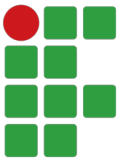


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DA PARAÍBA - CAMPUS CAMPINA GRANDE
CURSO SUPERIOR BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

ERICKSON TULIO RODRIGUES AZEVEDO

QUATTY'S: APLICAÇÃO PARA GERENCIAMENTO DE EQUIPES E LOCAIS PARA PRÁTICA DESPORTIVA

Campina Grande

2023

Erickson Tulio Rodrigues Azevedo

Quatty's: Aplicação para gerenciamento de equipes e locais para prática desportiva

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação

Orientador: Prof.Dr.Igor Barbosa da Costa

Campina Grande

2023

A994q Azevedo, Erickson Tulio Rodrigues.
Quatty's : aplicação para gerenciamento de equipes e locais para prática desportiva / Erickson Tulio Rodrigues Azevedo. - Campina Grande, 2022.
49f. : il.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Instituto Federal da Paraíba, 2023.
Orientador: Prof. Dr. Igor Barbosa da Costa.

1. Engenharia de software - desenvolvimento web 2. Desenvolvimento de sistema 3. Administração - desporto I. Costa, Igor Barbosa da II. Título.

CDU 004.4

Erickson Tulio Rodrigues Azevedo

Quatty's: Aplicação para gerenciamento de equipes e locais para prática desportiva

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação

Trabalho aprovado. Campina Grande, 22 de Junho de 2023:

Prof.Dr.Igor Barbosa da Costa
Orientador

Prof.Dr.Cesar Rocha Vasconcelos
Membro da Banca

Prof.Ma.Iana Daya Cavalcante Facundo Passos
Membro da Banca

Campina Grande
2023

Dedico este trabalho para meus pais, meus irmãos, amigos, tia e avó, que sempre me apoiaram e me fizeram persistir nessa jornada.

Agradecimentos

Gostaria de expressar minha profunda gratidão a todas as pessoas que me apoiaram ao longo desta jornada desafiadora. Houve momentos em que pensei em desistir, lutando para me tornar o melhor aluno e equilibrar o trabalho e os estudos, enfrentando dificuldades financeiras e outros obstáculos. No entanto, graças à ajuda de inúmeras pessoas, superei essas dificuldades. Portanto, gostaria de dedicar meus sinceros agradecimentos a:

Em primeiro lugar, agradeço a Deus por me conceder saúde, resiliência e oportunidades de triunfar, mesmo diante de tantas adversidades.

A minha família, Maria do Socorro e Wenderson Ewerton, que me apoiaram ao longo de todo o percurso de maneiras diversas. Agradeço por todo o amor, confiança e companheirismo.

A minha avó, Teresinha Medeiros, e minha tia, Telma Medeiros, que nos momentos difíceis em que eu, meu irmão e minha mãe não conseguíamos ver uma saída, nos deram apoio de várias formas, sempre incentivando-me a dar o melhor de mim.

Ao professor Igor Costa, meu orientador, pela orientação necessária para tornar este trabalho possível. Agradeço também por todos os ensinamentos, auxílio, confiança e paciência.

Aos meus amigos que estiveram presentes e me ajudaram em diversos momentos até aqui: Felipe Lamerck, Kelvi Henrique, Alisson Silva, Cauê Rennã, Alexsander Oliveira, Lucas Yago, Matheus Eduardo, Verônica Souza e tantos outros. Além disso, sou grato a todos os amigos que fiz durante toda essa jornada da graduação, como Mozart Lima, Iury Fernandes, Arthur Venâncio, Edivan Junior, Hevlla Souza, Amanda Camilo, Bianca Rangel, Ana Farias, Maria Luiza e muitos outros. Vocês também fazem parte dessa conquista e tornaram-na possível.

Mais uma vez, expresso minha profunda gratidão a todos vocês. Sou verdadeiramente abençoado por ter pessoas tão incríveis em minha vida, cujo apoio e amizade foram fundamentais para minha jornada acadêmica. Obrigado do fundo do meu coração.

Desvendando a harmonia sutil entre o vigor do esporte e o engenho tecnológico: uma dança sinfônica na busca incessante pela primazia.

Resumo

A prática regular de atividade física é essencial para a manutenção da saúde física e para promover o bem-estar mental e emocional. No entanto, indivíduos que se engajam nessa prática frequentemente se deparam com obstáculos, tais como a falta de locais adequados para realizar exercícios e a ausência de parceiros de treino, especialmente em esportes coletivos. Atualmente, diversas aplicações tecnológicas têm como objetivo auxiliar nessa demanda específica. Este trabalho analisou tais ferramentas, identificou suas deficiências e estabeleceu requisitos para suprir essas lacunas. Com base nesses requisitos, foi desenvolvido o protótipo de uma aplicação web chamada Quattys, que promove a integração de praticantes de esportes e facilita a reserva ou locação de espaços para a prática esportiva. Este trabalho apresenta os elementos fundamentais do processo de desenvolvimento do Quattys, desde sua concepção até a fase de implementação.

Palavras-chave: Desenvolvimento web; Acessibilidade; Sistema de administração; Engenharia de software; Atividades físicas.

Abstract

Regular physical activity is essential for maintaining our physical health and promoting our mental and emotional well-being. However, we often encounter obstacles when trying to engage in this healthy practice, such as the challenge of finding suitable exercise facilities and the lack of training partners, particularly in team sports. Fortunately, there are several technological applications available today that aim to assist with this specific demand. In this study, we analyzed these tools, identified their limitations, and established requirements to address these gaps. Based on these requirements, we developed a prototype of a web application called Quattys, which aims to facilitate the integration of sports enthusiasts and the reservation or rental of sports facilities. This paper presents the key aspects of the Quattys development process, from conception to implementation.

Key-words: Web development; Accessibility; Management system; Software engineering; Physical activities.

Lista de ilustrações

Lista de abreviaturas e siglas

DEV	*Development*
REST	*Representational State Transfer*
API	*Applications Protocol Interface*
CI	*Continuous Integration*
MVC	*Model-View-Controller*
CSS	*Cascading Style Sheets*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
SPA	*Single-Page Applications*
JSON	*JavaScript Object Notation*
XML	*eXtensible Markup Language*
GUI	*Graphical User Interface*
LDAP	*Lightweight Directory Access Protocol*
MVVM	*Model-View-ViewModel*
SGBD	*Sistema Gerenciador de Banco de dados*

Lista de Códigos

Sumário

1 Introdução

A Educação Física desempenha um papel fundamental no desenvolvimento biopsicossocial de crianças e adolescentes. Ela ocupa um lugar de destaque na Educação Básica, proporcionando acesso a práticas físico-esportivas e equipando o aluno para aproveitar ao máximo essas atividades. Os Parâmetros Curriculares Nacionais (PCN) designam a Educação Física como a disciplina responsável pelo ensino de esportes, ginástica, dança, jogos, atividades rítmicas e expressivas, e pelo conhecimento sobre o próprio corpo para todos os alunos (PIMENTA, 2015). Além de promover valores como trabalho em equipe, disciplina, espírito de superação e respeito ao próximo, a Educação Física também exerce impacto positivo na saúde e bem-estar dos jovens. Portanto, é uma disciplina essencial para o desenvolvimento holístico dos alunos (PIMENTA, 2015).

Contudo, apesar do papel crucial das atividades físicas na formação de jovens e adultos, o Brasil enfrenta desafios significativos em relação à infraestrutura adequada para a formação de atletas pelas instituições públicas. Muitas vezes, esses recursos são insuficientes ou inexistentes. De acordo com o Censo Escolar da Educação Básica de 2020, 47% das escolas do ensino fundamental I ao médio, que atendem jovens de 6 a 17 anos no Brasil, não possuem instalações adequadas para a prática desportiva (VECCHIOLI, 2021).

Além da infraestrutura inadequada, o Brasil também enfrenta problemas na gestão e manutenção de espaços públicos voltados à prática desportiva, como parques, praças e quadras. Frequentemente, esses locais ficam sem manutenção por períodos prolongados, prejudicando a população que depende desses espaços para a prática de atividades físicas.

Para enfrentar esses desafios, este trabalho propõe a implementação da tecnologia para criar uma plataforma de intermediação de acesso a espaços públicos e privados, como escolas, faculdades e outras instituições que demandam processos burocráticos para a liberação do espaço para práticas esportivas. Adicionalmente, esta plataforma tem como um de seus objetivos conectar pessoas interessadas em praticar esportes, criando uma rede de compartilhamento de informações sobre locais disponíveis para atividades físicas.

Assim, a tecnologia pode se tornar uma aliada poderosa na resolução do problema de infraestrutura e na melhoria da gestão dos espaços públicos voltados à prática desportiva no Brasil. A criação desta plataforma pode facilitar o acesso a esses espaços de maneira mais eficaz, permitindo que um número maior de pessoas possa praticar atividades físicas e, consequentemente, melhorar sua saúde e bem-estar.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste projeto é desenvolver um aplicativo que promova a integração de desportistas e facilite a reserva ou locação de espaços publico e privados para a prática de esportes. O intuito é contribuir para a melhoria da infraestrutura esportiva no Brasil e fomentar a atividade física entre a população.

1.1.2 Objetivos Específicos

A fim de atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Desenvolver uma interface que permita aos usuários navegar pela plataforma e realizar tarefas básicas sem necessidade de treinamento ou assistência;
- Estabelecer um banco de dados seguro e confiável, destinado ao armazenamento de dados sobre usuários, locais para prática esportiva e reservas;
- Implementar uma API RESTful, que possibilite uma comunicação eficaz entre o *frontend* e o *backend* da aplicação;
- Projetar um *frontend* responsivo e eficiente, que permita aos usuários realizar buscas por locais de prática esportiva, fazer reservas, verificar disponibilidade de espaços e efetuar pagamentos;
- Desenvolver o sistema visando assegurar a segurança e a privacidade dos dados dos usuários;

1.2 Relevância e Contribuições

O projeto proposto reveste-se de significativa relevância para a comunidade esportiva, ao buscar simplificar e otimizar o processo de reserva e locação de espaços públicos e privados dedicados à prática esportiva. Tal proposta fomenta uma comunidade esportiva mais engajada, organizada e segura, tornando a prática esportiva mais acessível e incentivando hábitos saudáveis.

Com a implementação do sistema proposto, atletas e gestores poderão interagir de forma mais eficiente, aprimorando a organização e a gestão das atividades esportivas. Adicionalmente, o sistema oferecerá um nível extra de segurança, permitindo que os gestores monitorem o uso dos seus espaços esportivos e adotem medidas preventivas, quando necessário.

