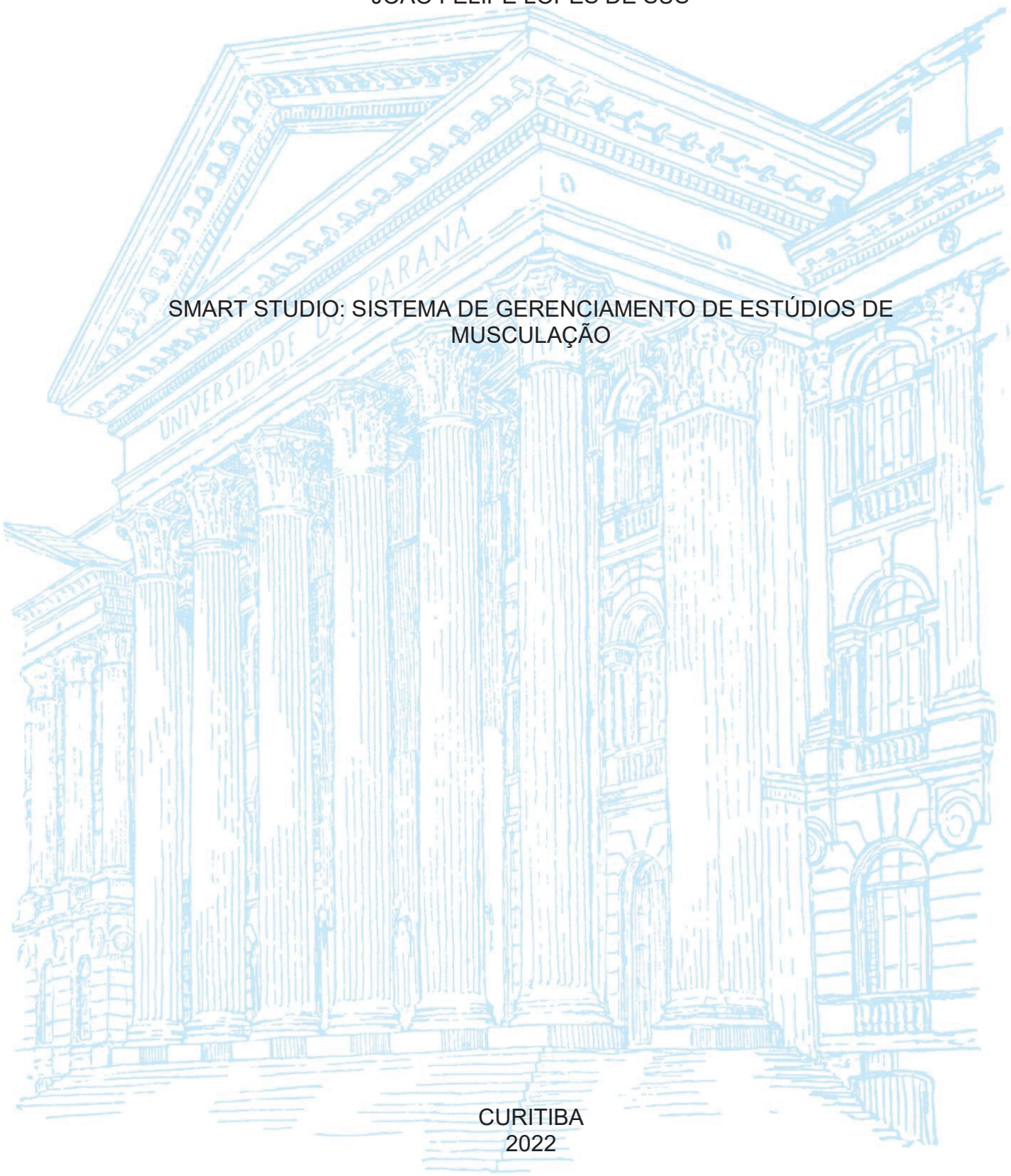
UNIVERSIDADE FEDERAL DO PARANÁ

JOÃO FELIPE LOPES DE SUS



SMART STUDIO: SISTEMA DE GERENCIAMENTO DE ESTÚDIOS DE MUSCULAÇÃO

CURITIBA
2022

JOÃO FELIPE LOPES DE SUS

SMART STUDIO: SISTEMA DE GERENCIAMENTO DE ESTUDIOS DE MUSCULAÇÃO

Monografia apresentada ao curso de Pós-graduação em Desenvolvimento Ágil de Software, Setor de Educação Profissional e Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Desenvolvimento Ágil de Software.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITBA
2022

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO DESENVOLVIMENTO ÁGIL DE SOFTWARE - 40001016375E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação DESENVOLVIMENTO ÁGIL DE SOFTWARE da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **JOÃO FELIPE LOPES DE SUS** intitulada: **SMART STUDIO: SISTEMA DE GERENCIAMENTO DE ESTUDIOS DE MUSCULACAO**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 09 de Dezembro de 2022.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Rua Doutor Alcides Vieira Arco-verde - Curitiba - Paraná - Brasil
CEP 81520260 - Tel: (41) 9887-40472 - E-mail: jaimewo@ufpr.br

Aos meus pais, avós e Dafne o meu amor por todo o apoio

AGRADECIMENTOS

Agradeço a Dafne por ter me apoiado durante toda esta jornada, também a meus pais e avós por sempre incentivarem a buscar novos conhecimentos.

Agradeço também aos professores da Especialização por todo o conhecimento transmitido e o exemplo dado, em especial ao Dr. Razer Anthom Nizer Rojas Montaño. Também, aos colegas do curso pela troca constante de conhecimentos, em especial, André, Giovanny, Hiago e Vanessa.

Se eu vi mais longe, foi por estar sobre ombro de gigantes.
(ISAAC NEWTOW)

RESUMO

Em um contexto em que profissionais de educação física notaram à demanda pela prestação de um serviço personalizado, houve um crescimento no número da abertura de estúdios de musculação uma vez que seu modelo de negócio trabalha com um baixo número de alunos, porém com rotinas de treino e acompanhamento feitos de forma mais próxima ao aluno buscando a todo momento adaptar rotinas de treino para que o objetivo do aluno seja alcançado. Em contraponto a grandes redes de academias em que seu modelo de negócio busca ter o máximo de alunos possível, o que em alguns