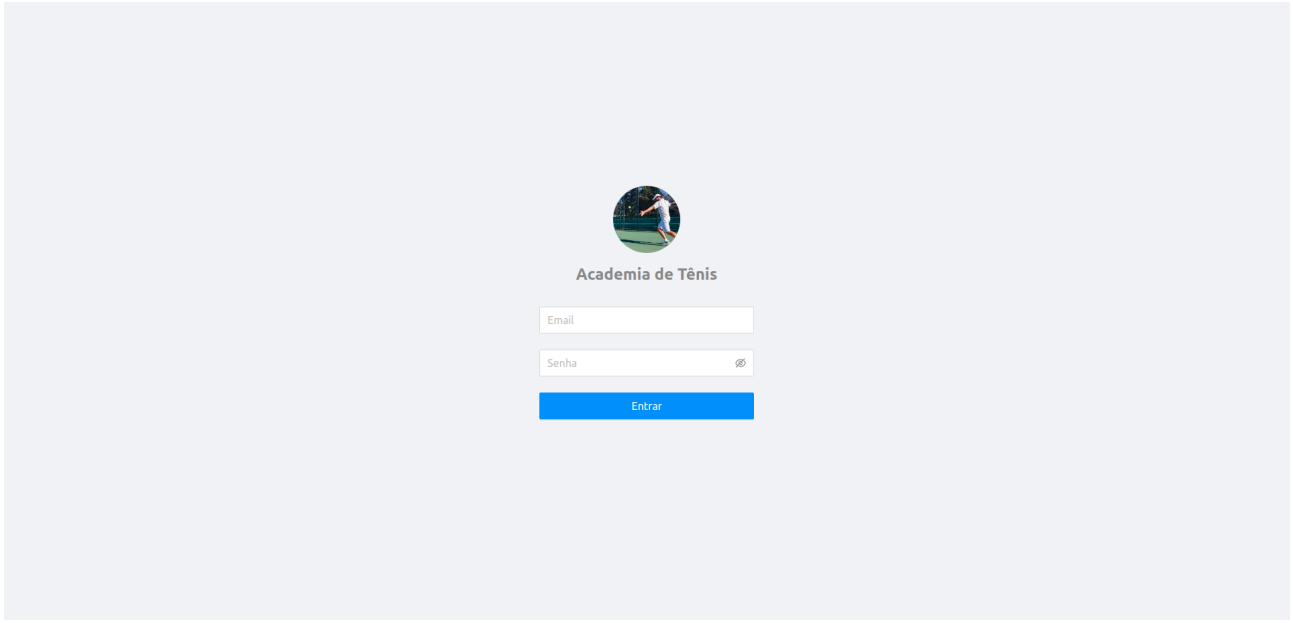
O usuário também pode acessar a aba “Coordenadores”, que permite gerenciar os coordenadores da academia, ilustrado na Figura 4.15.

Figura 4.15. Administração do Serviço - Coordenadores da academia



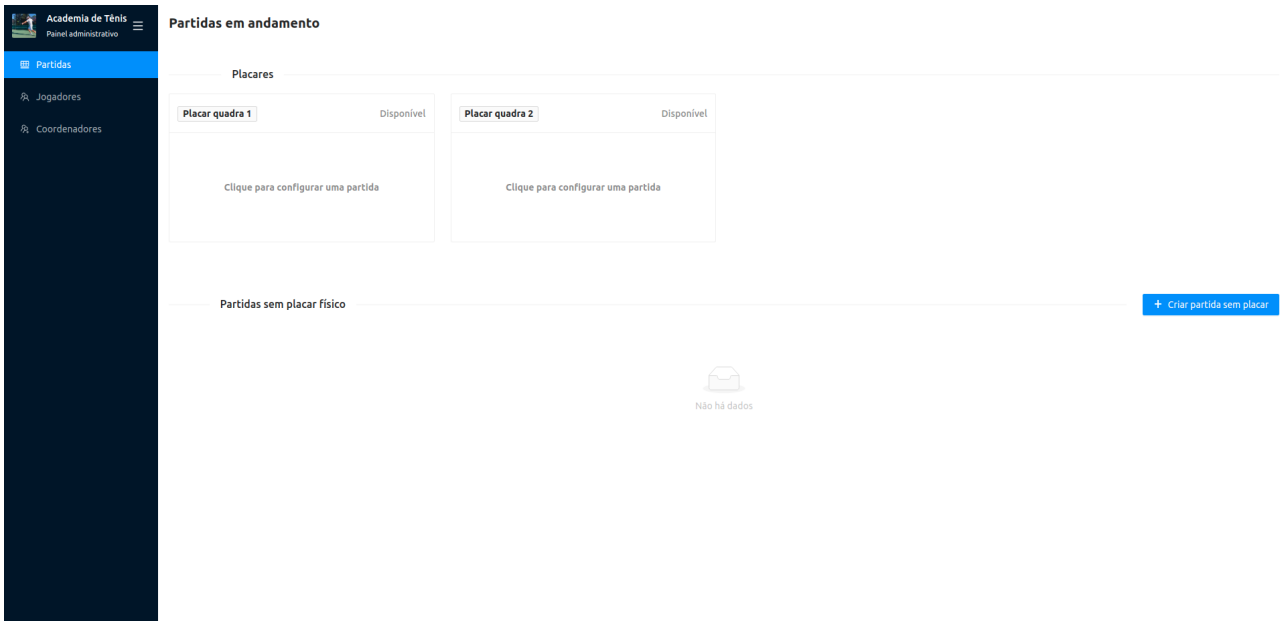
Ao clicar no botão “Acessar página da academia”, o usuário é redirecionado para as interfaces referentes à Coordenação da Academia, Controle do placar, Placar em tempo real e Placar em tempo real para monitores. A Figura 4.16 mostra a tela de autenticação do coordenador, que permite acesso à área administrativa da academia.