Segue abaixo alguns exemplos de como a ferramenta proposta poderá ser aplicada e os benefícios específicos que ela poderá oferecer para os usuários:

- Acesso facilitado a espaços públicos: um dos principais benefícios da ferramenta será a possibilidade de facilitar o acesso a espaços públicos destinados à prática desportiva. Por exemplo, se uma pessoa estiver buscando um local para jogar basquete em sua cidade, mas desconhecer a existência de quadras disponíveis, por meio da ferramenta, ela poderá facilmente localizar quadras próximas, verificar a disponibilidade e agendar o horário de uso. Dessa maneira, a ferramenta contribuirá para tornar mais acessíveis esses espaços e promoverá a prática de atividades físicas.

- Facilitação da liberação de espaços privados: a ferramenta também poderá facilitar a liberação de espaços privados, como escolas e faculdades, para a prática esportiva. Por exemplo, se um grupo de alunos desejar utilizar o ginásio de uma escola pública para jogar vôlei, mas não souberem como solicitar a liberação do espaço, por meio da ferramenta, eles poderão enviar a solicitação diretamente para a direção da escola e acompanhar o processo de autorização. Isso agilizará o processo e incentivará o uso desses espaços por parte da comunidade.

- Compartilhamento de informações e criação de grupos de prática esportiva: a ferramenta também permitirá que as pessoas compartilhem informações sobre locais para prática de atividades físicas, promovendo a criação de grupos de prática esportiva. Por exemplo, se um grupo de amigos desejar jogar futebol, mas não dispuser de um campo, por meio da ferramenta, eles poderão encontrar um campo disponível para agendar a partida. Além disso, a ferramenta permitirá que outras pessoas se juntem a esses grupos, ampliando a prática esportiva e incentivando a socialização. Essa funcionalidade também contribuirá para fortalecer a comunidade esportiva, incentivando a participação em atividades em grupo e a formação de novas amizades.

2 Fundamentação Teórica

Este Capítulo destina-se a estabelecer uma base teórica sólida para a compreensão do trabalho, apresentando conceitos fundamentais e ferramentas que foram utilizadas durante o desenvolvimento do *software*. A exposição desses conceitos e tecnologias é crucial para entender as escolhas feitas e o caminho seguido durante o projeto.

2.1 Arquitetura de Software

O termo "arquitetura de *software*"possui mais de uma definição. Uma das definições mais comuns considera que a arquitetura de *software* se preocupa com o projeto em uma perspectiva macro, deixando de lado a organização e as interfaces de classes individuais e focando em unidades de maior tamanho, como pacotes, componentes, módulos, subsistemas, camadas ou serviços (VALENTE, 2020).

Outra definição, proposta por Ralph Johnson, considera que a arquitetura de *software* trata-se do conjunto de decisões mais importantes do projeto. Essa definição leva em conta que a arquitetura não se trata apenas de um conjunto de módulos, mas sim de um conjunto de decisões, que inclui a definição dos módulos principais de um sistema, assim como outras decisões importantes, como a escolha da linguagem de programação e do banco de dados a ser utilizado durante o desenvolvimento (VALENTE, 2020).

2.1.1 Arquitetura em camadas

A arquitetura em camadas é um padrão amplamente utilizado no *design* de *software*, especialmente na construção de sistemas de informação corporativos. Essa abordagem remonta aos primórdios dos sistemas de software de maior porte nas décadas de 60 e 70. Nesse modelo, as classes são agrupadas em módulos maiores, conhecidos como camadas. Cada camada tem uma função específica no sistema e pode interagir somente com as camadas abaixo ou acima dela. Isso cria uma separação clara de responsabilidades e torna o sistema mais organizado e escalável (VALENTE, 2020).

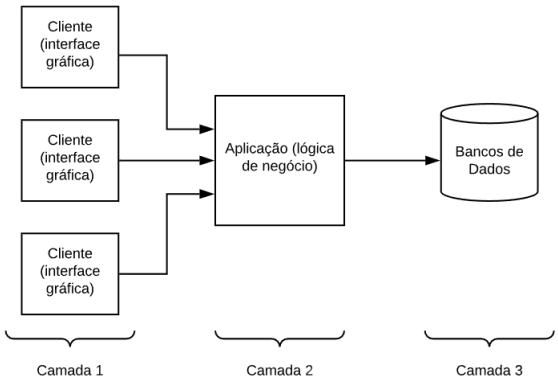
A Figura 1 apresenta um sistema com arquitetura em camadas, composto por três camadas distintas (VALENTE, 2020):

- Camada de Apresentação: é responsável por toda a interação do usuário com o sistema. Ela é encarregada tanto de exibir informações quanto de coletar e processar entradas e eventos de interfaces, como cliques em botões e marcação de texto. Essa

camada pode ser uma aplicação *desktop* em sistemas operacionais com interfaces gráficas, como o Linux, ou uma aplicação web.

- Camada de Aplicação: é responsável por implementar as regras de negócio. Esta camada é onde a maioria da lógica de processamento de dados ocorre.
- Camada de Dados: é responsável por gerenciar a persistência de dados e fornecer acesso aos dados para outras camadas. Esta camada pode incluir banco de dados, arquivos de dados e outras fontes de armazenamento de dados.

Figura 1 – Arquitetura em três camadas



Fonte: Adaptado de (VALENTE, 2020)

2.1.2 API REST

A API REST (*Representational State Transfer*) é uma interface de programação de aplicações que segue os princípios de arquitetura REST. Essa abordagem possibilita a interação com serviços web de forma eficiente e consistente, baseada em princípios simples e bem definidos.

A utilização de métodos HTTP (GET, POST, PUT, DELETE) para realizar operações em recursos, URLs para identificar esses recursos, formatos de representação de dados como *JSON* ou *XML*, e o estado de representação *(stateless)* entre cliente e servidor são alguns dos princípios que formam a base da arquitetura REST (HAT, 2020).

Por meio de uma API REST, os usuários podem realizar operações em recursos específicos, como recuperar dados, criar novos recursos, atualizar recursos existentes ou excluir recursos. Essas operações são realizadas por meio dos métodos HTTP mencionados acima, que são utilizados para indicar a ação a ser realizada no recurso.

Além disso, a arquitetura REST é baseada no princípio de estado de representação onde todas as requisições são independentes umas das outras, sem armazenar informações

do estado anterior da aplicação. Isso torna a aplicação mais escalável e fácil de manter, além de reduzir a sobrecarga de comunicação entre cliente e servidor(HAT, 2020).

2.1.3 Keycloack

O desenvolvimento de aplicações web pode ser desafiador, especialmente quando é necessário implementar métodos de segurança, como autenticação e autorização para usuários com diferentes perfis de acesso. Essa complexidade aumenta à medida que a aplicação cresce, pois é necessário proteger os dados dos usuários e garantir que a aplicação seja escalável.

Para lidar com essas questões, o Keycloak oferece uma solução de login único para aplicações web e serviços web RESTful. O objetivo do projeto é simplificar e tornar mais eficiente o processo de proteger aplicações e serviços. Os recursos de segurança que normalmente precisariam ser escritos pelos desenvolvedores para suas aplicações são fornecidos imediatamente e podem ser personalizados para atender às necessidades individuais (KEYCLOAK, 2023).

O Keycloak suporta diversos protocolos padrões de mercado, como o OAuth 2.0, OpenID Connect e SAML 2.0, e fornece uma série de recursos, incluindo:

- Atuar como um servidor de autenticação centralizado
- Fornecer federação de usuários para sincronizar usuários de servidores LDAP e Active Directory
- Integrar-se com provedores de identidade de terceiros, incluindo redes sociais
- Oferecer APIs REST e uma GUI de administração para gerenciamento centralizado de usuários, funções, mapeamentos de funções, clientes e configurações.

O Keycloak é uma solução moderna e escalável que traz agilidade e confiabilidade ao sistema. Além disso, permite que os usuários escolham a forma como desejam se autenticar, por meio de redes sociais ou cadastro no sistema. Com o Keycloak, desenvolvedores podem facilmente implementar segurança em suas aplicações, sem precisar escrever código complexo para isso.

2.2 Tecnologias

2.2.1 Java

Lançado em 1995 pela Sun Microsystems, o Java é uma linguagem de programação amplamente utilizada para desenvolvimento de aplicações web. Com mais de duas décadas

de existência, a linguagem continua sendo uma escolha popular entre os desenvolvedores, com milhões de aplicações Java em uso hoje. O Java é uma linguagem multiplataforma, orientada a objetos e centrada em rede, o que significa que pode ser usada como uma plataforma em si. Com uma comunidade grande e ativa de desenvolvedores, o Java é conhecido por sua velocidade, segurança e confiabilidade, e é usado em uma ampla variedade de aplicações, desde aplicações móveis e *software* empresarial até aplicações *big data* e tecnologias do servidor(JAVA, 2022).

2.2.2 Spring boot

O *Spring Boot* é um projeto de *software* livre que oferece uma abordagem simplificada e modular para criar aplicativos Java. O Spring é um *framework* Java criado para facilitar o desenvolvimento de aplicações web, seguindo padrões de projeto de injeção de dependência e inversão de controle (VMWARE, 2023). O *Spring* surgiu em resposta às dificuldades enfrentadas pelos desenvolvedores ao criar aplicações corporativas usando a distribuição J2EE, que, apesar de oferecer muitas opções e ferramentas, tinha algumas limitações que levavam a soluções complexas e pesadas, dependentes de muitas interfaces e configurações (DEVMEDIA, 2022).

O *Spring Boot* utiliza os módulos do Spring, incluindo o *Spring Data*, o *Spring Cloud* e o *Spring Security*, entre outros, para fornecer uma abordagem simplificada e ágil para o desenvolvimento de aplicativos baseados em Spring. Com o *Spring Boot*, é possível criar aplicativos web de forma mais rápida e fácil, pois muitos dos aspectos de configuração são automatizados. Além disso, o Spring Boot é altamente customizável, permitindo que os desenvolvedores escolham os módulos que desejam usar em seus projetos(MICROSOFT, 2023).

2.2.3 Angular

O Angular é um *framework* de código aberto utilizado para desenvolver aplicações voltadas para a web baseadas em SPA (*Single Page Application*). Foi desenvolvido pelos engenheiros da Google para criar interfaces gráficas utilizando principalmente HTML, CSS e TypeScript e lançado em 2012. O *framework* tem a arquitetura MVVM em sua estrutura organizacional, o que significa que possui as camadas Model, View e ViewModel (HOSTINGER, 2023).

Além disso, o Angular oferece diversos recursos para aprimorar a qualidade das aplicações construídas com ele e aumentar a produtividade do desenvolvimento, como criação de componentes, *templates*, diretivas, roteamento, modularização, serviços, injeção de dependências e ferramentas para execução de testes unitários.

O Angular é amplamente aceito pela comunidade de desenvolvimento de *software*

por ser *open source* e por ter um grande apelo comercial, sendo utilizado por grandes empresas e oferecendo muito material para estudo. O *framework* também é compatível com *Desktop* e *Mobile*, pode ser executado na maioria dos navegadores e tem um futuro promissor, com sua popularidade em constante crescimento e a disponibilidade de tutoriais atualizados para ajudar novatos a se familiarizarem com ele (TREINAWEB, 2020).