casos pode diminuir o tempo de dedicação de profissionais à alunos. Com o número reduzido de profissionais em estúdios de musculação, que em muitos casos pode ser apenas um profissional para gerenciar alunos, planejar e aplicar treinos bem como outros compromissos relacionados à parte financeira como o controle de mensalidades, pode sobrecarregar o profissional durante o gerenciamento do estúdio. Com isso a existência de uma plataforma que simplifique a execução destas atividades pode servir de grande ajuda na rotina destes profissionais. Neste trabalho foi desenvolvido um software que busca simplificar as rotinas envolvidas no gerenciamento de estúdios de musculação, onde foram aplicados métodos ágeis de desenvolvimento ao longo do projeto. Foi utilizada uma arquitetura web, com um *backend* feito em *Ruby on Rails* e *frontend* desenvolvido em *Vue.js.*

Palavras-chave: Sistemas de gestão de estúdios de musculação. Métodos ágeis. Ruby on Rails. Vue.js.

ABSTRACT

In a context in which physical education professionals noticed the demand for providing a personalized service, there was an increase in the number of new bodybuilding studios since their business model works with a small number of students, but with training routines and monitoring student closer, seeking to always improve training routines so that the student's objective is achieved. In contrast to large gyms where their business model seeks to have as many students as possible, which in some cases can reduce the time dedicated by professionals to students. With the reduced number of professionals in bodybuilding studios, which in many cases may be just one professional to manage students, plan and apply training as well as other commitments related to the financial part such as monthly payment control, this can overwhelm the professional while managing bodybuilding studios. Thus, the existence of a platform that simplifies the execution of these activities ca be of great help in the routine of these professionals. In this work, a software was developed seeking to simplify the routines involved in bodybuilding studios management, in which agile development methods were applied throughout the project. A web architecture was used, with a Ruby on Rails backend app and a Vue.js frontend app.