Figura 4.16. Coordenação da Academia - Autenticação do coordenador



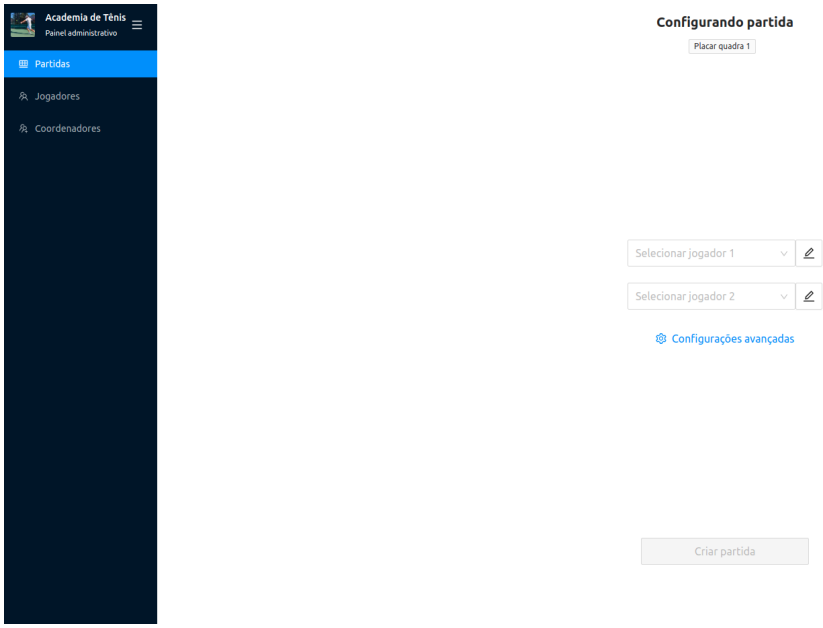
A tela inicial da parte administrativa lista todas as partidas em andamento e placares da academia, como mostra a Figura 4.17. Nesta tela, é possível criar tanto partidas que utilizam um placar físico, clicando em um placar disponível, ou criar partidas sem um placar físico, clicando no botão “Criar partida sem placar”.