Devido a esses pontos positivos, o Angular foi escolhido como o *framework* para a construção das interfaces gráficas do projeto. É uma escolha estratégica que oferece vantagens como a combinação de dados bidirecional, diretivas, estrutura do código e ambiente de testes integrados, o que resulta em uma aplicação robusta e escalável.

2.3 Banco de dados

Um banco de dados é um conjunto organizado de dados, armazenado eletronicamente em um sistema de computador. O gerenciamento desses dados é realizado por um Sistema de Gerenciamento de Banco de Dados (SGBD), que inclui o SGBD em si, os dados e os aplicativos associados. Em outras palavras, o banco de dados é a coleção de dados gerenciada pelo SGBD. O objetivo principal do banco de dados é armazenar, recuperar e manipular dados de maneira eficiente e segura (ORACLE, 2022).

2.3.1 PostgreSQL

O PostgreSQL é um banco de dados *open source* com mais de 30 anos de desenvolvimento e mais de 600 desenvolvedores ativos na comunidade. Ele é conhecido por sua confiabilidade, robustez, escalabilidade e bom desempenho, sendo amplamente aceito em serviços de nuvem de infraestrutura como serviço.

Algumas de suas principais características incluem facilidade de acesso, indexação por texto, robustez, capacidade de realizar consultas complexas e suporte ao modelo híbrido objeto-relacional. O PostgreSQL usa a linguagem SQL e pode ser processado em vários sistemas operacionais. Diante desses pontos, o PostgreSQL foi escolhido como banco de dados para armazenar as informações do sistema(POSTGRESQL, 2023).

2.4 DevOps

Nesta seção, serão apresentadas as tecnologias utilizadas para entregar o software de forma eficiente e rápida, integrando práticas de desenvolvimento de *software* (DEV) com operações de infraestrutura e sistemas (Ops). O objetivo é fornecer uma abordagem abrangente para o desenvolvimento de aplicativos, que permita aos desenvolvedores trabalhar de maneira mais colaborativa e eficiente, garantindo a entrega de produtos de alta qualidade em um curto espaço de tempo.

2.4.1 GitHub

O GitHub é uma plataforma de hospedagem em nuvem que permite que os desenvolvedores colaborem em projetos de *software* usando o sistema de controle de versão Git. Ao utilizar o *GitHub*, os desenvolvedores podem fazer mudanças em um projeto e manter seus códigos atualizados, salvando o progresso do desenvolvimento e mantendo um registro detalhado das alterações. Além disso, o *GitHub* permite resgatar versões anteriores do projeto, se necessário (L., 2023).

O *GitHub* é a escolha ideal para o sistema de versionamento do projeto, já que é amplamente aceito e considerado a plataforma de versionamento mais popular atualmente. Além disso, é conhecido por ser um dos repositórios mais seguros disponíveis na atualidade.

2.4.2 Container

Um container é um pacote padronizado contendo um *software*, em conjunto com suas dependências, a configuração, os dados e assim por diante, ou seja, tudo o que ele precisa para executar. Do ponto de vista do sistema operacional, um contêiner representa um processo isolado ou um grupo de processos que existe em seu próprio *namespace*. O funcionamento dos processos internos de um container não pode ser visto por quem está fora dele, e vice-versa. Um contêiner não pode acessar recursos que pertencem a outro contêiner ou processo que está fora dele. Os limites de um contêiner funcionam como uma cerca que impede os processos de saírem executando por aí, usando recursos de terceiros (ARUNDEL, JUL 2019).

2.4.3 Docker

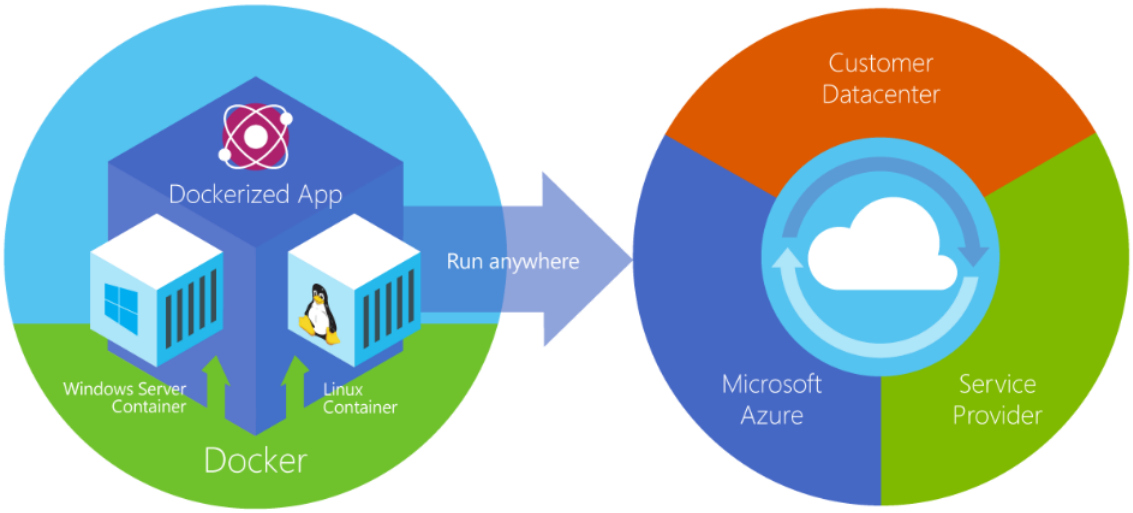
O Docker é um projeto de *software* livre utilizado para automatizar a implantação de aplicações em contêineres autossuficientes portáteis que podem ser executados na nuvem ou localmente. Ele se baseia em funcionalidades do *kernel* do Linux, como cGroups e *namespaces*, para segregar processos e permitir que eles sejam executados de forma independente. Isso ajuda a aumentar a eficiência do uso da infraestrutura e a manter a segurança dos sistemas.

Além disso, o Docker é uma empresa que promove e aprimora essa tecnologia, trabalhando em conjunto com fornecedores de nuvem, do Linux e do Windows (MICROSOFT, 2022).

As ferramentas de container, incluindo o Docker, utilizam um modelo de implantação baseado em imagem, que facilita o compartilhamento de uma aplicação ou conjunto de serviços, juntamente com todas as suas dependências, em diferentes ambientes. Com o Docker, também é possível automatizar a implantação da aplicação usando o ambiente de containers (HAT, 2023).

A Figura 2 apresenta um exemplo de implantação de contêiner em nuvem, onde vários contêineres são executados em uma plataforma de orquestração, como o Kubernetes, que gerencia os contêineres e suas interações.

Figura 2 – Exemplo de implantação de contêiner em nuvem



Fonte: Adaptado de (MICROSOFT, 2022)

2.5 Scrum

Scrum é um método ágil, iterativo e incremental para gerenciamento de projetos. Foi proposto por Jeffrey Sutherland e Ken Schwaber em um artigo publicado pela primeira vez em 1995. Entre todos os métodos ágeis, Scrum é o mais conhecido e amplamente utilizado (VALENTE, 2020).

Uma das características distintivas do *Scrum* é a definição clara de papéis, artefatos e eventos. Esses elementos fornecem uma estrutura para o gerenciamento de projetos e permitem uma comunicação clara entre as partes interessadas. Os papéis definidos no *Scrum* incluem o Dono do Produto, o *Scrum Master* e os Desenvolvedores. Os artefatos incluem o *Backlog* do Produto, o *Backlog* da *Sprint*, o Quadro *Scrum* e o Gráfico de *Burndown*. Os eventos incluem o Planejamento da *Sprint*, a própria *Sprint*, as Reuniões Diárias, a Revisão da *Sprint* e a Retrospectiva.

O *Scrum* oferece uma maneira eficaz e colaborativa de gerenciar projetos, enfatizando a transparência, inspeção e adaptação. Com essa metodologia, equipes podem trabalhar de maneira mais eficiente e produzir resultados de alta qualidade em um ambiente de constante mudança.

3 Metodologia

Neste Capítulo, são explorados os elementos fundamentais do processo de desenvolvimento do software, que foi estruturado em três partes principais: o processo de desenvolvimento, o levantamento de requisitos e a arquitetura do sistema.

Na primeira parte, o processo de desenvolvimento é abordado, enfatizando a gestão do projeto e a organização das tarefas. Serão discutidas as histórias de usuário, a formação do *backlog* de tarefas e a implementação de *sprints* com prazos definidos, elementos essenciais na condução do projeto.

Na segunda parte, o foco é o levantamento de requisitos, tanto funcionais quanto não funcionais. O processo de identificação e organização desses requisitos é fundamental para entender as necessidades do usuário e as funcionalidades que o sistema deve oferecer.

A terceira e última parte trata da arquitetura do sistema, uma peça-chave no desenvolvimento de *software*. Nesta seção, serão apresentadas as decisões de *design* que orientaram a construção do sistema, a organização do código, a interação entre os componentes e as tecnologias utilizadas. Será discutida a importância de uma arquitetura robusta para a escalabilidade, manutenibilidade e evolução do software. Adicionalmente, será mostrado como a infraestrutura do projeto foi gerenciada com o uso da ferramenta *Docker*.

3.1 Processo de Desenvolvimento de Software

A metodologia de desenvolvimento de software adotada neste projeto foi uma abordagem híbrida que combina aspectos de modelos de processo preditivos e ágeis. A intenção foi aproveitar as vantagens de ambos os métodos, permitindo um planejamento estruturado e flexibilidade durante a codificação.

As primeiras fases do projeto seguiram uma estrutura mais preditiva, organizadas em etapas sequenciais claramente definidas. Estas incluíram o levantamento de requisitos, onde as necessidades do sistema foram identificadas e documentadas, e a fase de modelagem, onde foram criados modelos visuais do sistema para orientar a implementação. Esta abordagem permitiu a criação de um plano sólido e detalhado para o desenvolvimento do sistema.

A fase de codificação, por outro lado, foi tratada com um enfoque mais ágil, especificamente utilizando o framework Scrum. Esta fase foi dividida em várias iterações, conhecidas como "sprints", cada uma com duração de duas semanas. Cada sprint começava com uma reunião de planejamento, na qual eram selecionadas as tarefas a serem completa-

das durante a sprint. No final de cada sprint, era feita uma revisão para avaliar o trabalho concluído e planejar a próxima sprint.

Esta combinação de abordagens permitiu que o projeto se beneficiasse do planejamento e da previsibilidade de um modelo de processo preditivo, enquanto mantinha a flexibilidade e a capacidade de resposta a mudanças proporcionada por uma abordagem ágil durante a fase de codificação.

Nas próximas seções, sãos descritos mais detalhadamente cada uma dessas etapas e como elas foram implementadas no contexto deste projeto.