Figura 4.17. Coordenação da Academia - Partidas em andamento



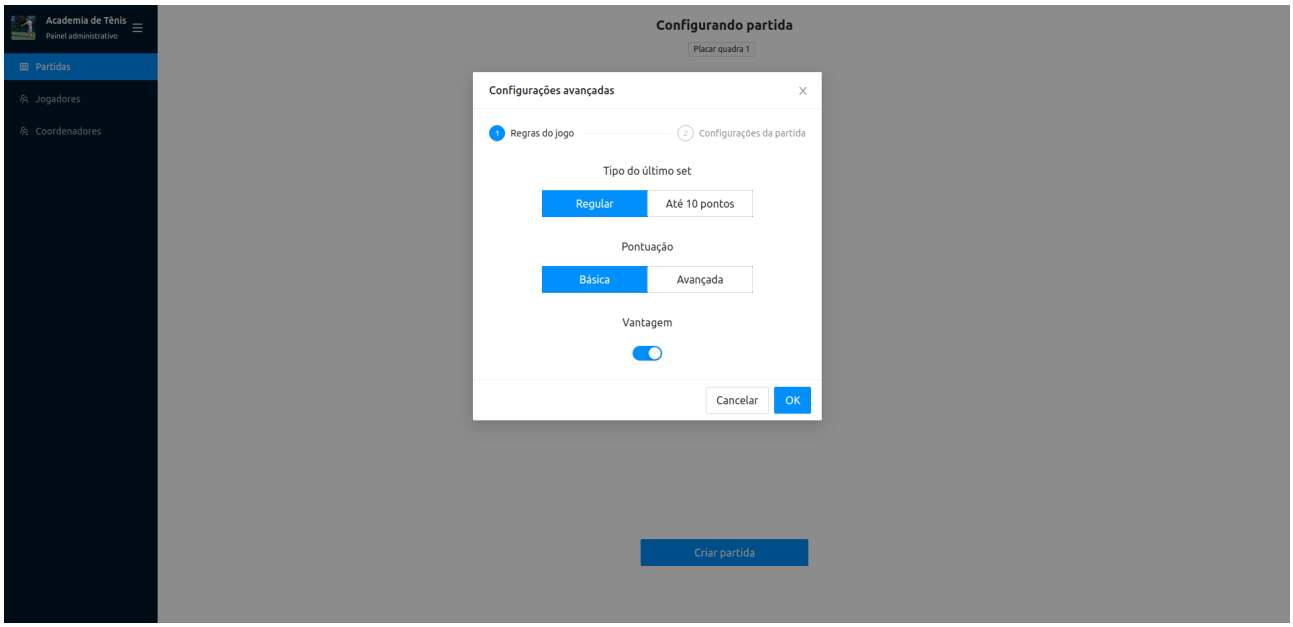
Ao escolher uma das opções mencionadas acima, é apresentada para o coordenador uma tela para selecionar os jogadores que irão participar da partida, dentre os que estão matriculados na academia. Porém, também existe a flexibilidade de digitar um nome qualquer ao invés de escolher um jogador, clicando no ícone de lápis, ilustrado na Figura 4.18.

Figura 4.18. Coordenação da Academia - Seleção dos jogadores



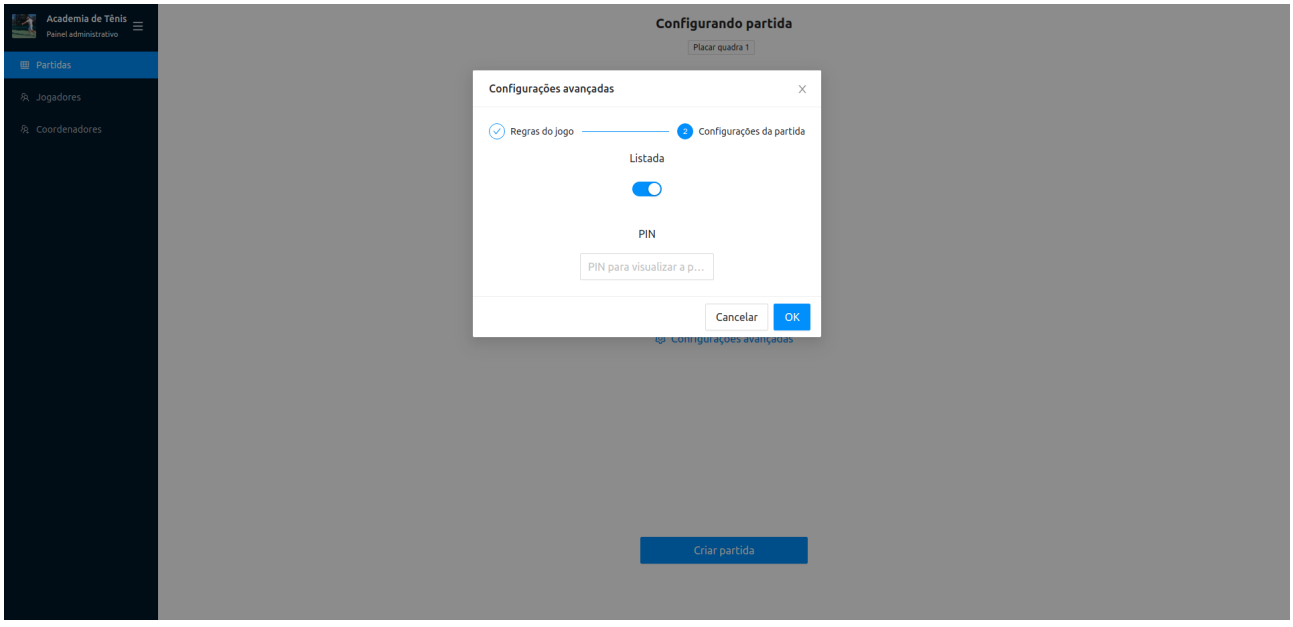
Como mencionado na Seção 4.1, uma das funcionalidades do sistema é a possibilidade de customização de algumas regras do tênis. Sendo assim, após selecionar os jogadores é apresentada uma interface onde é possível modificar alguns parâmetros da partida, como mostra a Figura 4.19.

Figura 4.19. Coordenação da Academia - Regras da partida



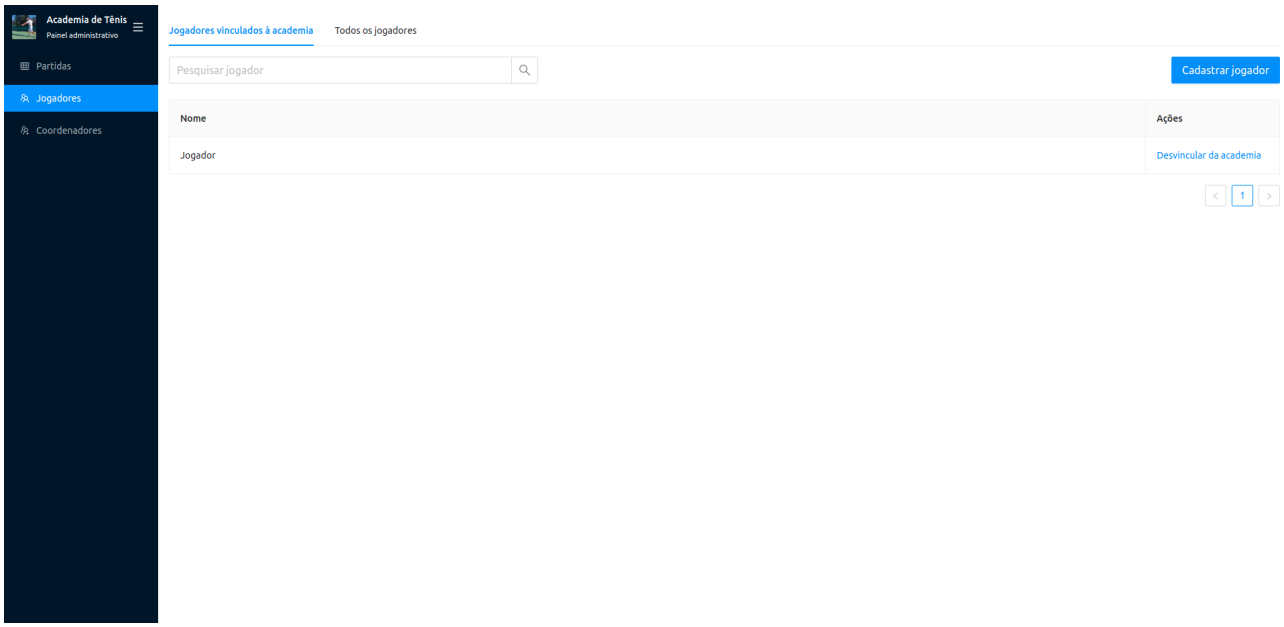
A próxima etapa é definir se a partida será listada (ficará visível na página inicial da academia) ou se será protegida por um PIN, como ilustrado na Figura 4.20. Ao finalizar este processo, é exibida uma caixa de confirmação para o usuário com todas as configurações que serão utilizadas no jogo. Após confirmar que tudo está correto, tudo estará pronto para a criação da partida.

Figura 4.20. Coordenação da Academia - Configurações da partida



Ainda na área administrativa da academia, o coordenador pode gerenciar os jogadores matriculados, como é possível ver na Figura 4.21. Além disso, na imagem também é possível visualizar um menu com o título “Coordenadores”, que leva à uma interface semelhante à da Figura 4.15.

Figura 4.21. Coordenação da Academia - Jogadores



Entrando na parte relacionada ao Placar em tempo real, a Figura 4.22 mostra a página inicial da academia. Nesta tela, é possível visualizar os avisos da academia, bem como as partidas que estão acontecendo no momento. A interface também é adaptada para visualização em dispositivos móveis, como mostra a Figura 4.23.