3.1.1 Levantamento de Requisitos

O ponto inicial da análise e modelagem do sistema foi a elaboração de histórias de usuário. As histórias de usuário são descrições concisas e objetivas de um recurso que um usuário necessita para atingir uma meta específica. Estas histórias permitiram delinear as funcionalidades essenciais e as operações que o sistema deve proporcionar para atender às necessidades dos usuários.

As histórias de usuário apresentaram tanto os requisitos funcionais quanto os não funcionais do sistema, seguindo a estrutura padrão de "Como um [usuário], eu quero [funcionalidade], para que eu possa [objetivo]". Este formato mantém o enfoque no usuário final, facilitando a compreensão de suas necessidades de forma mais precisa.

Algumas histórias de usuário elaboradas para abordar as necessidades dos usuários e garantir que a aplicação satisfaça tais necessidades foram:

- Como um atleta, quero me registrar em uma comunidade *online* para ter acesso a informações sobre eventos esportivos e outros atletas.
- Como um atleta, quero poder registrar minhas medidas corporais na plataforma para acompanhar minha evolução ao longo do tempo.
- Como um atleta, quero poder pesquisar locais para praticar meu esporte favorito.
- Como um gerente, quero disponibilizar meu espaço para aluguel em uma plataforma *online* para aumentar a visibilidade e o acesso aos usuários.
- Como um gerente, quero visualizar todas as solicitações de aluguel recebidas para poder avaliar e responder a elas de maneira eficiente.
- Como um gerente, quero poder estipular datas e horários em que o espaço não está disponível para aluguel para que eu possa gerenciar meu negócio de forma adequada.
- Como um gerente, quero poder expor fotos do meu espaço disponível para que os usuários possam visualizar o local antes de fazer uma reserva.

- Como um gerente, quero poder alterar a lista de esportes praticáveis a qualquer momento para que eu possa gerenciar as atividades que acontecem no meu espaço.
- Como um gerente, quero saber qual esporte será praticado previamente no local do aluguel para que eu possa preparar o espaço adequadamente.
- Como um administrador de comunidade, quero poder alugar um ginásio para realizar eventos e atividades esportivas para os membros da comunidade.
- Como um administrador de comunidade, quero poder designar outros administradores para ajudar a gerenciar a comunidade.
- Como um administrador de comunidade, quero poder adicionar ou excluir membros da comunidade para manter a organização e a segurança da plataforma.

Com as histórias de usuário definidas, foi possível estabelecer o *backlog* do projeto, que consiste em uma lista de tarefas ordenadas por prioridade, representando o trabalho a ser realizado no projeto. O *backlog* é orientado pelas necessidades dos usuários, auxiliando na manutenção da organização do projeto e no estabelecimento de objetivos claros.

O *backlog* foi estruturado de forma modular, dividido em backlogs épicos, cada um representando um módulo e contendo histórias associadas a ele. Um *backlog* épico é um grupo de tarefas que compreende um conjunto de histórias relacionadas a uma funcionalidade maior do *software*, permitindo que as equipes de desenvolvimento dividam as funcionalidades em partes menores e gerenciem o trabalho de forma mais eficiente.

Assim, cada *backlog* épico foi dividido em uma lista de histórias de usuário que detalham as funcionalidades específicas a serem implementadas. Isso manteve o foco nas necessidades dos usuários e assegurou que o software fosse desenvolvido de forma iterativa e orientada pelos objetivos do projeto.

Os módulos criados para organizar as histórias de usuário e as funcionalidades a serem desenvolvidas no projeto incluíam:

- Módulo Esportivo: inclui as histórias associadas aos usuários atletas e administradores de comunidade. Engloba as funcionalidades relacionadas ao registro, pesquisa e acompanhamento de medidas corporais, além de permitir a reserva de espaços para a prática de esportes.
- Módulo Administrativo: abrange todas as histórias relacionadas ao gerente do ginásio, incluindo as funcionalidades relacionadas à disponibilização de espaços para aluguel, gerenciamento de solicitações de aluguel, definição de horários disponíveis e exposição de fotos dos espaços.

- Módulo Segurança: engloba todas as histórias associadas à segurança da plataforma. Este módulo aborda as funcionalidades relacionadas ao controle de acesso dos usuários, gestão de autenticação e autorização, além da proteção de dados pessoais.
- Módulo Arquitetura e Documentação: reúne as histórias associadas à documentação e ao estudo de arquitetura e design do software. Ele abrange as funcionalidades relacionadas à criação de diagramas e documentação do código, bem como a pesquisa e adoção de melhores práticas de desenvolvimento.
- Módulo Design: concentra-se no desenvolvimento do protótipo relacionado à UX/UI. Ele envolve as funcionalidades relacionadas à criação de protótipos e componentes que vão ser implementados no sistema.
- Módulo Anexos: este módulo reúne as histórias relacionadas a arquivos e fotos. Ele aborda as funcionalidades relacionadas ao upload e gerenciamento de imagens, documentos e outros arquivos relevantes para a aplicação.

A definição dos módulos permitiu uma organização mais eficaz do *backlog* do projeto, facilitando o gerenciamento das tarefas e garantindo que todos os aspectos da aplicação fossem adequadamente considerados e endereçados.

3.2 Sistemas Relacionados

Esta seção destaca alguns sistemas que compartilham funcionalidades semelhantes e, portanto, podem ser considerados como concorrentes ao projeto proposto:

- *WebQuadras*: Trata-se de uma aplicação voltada para o gerenciamento de horários de espaços esportivos. Contudo, a aplicação é exclusiva para o gerenciamento, o que significa que não permite que usuários externos efetuem a reserva ou aluguel do espaço, como é proposto neste projeto (QUADRAS, 2023).
- *Fintta*: Esta aplicação tem como objetivo servir como ponte entre jogadores e locais para a prática esportiva. No entanto, não apresenta uma comunidade engajada e limita-se apenas a recursos privados (FINTTA, 2023).
- *Peladeiros*: Trata-se de uma aplicação voltada para o gerenciamento de comunidades esportivas, especificamente focada no futebol. Oferece a possibilidade de armazenar informações sobre partidas e gerir os gastos da equipe. No entanto, a última atualização ocorreu em 14 de fevereiro de 2019 (CABRAL, 2023).
- *Appito*: Este é um aplicativo destinado à criação de uma comunidade esportiva gamificada, no qual é possível organizar partidas e solicitar o uso de locais privados.

É o principal concorrente do projeto proposto. Entretanto, como o aplicativo é exclusivamente voltado ao futebol, existem segmentos não explorados em outros esportes que o *Appito* não é capaz de atender (TECNOLóGICAS, 2023).

É importante considerar esses sistemas devido às suas funcionalidades semelhantes às propostas por este projeto. No entanto, cada um deles possui suas próprias limitações e pontos fortes, o que pode ser levado em conta para aprimorar o projeto em questão e torná-lo mais competitivo no mercado.

A Tabela 1 apresenta os requisitos e limitações encontrados nos sistemas relacionados. O objetivo deste trabalho é desenvolver uma ferramenta que solucione essas limitações e atenda aos requisitos identificados, fornecendo assim uma solução mais completa e eficiente.

Requisitos	WebQuadras	Appito	Quattys
Multiplataforma	x		x
Gestão de espaços privados	x	x	x
Gestão de espaços públicos			x
Integração de usuários		x	x
Diversidade esportiva	x		x

Tabela 1 – Requisitos atendidos pelas plataformas e o trabalho

A Tabela 1 apresenta os requisitos atendidos pelas plataformas WebQuadras, Appito e Quattys, bem como o trabalho proposto. Esses requisitos são cruciais para o desenvolvimento de uma solução abrangente e eficiente no gerenciamento de comunidades, agendamentos e espaços.

Em termos de multiplataforma, tanto o WebQuadras quanto o Quattys atendem a esse requisito, permitindo que os usuários acessem as funcionalidades da plataforma por meio de diferentes dispositivos, como smartphones, tablets e computadores. Já o Appito está limitado ao uso exclusivo em dispositivos móveis, restringindo a flexibilidade de acesso.

Quanto à gestão de espaços privados, todas as plataformas demonstram capacidade de atender a essa necessidade. Tanto o WebQuadras quanto o Appito e o Quattys permitem aos usuários gerenciar espaços privados, como quadras esportivas, salas de reunião ou estabelecimentos comerciais.

No que diz respeito à gestão de espaços públicos, apenas o Quattys atende a esse requisito específico, fornecendo recursos para o gerenciamento de espaços públicos, como parques, praças ou ginásios municipais.

A integração de usuários é um requisito atendido pelo Appito e pelo Quattys, permitindo que os usuários se conectem, interajam e compartilhem informações dentro da plataforma. O WebQuadras não oferece esse recurso específico.

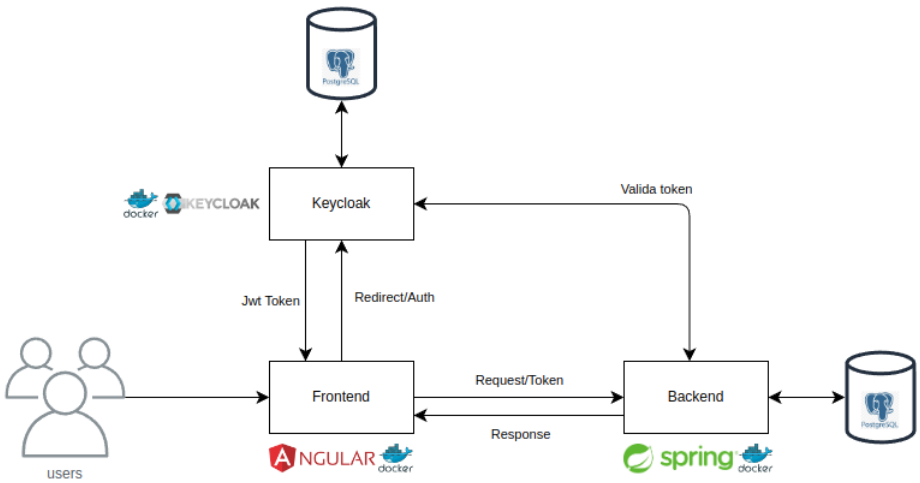
Por fim, no que diz respeito à diversidade esportiva, o sistema WebQuadras oferece a opção de aluguel para a prática de diversas atividades esportivas. Além disso, o Quattys também disponibiliza essa funcionalidade e oferece um sistema de busca para encontrar pessoas interessadas em praticar essas atividades por meio de suas comunidades. Por outro lado, a plataforma Fintta é especializada exclusivamente no aluguel de quadras para a prática de futebol.

Considerando esses requisitos e a análise das plataformas existentes, o trabalho proposto visa desenvolver uma solução que atenda a todos esses requisitos, proporcionando uma experiência multiplataforma, gestão de espaços privados e públicos, integração de usuários e diversidade de atividades. Dessa forma, o projeto pretende preencher lacunas existentes e oferecer uma plataforma abrangente e versátil para o gerenciamento de comunidades e espaços.