Figura 4.22. Placar em tempo real

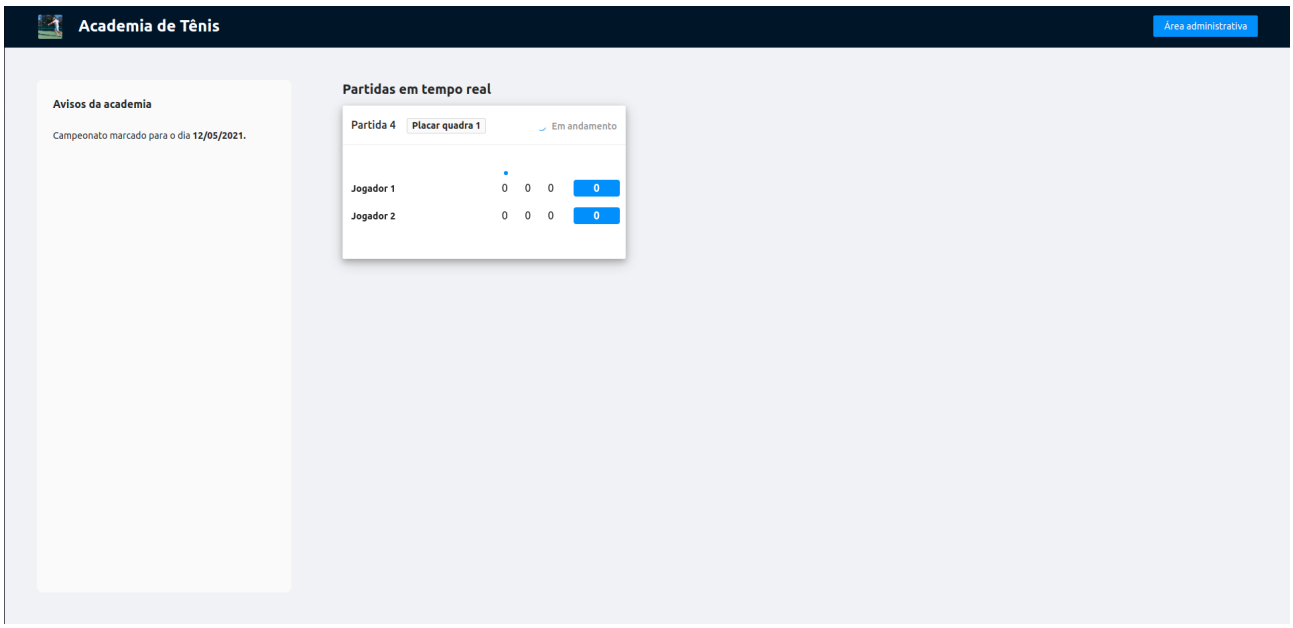
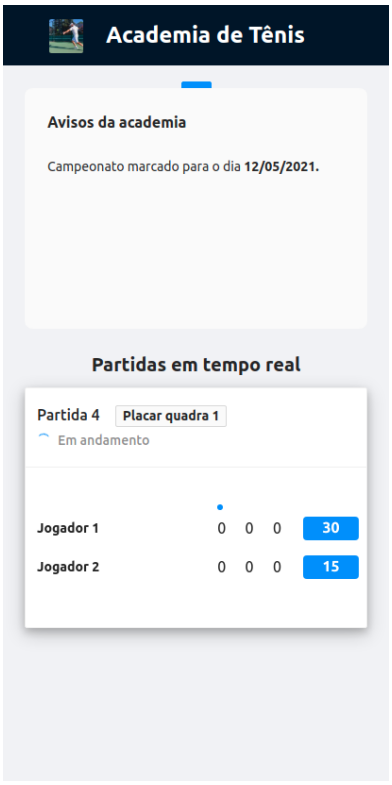


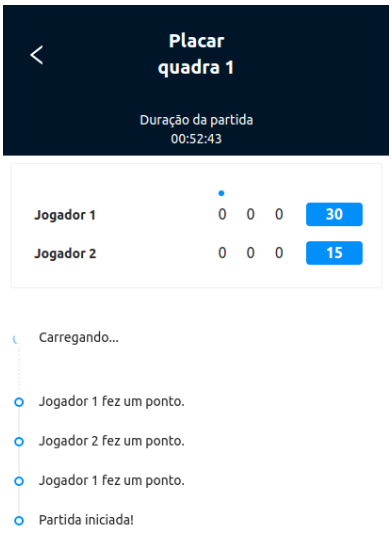
Figura 4.23. Placar em tempo real (dispositivos móveis)



Ao clicar em uma partida, é possível acompanhar o jogo com mais detalhes, como o histórico dos pontos. A interface também é adaptada tanto para dispositivos *desktop* (Figura 4.24) quanto dispositivos móveis (Figura 4.25).