3.2.1 Arquitetura do Sistema

A arquitetura do sistema desempenha um papel fundamental em qualquer projeto de desenvolvimento de software. Esta estrutura define a organização geral do sistema, incluindo a disposição dos componentes, a interação entre eles e os padrões de comunicação adotados. A arquitetura funciona como a base do sistema, e sua qualidade influencia diretamente a capacidade do software de cumprir os requisitos de negócio, escalabilidade, manutenibilidade, desempenho e segurança.

Figura 3 – Arquitetura de Comunicação entre os Pacotes



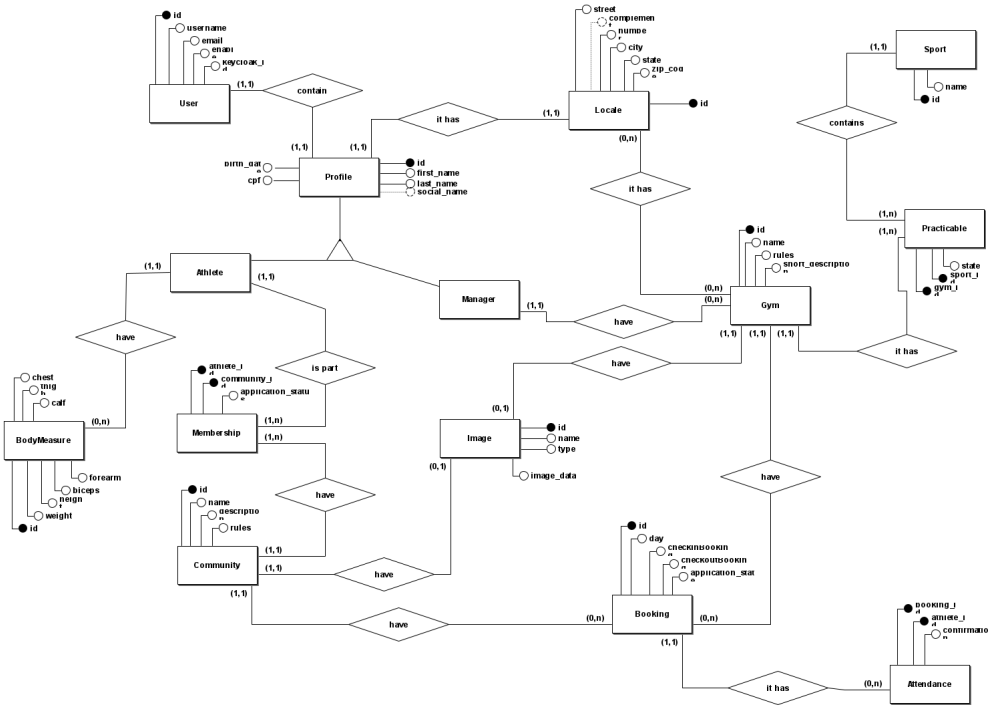
Fonte: Elaborado pelo autor, 2023

A Figura 3 ilustra o diagrama de pacotes, exibindo as diferentes partes integrantes do sistema. Para a realização do registro de usuários, autenticação e autorização de recursos,

foi selecionada a plataforma Keycloak, reconhecida por seu alto nível de segurança e confiabilidade no gerenciamento de identidade e acesso. O frontend foi construído com o auxílio do Angular, um robusto e popular framework JavaScript que possibilita a criação de interfaces de usuário dinâmicas e interativas. O backend, por sua vez, foi desenvolvido com o uso de Spring 3 e Java 17, responsáveis por implementar a lógica de negócios do sistema e processar as solicitações enviadas pelo frontend.

A Figura 4 apresenta o modelo de entidade e relacionamento do projeto Quattys, que ilustra todas as entidades e suas interrelações no sistema. Esse diagrama foi concebido com base nas histórias de usuário e no backlog de tarefas, proporcionando uma representação precisa e detalhada das entidades do sistema.

Figura 4 – Modelo de entidade e relacionamento



Fonte: Elaborado pelo autor, 2023

Para armazenar os dados do sistema, foi escolhido o PostgreSQL, um poderoso e escalável sistema de gerenciamento de banco de dados relacional de código aberto. Além disso, o Docker foi adotado como uma solução para criar containers dos serviços, o que facilita o desenvolvimento e a reprodução precisa dos componentes na nuvem. Mais detalhes sobre o uso do Docker podem ser encontrados no Anexo A.

4 Demonstração

Este Capítulo apresenta uma demonstração prática do uso da aplicação web da Quattys com a finalidade de ilustrar suas principais funcionalidades e fluxos. Para esta demonstração, são usados dados fictícios e imagens das telas da aplicação, desenvolvidas no Figma, com base nos protótipos da interface de usuário. É relevante enfatizar que as informações mostradas durante a demonstração são meramente ilustrativas, não correspondendo a dados reais.

4.0.1 Login, Criação de conta e Recuperação de senha

Ao acessar a aplicação web da Quattys, o usuário é direcionado para a tela de login (Figura 5). Nessa tela, são disponibilizadas três opções ao usuário:

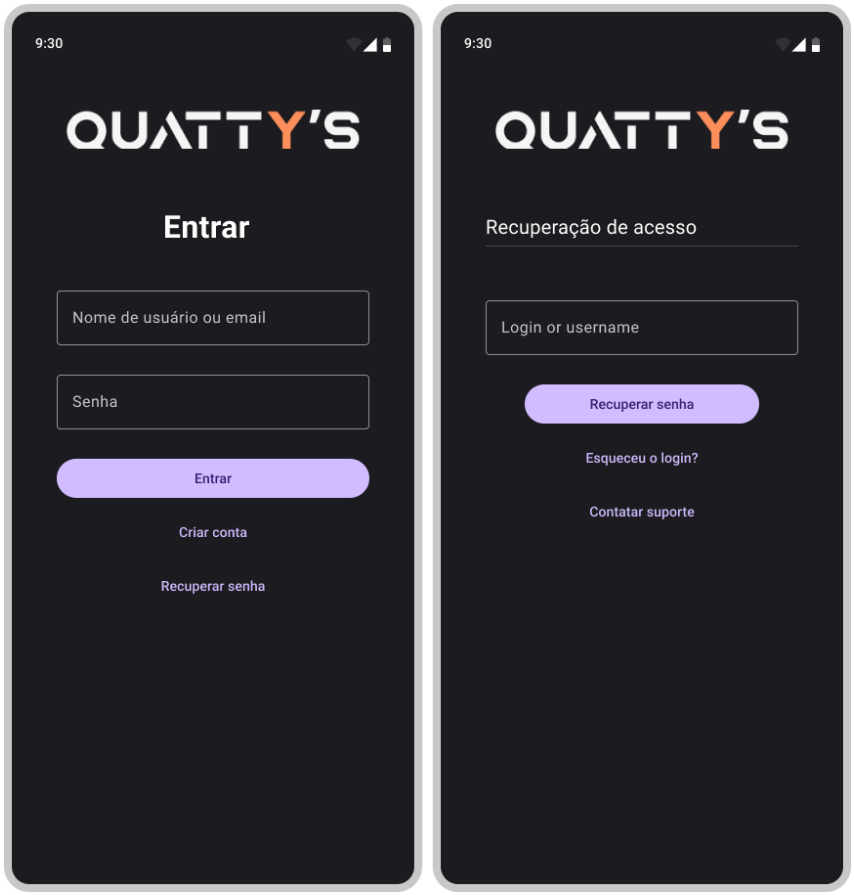
- Autenticação: Aqui, é necessário que o usuário forneça seu e-mail ou nome de usuário, juntamente com a senha previamente cadastrada. Estas informações garantem o acesso aos recursos adicionais disponíveis na aplicação.
- Criação de conta: Esta opção é destinada aos usuários que ainda não possuem uma conta na plataforma e os direciona para um processo de cadastro. Neste procedimento, são solicitadas as informações necessárias para utilizar a aplicação, incluindo dados pessoais relevantes.
- Recuperação de senha: Caso o usuário esqueça sua senha, basta selecionar esta opção e informar seu login ou nome de usuário. Será enviado um e-mail ao proprietário da conta com as instruções necessárias para redefinir a senha.

A Figura 5 ilustra a tela inicial da aplicação web da Quattys, apresentando visualmente as opções de autenticação, criação de conta e recuperação de acessos. Estas funcionalidades oferecem uma experiência segura e personalizada aos usuários, permitindo a escolha da opção mais adequada às suas necessidades e o avanço no uso da plataforma com facilidade.

Caso o usuário ainda não possua uma conta registrada, é necessário criar uma nova conta. Para isso, basta clicar no botão "Criar conta"na tela de login (Figura 7), o que direciona o usuário para um fluxo de criação de conta em três etapas:

1. Informações de registro: Nesta primeira etapa, o usuário precisa fornecer informações essenciais para o uso da aplicação, como nome de usuário, email e senha. Também é solicitado que selecione o tipo de usuário que melhor se encaixa em seu perfil.

Figura 5 – Tela de Login, criação de conta e recuperação de acessos da aplicação web da Quattys



Fonte: Elaborado pelo autor, 2023

2. Informações pessoais: Após preencher as informações de registro, o usuário é direcionado para a inserção de suas informações pessoais. Isso inclui dados como nome completo, data de nascimento, CPF, endereço e outras informações relevantes.

3. Conclusão do cadastro: Após fornecer todas as informações necessárias, o usuário pode revisar e confirmar seus dados. Em seguida, pode finalizar o processo de criação da conta e ter acesso completo aos recursos da aplicação.

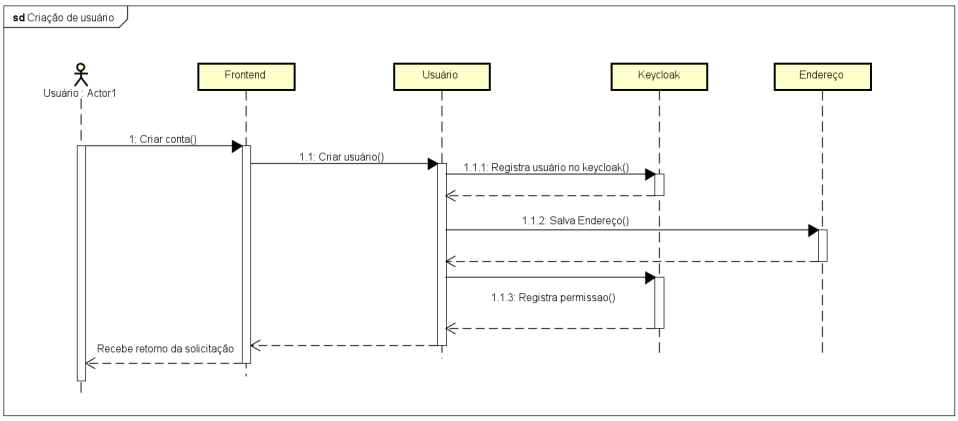
Para garantir a segurança das informações dos usuários, a aplicação utiliza o sistema Keycloak para armazenar e gerenciar informações sensíveis, como senhas. Isso exige que o *backend* da aplicação realize chamadas a um serviço externo.

A Figura 6 ilustra o processo de registro do usuário na aplicação, seguindo as etapas a seguir, o que garante a validação e a integridade dos dados fornecidos:

1. O usuário realiza uma solicitação de registro na aplicação.

2. A aplicação chama o endpoint de usuário e, em seguida, realiza uma chamada ao

Figura 6 – Diagrama de Sequência - Criação de usuário



Fonte: Elaborado pelo autor, 2023

Keycloak para salvar as informações de acesso, como email, primeiro nome, último nome, senha e nome de usuário.

3. Verificada a ausência de erros, a aplicação salva o endereço informado pelo usuário no banco de dados, juntamente com as informações de perfil e usuário.

4. Essas informações são armazenadas no banco de dados da aplicação para permitir o rastreamento de alterações feitas pelo usuário e para estabelecer relacionamentos com outras entidades.

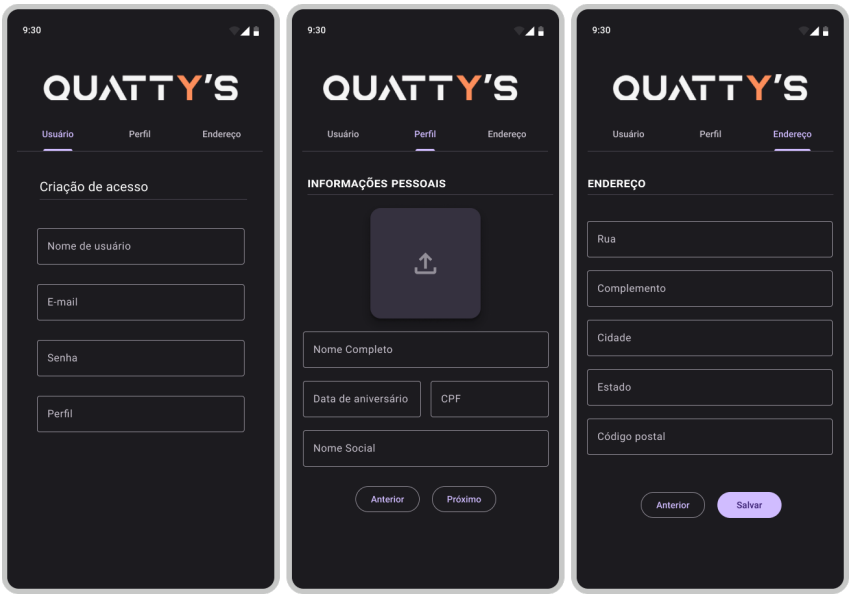
5. Por fim, de acordo com o tipo de perfil escolhido pelo usuário, são feitas edições no Keycloak para adicionar as novas características.

Durante cada etapa do processo, são aplicadas validações para garantir a integridade dos dados, como evitar a repetição de nome de usuário ou email, além de garantir que os campos de endereço não sejam deixados em branco. Caso o formulário não atenda a essas validações, os dados não são salvos e é retornado um erro ao usuário.

Além disso, todas essas validações também são implementadas no frontend, proporcionando alertas em tempo real ao usuário e melhorando sua experiência com a aplicação. Dessa forma, garantimos a consistência e a qualidade dos dados registrados. A Figura 7 apresenta o fluxo via interface gráfica que o usuário utilizará para registrar suas informações.

No caso de um usuário não se recordar de suas credenciais de acesso, é possível recuperá-las. Para isso, é necessário clicar no botão "Recuperar Senha", localizado na tela de login (Figura 5). O usuário é então redirecionado para a tela de recuperação de senha (Figura 5), onde deve informar seu e-mail ou nome de usuário. O sistema envia instruções para o endereço de e-mail do usuário para a recuperação de suas credenciais de acesso.

Figura 7 – Tela de criação de usuário da aplicação Quattys

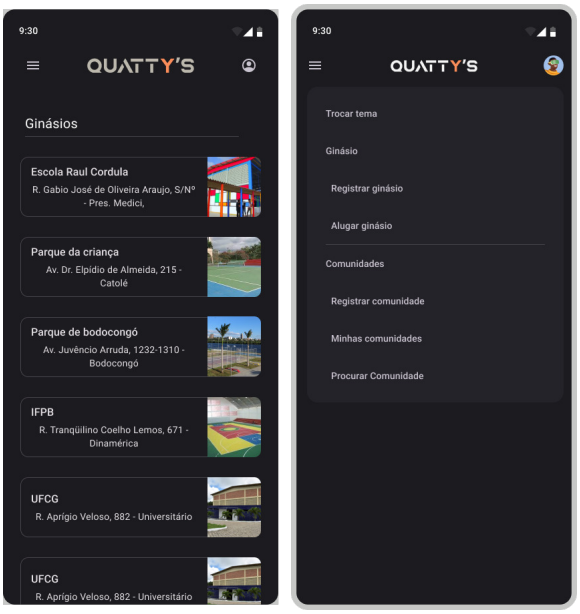


Fonte: Elaborado pelo autor, 2023

4.0.2 Tela Principal de Navegação

Depois de autenticado na tela de login (Figura 5), o usuário é encaminhado para a tela inicial do aplicativo (Figura 8). Esta interface fornece acesso a uma variedade de funcionalidades e informações pertinentes.

Figura 8 – Interface inicial e Menu



Fonte: Elaborado pelo autor, 2023

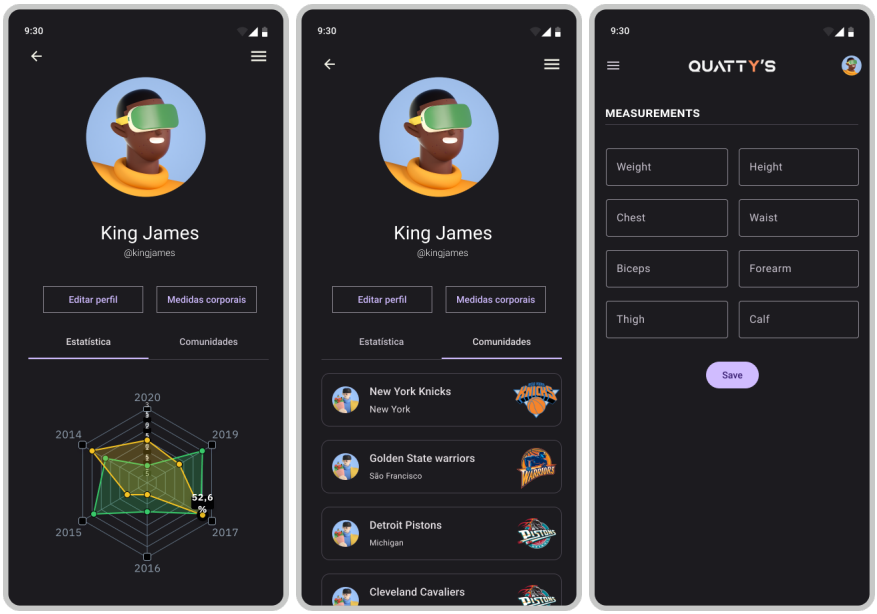
A tela inicial exibe, por padrão, os ginásios cadastrados e disponíveis para aluguel. No canto superior esquerdo, há um ícone de menu, representado por três linhas paralelas,

que desdobra uma lista de opções ao ser selecionado. As opções de menu, ilustradas na Figura 8, variam conforme o perfil do usuário, que pode ser de gerente ou atleta.

Usuários com perfil de gerente têm opções para registro, edição e visualização de solicitações de ginásios. Já para o perfil de atleta, estão disponíveis opções para alugar ginásios, criar comunidades, visualizar as comunidades em que participa e uma ferramenta de busca para encontrar outras comunidades. Esta personalização do menu, com base no perfil do usuário, permite uma experiência mais focada e adaptada às necessidades individuais dentro do aplicativo.

Adicionalmente, no canto superior direito da tela inicial, há um ícone que pode ser a foto do usuário. Ao clicar neste ícone, o usuário é redirecionado para a aba de perfil pessoal.

Figura 9 – Interface de perfil do atleta



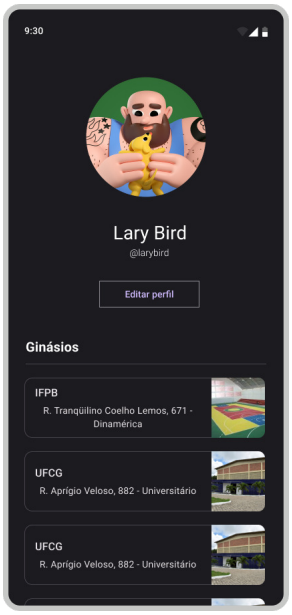
Fonte: Elaborado pelo autor, 2023

A Figura 9 apresenta a interface do perfil de um atleta, ressaltando as informações e recursos únicos disponíveis para este perfil. Nesta página, o atleta pode acessar suas estatísticas, incluindo medidas corporais registradas ao longo do tempo, visualizar todas as comunidades das quais faz parte e tem a opção de editar seu perfil, exceto o nome de usuário e o e-mail.

A Figura 10 demonstra o perfil de um gerente, listando todos os ginásios sob sua administração. Esta visão é exclusiva para os gerentes, permitindo-lhes gerenciar de maneira eficaz todos os ginásios que administram.

Ao acessar a página de perfil como gerente, além da listagem de ginásios, o usuário também pode acessar outras funcionalidades e opções específicas para seu papel. Ao

Figura 10 – Interface de perfil do gerente



Fonte: Elaborado pelo autor, 2023

selecionar um ginásio listado, o gerente é direcionado para o menu administrativo, que inclui uma visão detalhada do ginásio, visualização das solicitações recebidas e gerenciamento de reservas.

4.0.3 Registro de Ginásios

Uma das funcionalidades que se destaca para o gerente é o cadastro de ginásios. O processo de criação, ilustrado na Figura 11, ocorre em três etapas.

Durante a primeira etapa, o gerente insere informações como o nome, uma breve descrição, horários e dias de funcionamento do ginásio, bem como regras específicas do local. O gerente pode ainda selecionar os esportes praticáveis no ginásio a partir de uma lista disponível na interface.