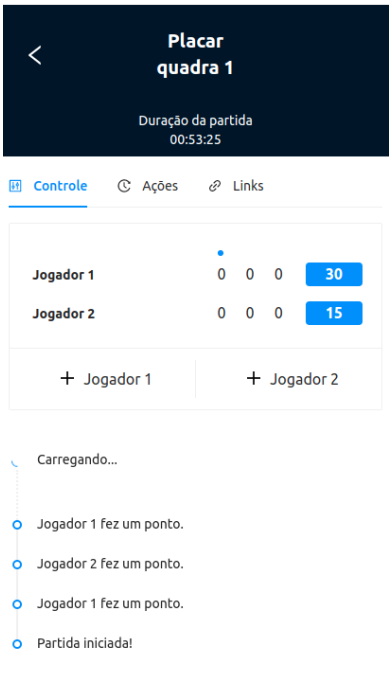
Figura 4.24. Placar em tempo real - Detalhes da partida

Figura 4.25. Placar em tempo real - Detalhes da partida (dispositivos móveis)



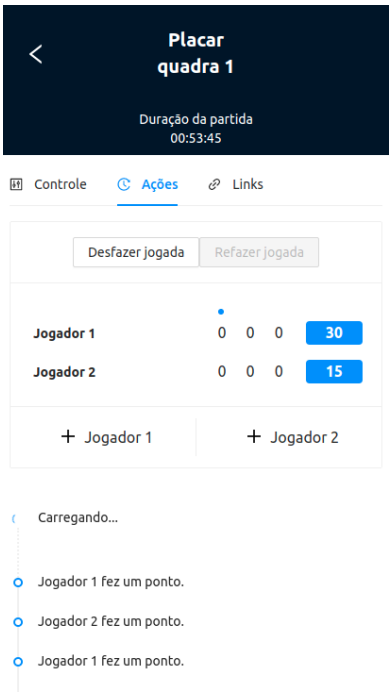
Saindo da parte do Placar em tempo Real, e entrando na parte do Controle do placar, a Figura 4.26 mostra a tela principal de controle das partidas, com os botões para adicionar os pontos para os jogadores.

Figura 4.26. Controle do placar - Tela principal



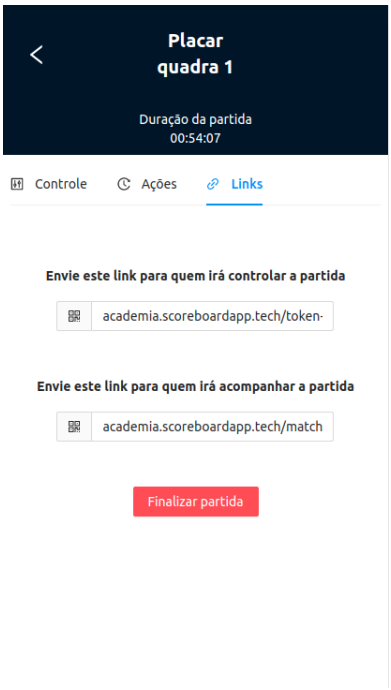
Ao mudar para a aba “Ações”, é possível desfazer ou refazer as últimas jogadas, como mostra a Figura 4.27.