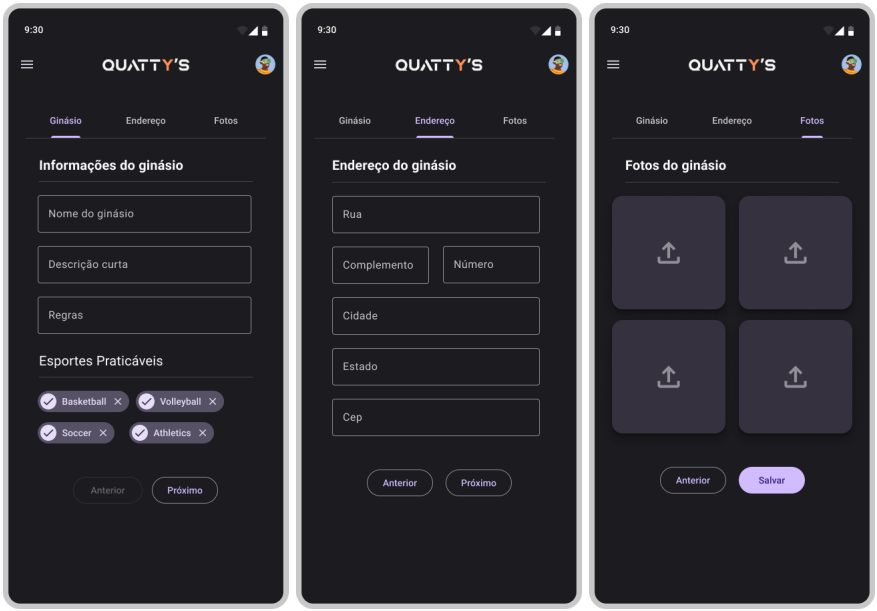
Na segunda etapa, são solicitadas informações sobre o endereço do ginásio, incluindo rua, número, bairro, cidade e CEP. Estes detalhes facilitam a localização do ginásio pelos usuários e permitem o planejamento de visitas.

Finalmente, na terceira etapa, o gerente pode adicionar fotos do ginásio para fornecer uma apresentação visual atraente. As fotos podem evidenciar as instalações, equipamentos disponíveis e a atmosfera do ginásio, fornecendo aos usuários uma visão mais completa antes de optarem pela locação.

Este processo estruturado de registro em três etapas assegura que todas as informações essenciais sejam fornecidas de maneira organizada, tornando o ginásio atraente e confiável para os usuários do aplicativo. A Figura 11 apresenta o fluxo deste processo,

orientando o gerente durante o registro do ginásio.

Figura 11 – Processo de criação de ginásio



Fonte: Elaborado pelo autor, 2023

Após a conclusão do registro, o ginásio fica disponível para locação pelos usuários. Na Figura 12, é exibida a interface para a qual o usuário é direcionado ao selecionar qualquer ginásio na página inicial (Figura 8).

Figura 12 – Interface de reserva do ginásio

Fonte: Elaborado pelo autor, 2023

Nesta interface, o usuário pode acessar todas as informações relevantes do ginásio, incluindo fotos da infraestrutura, horário de funcionamento, endereço, agendamentos

confirmados e regras de uso. Tais informações são fundamentais para que o usuário faça uma escolha informada e alugue o ginásio que atende às suas necessidades.

Através desta interface, o usuário pode solicitar um agendamento clicando no botão localizado no final da página. Ao clicar nesse botão, o usuário é redirecionado para a primeira etapa do agendamento, onde deve concordar com as regras estabelecidas pelo ginásio. Depois de confirmar seu acordo, o usuário pode escolher a comunidade para a qual deseja fazer o agendamento e selecionar os horários disponíveis.

Esse processo de agendamento garante que os usuários cumpram as regras estabelecidas pelo ginásio e possam realizar suas reservas de maneira rápida e eficiente. A Figura 12 oferece uma visão clara desta interface, guiando o usuário durante o processo de reserva do ginásio.

4.0.4 Registro de Comunidades

A aplicação amplia sua funcionalidade para além do aluguel de espaços fechados, oferecendo aos usuários a capacidade de estabelecer e fazer parte de comunidades. Esta funcionalidade fomenta a interação entre usuários com interesses e atividades esportivas similares.

Figura 13 – Criação de nova comunidade

Fonte: Elaborado pelo autor, 2023

Usuários com perfil de atleta possuem a autonomia para formar uma comunidade. A Figura 13 demonstra o formulário necessário para a criação de uma nova comunidade. Este formulário requisita detalhes como uma imagem representativa da comunidade, o

nome da comunidade, uma descrição das atividades planejadas pela comunidade e as diretrizes para a participação nela.

A formação de uma comunidade permite ao usuário estabelecer um ambiente virtual para reunir pessoas com interesses semelhantes. Isso facilita a organização de eventos, competições, treinamentos em grupo e outras atividades relacionadas ao esporte praticado por essa comunidade.

Além da criação de comunidades, o aplicativo proporciona aos usuários a chance de participar de comunidades existentes. Isso favorece aqueles que preferem não gerenciar uma comunidade, permitindo que se associem a grupos que compartilham de atividades esportivas em comum.

A Figura 14 ilustra a interface que apresenta todas as comunidades cadastradas. Nesta interface, os usuários podem explorar as opções e descobrir comunidades que se alinhem aos seus interesses. Caso o usuário tenha uma comunidade específica em mente, ele pode realizar uma busca pelo nome da comunidade para encontrá-la e solicitar a adesão a ela.

Figura 14 – Listagem das comunidades

Fonte: Elaborado pelo autor, 2023

Essa funcionalidade facilita o encontro entre pessoas com interesses esportivos semelhantes, possibilitando que os usuários localizem e se associem a comunidades específicas que atendam às suas preferências e metas. Participar de uma comunidade oferece a chance de se engajar em atividades esportivas em grupo, compartilhar conhecimentos e experiências e expandir sua rede social no contexto esportivo.

A interface de listagem de comunidades e a função de busca facilitam aos usuários

a exploração de diversas opções e a descoberta da comunidade mais adequada às suas necessidades e interesses.

Caso um usuário se interesse por uma comunidade específica, ao selecionar um dos cartões de listagem disponíveis na Figura 14, ele será redirecionado para a tela do Perfil da Comunidade. Esta tela fornecerá detalhes extensivos sobre a comunidade e suas atividades.

Figura 15 – Perfil da comunidade

Fonte: Elaborado pelo autor, 2023

No Perfil da Comunidade, os usuários podem acessar diversas informações relevantes, incluindo a lista de membros da comunidade e os diferentes cargos existentes. Além disso, aqueles interessados em se associar à comunidade podem se candidatar, estando sujeitos à aprovação de um administrador ou moderador.

Para aqueles já aceitos na comunidade, o Perfil da Comunidade disponibiliza recursos adicionais. Eles podem acompanhar encontros programados, mantendo-se atualizados sobre as atividades e eventos planejados. Se o usuário tiver um perfil de administrador ou moderador, ele terá a capacidade de gerenciar as informações da comunidade, incluindo a adição ou remoção de membros.

A interface do Perfil da Comunidade foi projetada para fornecer todas as informações necessárias sobre uma comunidade específica. Ela facilita a interação entre os membros, permitindo que se engajem nas atividades da comunidade, solicitem adesão e estejam informados sobre os eventos planejados.

O acesso a essas informações é crucial para que os usuários possam tomar decisões informadas sobre sua participação nas comunidades e se engajar de forma significativa

com outras pessoas que compartilham interesses esportivos semelhantes. Esta interação fortalece a comunidade esportiva e oferece uma experiência enriquecedora para todos os participantes.

5 Considerações Finais e Trabalhos Futuros

Este trabalho, disponível para consulta no *GitHub*[1], centrou-se na apresentação do processo de desenvolvimento do Quattys, um sistema Web projetado para aprimorar a interação entre atletas e locais apropriados para a prática de esportes.

Durante as etapas de avaliação de mercado e definição de requisitos, foi identificada a necessidade de uma plataforma que simplificasse a localização e conexão entre atletas. Embora já existam plataformas com propósitos similares, muitas são segmentadas, atendendo a um esporte específico ou apresentando funcionalidades restritas à sua comunidade. Diante dessa demanda, o Quattys foi concebido com o intuito de fomentar comunidades para aqueles que desejam praticar esportes, mas encontram barreiras para localizar indivíduos com interesses similares ou espaços adequados para a prática.

O desenvolvimento do Quattys teve como foco primordial promover a qualidade de vida da população, ao proporcionar uma solução inovadora para a localização de parceiros esportivos e espaços adequados para a prática esportiva. Ademais, o Quattys objetiva fornecer uma oportunidade para gerentes de instalações esportivas, tanto públicas quanto privadas, gerenciarem seus horários de maneira eficaz, otimizando a utilização de suas instalações.

Os resultados alcançados no desenvolvimento do protótipo englobam uma série de funcionalidades cruciais. O Quattys permite o registro, listagem e edição de ginásios e comunidades, além de possibilitar a solicitação de aluguel de um ginásio e o registro de usuários em uma ou mais comunidades. Os usuários também podem visualizar os agendamentos das comunidades das quais fazem parte, perfis individuais de gestores de ginásios e atletas, e usufruir da aplicação em dispositivos móveis, com a opção de temas claros e escuros para melhorar a experiência do usuário.

Com vistas a aprimoramentos futuros, vários aspectos do Quattys estão planejados para desenvolvimento e implementação:

- **Lista de Presença:** Este recurso permitirá que os usuários confirmem sua presença em um ginásio reservado. Isso ajudará os gestores dos ginásios a acompanhar quem realmente utilizou o espaço, auxiliando na gestão do uso efetivo do local e na manutenção da segurança.
- **Mecanismos de Pagamento:** A intenção é incorporar uma funcionalidade que permita aos usuários realizar o pagamento pelo aluguel de ginásios privados diretamente através da plataforma. Isso simplificará o processo de aluguel e aumentará a

[1] https://github.com/Erickson-Eng/TCC

eficiência e a conveniência para ambas as partes envolvidas, os usuários e os gestores de ginásios.

- **Painel de Gestão para Ginásios Privados:** Este recurso proporcionará aos gestores de ginásios privados uma visão geral e a capacidade de gerenciar a utilização de seus espaços, incluindo o agendamento, a confirmação de pagamento, a manutenção e outros aspectos relevantes. Isso oferecerá aos gestores uma ferramenta robusta para administrar eficientemente suas instalações.

- **Mecanismos de Validação:** Estão previstos mecanismos de validação para evitar fraudes, como a inclusão de ginásios inexistentes ou a representação falsa de um gestor de ginásio. Isso poderia incluir a necessidade de verificação de identidade ou validação do local antes que um ginásio possa ser listado na plataforma. Isso garantirá a segurança e a confiabilidade dos ginásios listados na plataforma Quattys.

Além dessas funcionalidades, também foram elaborados outros documentos e artefatos relevantes para auxiliar na compreensão e desenvolvimento futuro do projeto. Estes incluem documentos UML, que fornecem uma representação visual das estruturas do sistema, modelo de entidade relacionamento do banco de dados, que descreve como os dados serão armazenados e gerenciados, e um *protótipo UX/UI*[2], que oferece uma visão preliminar do design e funcionalidade do sistema. Esses recursos auxiliarão a equipe de desenvolvimento na orientação e implementação das melhorias planejadas.