Figura 4.27. Controle do placar - Ações



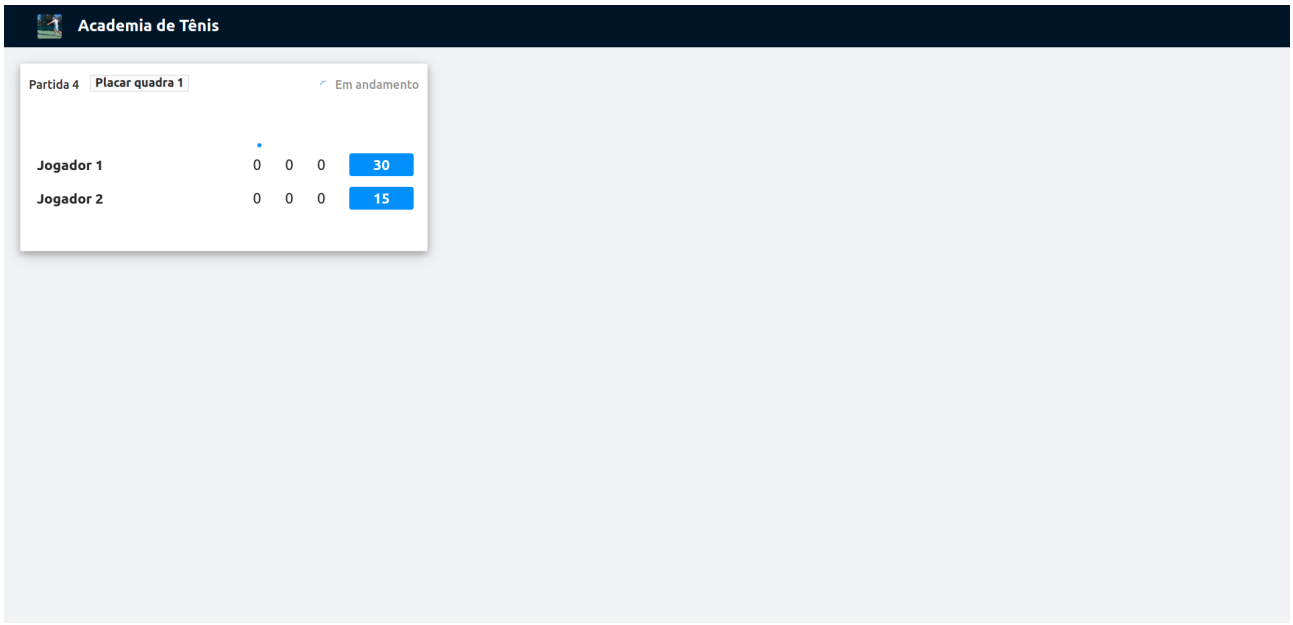
Finalizando a parte do Controle do placar, a aba “Links” permite visualizar e compartilhar os links para controle e acompanhamento da partida. Para o coordenador da academia, esta tela também permite finalizar a partida, como é possível visualizar na Figura 4.28.

Figura 4.28. Controle do placar - Links



Por último, a Figura 4.29 mostra a interface do Placar em tempo real para monitores, que conta com elementos mais destacados e fontes maiores.

Figura 4.29. Placar em tempo real para monitores



4.3.5 Teste

Após desenvolvido o sistema, a intenção inicial consistia em realizar testes com os usuários, a fim de levantar os pontos positivos e quais pontos poderiam ser melhorados. Para isto, uma possibilidade seria instalar o sistema em alguma academia de tênis da cidade durante alguma competição interna. Porém, tendo em vista a atual situação da Pandemia, não foi possível testar a plataforma com os usuários na academia. Para validar o funcionamento dos casos de uso, foram implementados testes unitários para as principais classes.

5 CONCLUSÃO

O tênis é um esporte que tem sido cada vez mais praticado como recreação no Brasil. Em grande parte das vezes, a modalidade é promovida por clubes recreativos, associações desportivas e academias especializadas. Uma das principais estratégias adotadas por essas organizações para manter o engajamento entre os praticantes, é a utilização de recursos do esporte profissional no esporte amador. Porém, essa estratégia nem sempre é factível, tendo em vista a dificuldade em encontrar opções cuja relação custo-benefício seja adequada à realidade das academias.

Sendo assim, este trabalho teve como objetivo desenvolver uma plataforma de software que permita as academias controlar e transmitir os resultados de jogos em seus torneios amadores, assim como acontecem nos torneios profissionais. A solução foi concebida levando em consideração os requisitos identificados e as restrições impostas pela realidade econômica e de infraestrutura da maioria das academias. A plataforma desenvolvida é composta por um conjunto de módulos para controlar o placar das partidas, transmitir a pontuação em tempo real, armazenar as estatísticas das partidas e disponibilizar o sistema como um serviço.

As funcionalidades implementadas na aplicação produzem benefícios para diferentes personagens envolvidos na participação de torneios de tênis amador. Para a academia, o maior benefício é o engajamento dos praticantes ao proporcionar um recurso visto nos torneios profissionais. Para o coordenador da academia, o sistema permite que o placar seja controlado a partir de uma interface web simples, contando com um suporte para permitir que ajudantes voluntários contribuam com a organização do torneio. Por fim, para os jogadores, a aplicação proporciona uma experiência que geralmente só é encontrada no meio profissional, como o armazenamento das estatísticas e a transmissão das partidas pela internet.

Para a validação do trabalho somente foi possível realizar testes unitários para verificar o funcionamento das responsabilidades implementadas nos casos de uso. Infelizmente, a validação *in loco* da plataforma não foi possível devido as medidas restritivas decorrentes da pandemia durante o desenvolvimento do trabalho. Como trabalho futuro, pretende-se utilizar a plataforma em um torneio real para identificar possíveis inconsistências e pontos de aprimoramento e o desenvolvimento de novos módulos para auxiliar no gerenciamento do torneio.

REFERÊNCIAS

CHEBUDIE, Abiy Biru; MINERVA, Roberto; ROTONDI, Domenico. **Towards a definition of the Internet of Things (IoT)**. 72, 73, 74 p. Tese (Doutorado), 08 2014.

EXAME. **Dobra o número de fãs de tênis no Brasil**. 2015. Disponível em: https://exame.com/blog/esporte-executivo/dobra-o-numero-de-fas-de-tenis-no-brasil/.

FACEBOOK. **Tutorial: Intro to React**. 2020. https://reactjs.org/tutorial/tutorial.html#what-is-react. [Acessado em: 11-Abril-2020].

FREDERICO, Rogério. **Saiba como funciona o Hawk-Eye no tênis**. 2017. https://blog.lptennis.com/saiba-como-funciona-o-hawk-eye-no-tenis/. [Acessado em: 11-abril-2020].

GOOGLE, Bob Lee; SPRINGSOURCE, Rod Johnson. **JSR 330: Dependency Injection for Java**. 2009. Disponível em: https://jcp.org/en/jsr/detail?id=330.

HIVEMQ. **Enterprise ready MQTT to move your IoT data**. 2020. https://www.hivemq.com/. [Acessado em: 27-Abril-2020].

IBM Cloud Education. **Docker**. 2020. https://www.ibm.com/cloud/learn/docker. [Acessado em: 13-Abril-2020].

KELLY, Alan. **The latest developments in tennis technology**. 2017. https://www.epiruslondon.com/blogs/tennis-hacks-blog/a-rundown-of-the-latest-developments-in-tennis-tech/. [Acessado em: 11-abril-2020].

LIGHT, Roger. Mosquitto: server and client implementation of the mqtt protocol. **Journal of Open Source Software**, The Open Journal, v. 2, n. 13, p. 265, 2017. Disponível em: https://doi.org/10.21105/joss.00265.

MARTIN, Robert C.; COPLIEN, James O. **Clean code: a handbook of agile software craftsmanship**. Upper Saddle River, NJ [etc.]: Prentice Hall, 2009. 157 p. ISBN 9780132350884 0132350882. Disponível em: https://www.amazon.de/gp/product/0132350882/ref=oh_details_o00_s00_i00.

MDN contributors. **An overview of HTTP**. 2019. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. [Acessado em: 13-Abril-2020].

MOSCAJS. **moscajs/aedes**. moscajs, 2020. Original-date: 2015-03-06T12:33:52Z. Disponível em: https://github.com/moscajs/aedes.

OASIS. **MQTT Version 3.1.1**. 2014. https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. [Acessado em: 27-Abril-2020].

OASIS. **MQTT Version 5.0**. 2019. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. [Acessado em: 27-Abril-2020].

ORACLE. **Database**. 2020. https://www.oracle.com/database/what-is-database.html. [Acessado em: 13-Abril-2020].

PUDER, Arno; RöMER, Kay; PILHOFER, Frank. **Distributed systems architecture - a middleware approach.** [S.l.: s.n.], 2006. 12 p. ISBN 978-1-55860-648-7.

RANGER, Steve. **What is the IoT? Everything you need to know about the Internet of Things right now**. 2020. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/. [Acessado em: 11-Abril-2020].

REDHAT. **O que é a Internet das Coisas (IoT)?** 2020. https://www.redhat.com/pt-br/topics/internet-of-things/what-is-iot/. [Acessado em: 11-Abril-2020].

WHO. **Physical activity**. World Health Organization, 2018. Disponível em: https://www.who.int/en/news-room/fact-sheets/detail/physical-activity.

YUAN, Michael. **Conhecendo o MQTT**. 2017. https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html. [Acessado em: 11-Abril-2020].