Em conclusão, o Quattys será hospedado na nuvem, o que permitirá o acesso remoto dos usuários, garantindo maior acessibilidade e segurança aos dados

[2] https://www.figma.com/file/LNMmZNoka4iAjUFMJ6H6S9/Quattys?type=design&node-id=479%3A7522&t=ufn59l7TtN8s8Gut-1

Referências

ARUNDEL, J. D. J. **DevOps Nativo de Nuvem com Kubernetes: Como Construir, Implantar e Escalar Aplicações Modernas na Nuvem**. [S.l.]: Novatec Editora, JUL 2019. ISBN 8575227785. Citado na página 22.

CABRAL, W. 2023. Disponível em: https://play.google.com/store/apps/details?id=br.com.peladeiros&hl=pt_BR&gl=US&pli=1. Acesso em: 10 Abr 2023. Citado na página 27.

DEVMEDIA. **Como começar com Spring?** 2022. Disponível em: https://www.devmedia.com.br/exemplo/como-comecar-com-spring/73. Acesso em: 4 Dez 2022. Citado na página 20.

FINTTA. **ORGANIZE, RESERVE E RACHE SUAS PARTIDAS!** 2023. Disponível em: https://fintta.com/. Acesso em: 10 Abr 2023. Citado na página 27.

HAT, R. **API REST**. 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 10 Abr 2023. Citado 2 vezes nas páginas 18 e 19.

HAT, R. **O que é o Docker?** 2023. Disponível em: https://www.redhat.com/pt-br/topics/containers/what-is-docker. Acesso em: 5 Fev 2023. Citado na página 22.

HOSTINGER. **O Que é Angular? Guia para Iniciantes**. 2023. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-angular. Acesso em: 25 Fev 2023. Citado na página 20.

JAVA. **O que é tecnologia Java e por que preciso dela?** 2022. Disponível em: https://www.java.com/pt-BR/download/help/whatis_java.html. Acesso em: 4 Dez 2022. Citado na página 20.

KEYCLOAK. **Documentation**. 2023. Disponível em: https://www.keycloak.org/documentation. Acesso em: 11 Mai 2023. Citado na página 19.

L., A. **O Que é GitHub e Como Usá-lo**. 2023. Disponível em: www.hostinger.com.br/tutoriais/o-que-github. Acesso em: 10 Abr 2023. Citado na página 22.

MICROSOFT. **O que é o Docker?** 2022. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/container-docker-introduction/docker-defined. Acesso em: 5 Dez 2022. Citado 2 vezes nas páginas 22 e 23.

MICROSOFT. **O que é o Java Spring Boot?** 2023. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-java-spring-boot/. Acesso em: 10 Abr 2023. Citado na página 20.

ORACLE. **O que é um Banco de Dados?** 2022. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 5 Dez 2022. Citado na página 21.

PIMENTA, J. S. R. **Preferência e Prática Físico-Esportiva em Escolares do Ensino Fundamental da Cidade de Ouro Preto**. 2015. Disponível em: https://www.monografias.ufop.br/handle/35400000/80. Acesso em: 09 Feb 2023. Citado na página 14.

POSTGRESQL. **Documentation**. 2023. Disponível em: https://www.postgresql.org/files/documentation/pdf/14/postgresql-14-A4.pdf. Acesso em: 10 Abr 2023. Citado na página 21.

QUADRAS, W. **RESERVE SUA QUADRA ONLINE!** 2023. Disponível em: https://www.webquadras.com.br/quadras/saopaulo. Acesso em: 10 Abr 2023. Citado na página 27.

TECNOLóGICAS, A. S. **O futuro do futebol**. 2023. Disponível em: https://play.google.com/store/apps/details?id=com.apitador.app. Acesso em: 10 Abr 2023. Citado na página 28.

TREINAWEB. **O que é o Angular e para que serve?** . 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-angular-e-para-que-serve. Acesso em: 4 Dez 2022. Citado na página 21.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [S.l.]: Editora: Independente, 395 páginas, 2020. ISBN 978-65-00-01950-6. Citado 3 vezes nas páginas 17, 18 e 23.

VECCHIOLI, D. **Quase metade das escolas brasileiras não tem local para prática de esporte**. 2021. Disponível em: https://www.uol.com.br/esporte/colunas/olhar-olimpico/2021/12/14/quase-metade-das-escolas-brasileiras-nao-tem-local-para-praticar-esporte.html. Acesso em: 09 Feb 2023. Citado na página 14.

VMWARE. **Spring Boot**. 2023. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 10 Abr 2023. Citado na página 20.

Apêndices

APÊNDICE A – Utilização do Docker

O Docker desempenha um papel essencial neste projeto, pois permite a automação da criação de um ambiente único e isolado para a aplicação. Com o Docker, é possível gerar de forma eficaz e rápida imagens dos serviços backend, Keycloak, e seus respectivos bancos de dados. Tais imagens podem ser facilmente distribuídas e implantadas em ambientes distintos, sem preocupações relativas a diferenças entre sistemas operacionais ou configurações de máquina. Isso assegura a consistência do ambiente e minimiza a possibilidade de erros e incompatibilidades.

O Dockerfile é um arquivo utilizado para criar uma imagem Docker, que é um pacote contendo todas as dependências e configurações necessárias para montar e executar o backend da aplicação.

O Código A.1 exemplifica o Dockerfile utilizado na criação da Docker Image da aplicação Quattys. As instruções utilizadas no Dockerfile incluem:

1. *FROM*: Nome e versão da Docker Image do Maven utilizada como base para compilar a aplicação;
2. *WORKDIR*: Define o diretório de trabalho dentro do container. Caso ele não exista, é automaticamente criado durante a execução do Dockerfile;
3. *COPY*: Copia arquivos e pastas do host para dentro do container, no diretório especificado na instrução do *WORKDIR*;
4. *RUN*: Executa o comando maven dentro do container responsável por compilar e empacotar a aplicação em um arquivo .jar;
5. *FROM*: Nome e versão da Docker Image do Java usada como base para executar a aplicação java compilada pelo Maven no comando *RUN*;
6. *COPY*: Copia os arquivos e pastas da primeira imagem para a atual;
7. *ENTRYPOINT*: Especifica o comando a ser executado quando o container é iniciado.

Código A.1 – Dockerfile do backend

```
1  FROM maven:3.8.7-amazoncorretto-17 as build
2  WORKDIR /app
3  COPY . .
4  RUN mvn clean package -DskipTests
5
6  FROM amazoncorretto:17-al2-full
```

```
7  WORKDIR /app
8  COPY --from=build ./app/target/*.jar backend.jar
9  ENTRYPOINT ["java", "-jar" ,"backend.jar"]
```

No entanto, o ambiente engloba diversos componentes, como Keycloak, frontend, e os bancos de dados do Keycloak e do backend, que não estão contemplados na estrutura do Dockerfile. Assim, é necessário utilizar o Docker Compose, uma ferramenta que permite montar todo o ambiente, incluindo imagens, containers, e sincronizar os elementos de acordo com a ordem de precedência. Ao executar o comando *docker-compose up* na raiz do projeto backend, o Docker Compose é acionado para realizar o build das imagens e instanciar os containers necessários para executar a aplicação. Dessa forma, a montagem do ambiente torna-se mais simples e eficiente, permitindo a execução da aplicação em diferentes ambientes de forma consistente e confiável.

O Código A.2 apresenta o arquivo docker-compose usado para montar os containers relacionados ao backend. Há, no total, quatro serviços sendo instanciados: *backend*, db_backend, Keycloak. As instruções contidas no arquivo realizam as seguintes ações:

1. *version*: Especifica a versão do formato de arquivo do Docker Compose em uso.
2. *volumes*: Nomeia um volume (postgres_data) que será usado para armazenar os dados do banco de dados PostgreSQL utilizado pelo serviço backend.
3. *backend*: Define um contêiner para executar a aplicação backend.
4. *image*: Especifica a versão da imagem que será utilizada para criar o contêiner.
5. *build*: Especifica o diretório onde o Dockerfile está localizado.
6. *ports*: Realiza o mapeamento da rede, onde o primeiro argumento é a porta externa e o segundo a interna, separados por dois pontos.
7. *db_backend*: Define um contêiner para executar o banco de dados PostgreSQL.
8. *image*: Especifica a versão da imagem que será utilizada para criar o contêiner, neste caso, o postgres 14.
9. *restart*: Define quando o contêiner deve ser reiniciado.
10. *ports*: Realiza o mapeamento da rede.

Código A.2 – Arquivo docker-compose do backend Quattys

```
1  version: '3.7'
2  volumes:
3    postgres_data:
4      driver: local
```

```
5
6  services:
7    backend:
8      image: backend:latest
9      build:
10       context: .
11       dockerfile: Dockerfile
12     ports:
13       - "8888:8888"
14     depends_on:
15       - keycloak
16       - db_backend
17     environment:
18       - SPRING_DATASOURCE_URL=jdbc:postgresql://db_backend:5432/quattys?currentSchema=public
19       - SPRING_DATASOURCE_USERNAME=postgres
20       - SPRING_DATASOURCE_PASSWORD=postgres
21       - KEYCLOAK_URL=keycloak
22
23   db_backend:
24     image: postgres:14
25     container_name: db_backend
26     restart: on-failure
27     ports:
28       - "5432:5432"
29     environment:
30       - POSTGRES_PASSWORD=postgres
31       - POSTGRES_USER=postgres
32       - POSTGRES_DB=quattys
33
34   postgres:
35     image: postgres:10
36     container_name: db_keycloak
37     ports:
38       - "5433:5432"
39     restart: always
40     volumes:
41       - postgres_data:/var/lib/postgresql/data
42     environment:
43       - POSTGRES_USER=keycloak
44       - POSTGRES_PASSWORD=password
45       - POSTGRES_DB=keycloak
46
47   keycloak:
48     image: quay.io/keycloak/keycloak:20.0.3
49     command:
50       - start-dev
51     environment:
52       DB_VENDOR: POSTGRES
53       DB_ADDR: postgres
54       POSTGRES_DB: keycloak
55       POSTGRES_USER: keycloak
56       POSTGRES_PASSWORD: password
57       KEYCLOAK_ADMIN: admin
58       KEYCLOAK_ADMIN_PASSWORD: admin
59     ports:
60       - "8080:8080"
61     depends_on:
62       - postgres
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Versão Final do TCC

Assunto: Versão Final do TCC
Assinado por: Erickson Tulio
Tipo do Documento: Tese
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Erickson Tulio Rodrigues Azevedo**, **ALUNO (201721250018) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 22/06/2023 17:53:01.

Este documento foi armazenado no SUAP em 22/06/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 861828
Código de Autenticação: 8bd9547ef8