Keywords: Bodybuilding studio management systems. Agile methods. Ruby on Rails. Vue.js.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

© - copyright
@ - arroba
® - marca registrada
Σ - somatório de números
Π - produtório de números

SUMÁRIO

1 INTRODUÇÃO

Com a busca por uma melhor qualidade de vida e melhora do bem-estar físico, social e emocional, aumentou a procura pela prática de exercícios físicos, uma vez que existe uma correlação entre qualidade de vida e prática de exercícios físicos como observado por Macedo *et. al.* (2012, p. 19).

A prática de musculação aumentou por ser uma das modalidades de exercício físico mais acessíveis, e com isso surgiram demandas na preparação de rotinas de treino personalizadas, visando atender objetivos e disponibilidades individuais, buscando prestar o serviço adequado e que proporcione melhora do bem-estar físico, social e emocional (PANTA *et. al.*, 2017).

Em contraponto ao serviço prestado por grandes academias que têm sua estratégia de ganho baseada no número de alunos, resultando em uma abordagem generalista na preparação de rotinas de treino, os estúdios de musculação têm a proposta de prestar um serviço personalizado de acordo com o objetivo e disponibilidade do aluno (PARENTE, 2022).

O profissional consegue prestar um serviço de melhor qualidade com o menor número de alunos sendo atendidos simultaneamente se atentando a execução dos exercícios. Assim ele pode observar o progresso ou estagnação do aluno e com isso preparar um plano de ação ou nova rotina de treino com base nas observações feitas durante os treinos (BUSTAMANTE, 2018).

Isso posto, o objetivo deste trabalho é desenvolver um sistema web que facilite o dia a dia de profissionais simplificando o gerenciamento dos dados relacionados ao estúdio, como alunos, planos, mensalidades e rotinas de treino, permitindo, com isso, permitindo que o profissional consiga dedicar mais tempo aos alunos e no crescimento de seu estúdio.

1.1 PROBLEMA

Devido aos objetivos distintos de cada aluno, surge a necessidade de que sejam preparadas rotinas de treino personalizadas, impactando diretamente nos ganhos do aluno em comparação com a execução de uma rotina de treino genérica.

Com a diversidade de objetivos e rotinas de treino, surge a necessidade do acompanhamento e ajuste das rotinas de acordo com observações na execução dos

exercícios e variação em métricas, que variam de acordo com o objetivo do aluno (EVANGELISTA *et. al.*, 2021).

Como observado por Liz *et. al.* (2015), a demora na percepção de resultado pelo aluno é um fator considerável na desistência da prática de musculação e por ser subjetiva pode estar em desacordo com a realidade. Com o auxílio de algumas métricas colhidas durante os treinos, como quantidade de peso utilizada, a diminuição na necessidade de descanso, aumento no número de repetições e diminuição no percentual de gordura podem servir como um pilar motivacional para que esse aluno continue a treinar uma vez que fica perceptível a evolução atingida.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver uma aplicação web que auxilie no gerenciamento de um estúdio de musculação, simplificando o armazenamento e visualização de informações relacionadas a alunos, rotinas de treino, planos e mensalidades.

1.2.2 Objetivos específicos

Os objetivos específicos do trabalho são:

a) Permitir que o personal trainer gerencie os alunos;
b) Possibilitar criação e aplicação de rotinas de treino;
c) Gerenciar horários e rotinas de treino atribuídas a alunos;
d) Permitir que os alunos acompanhem os treinos agendados;
e) Centralizar o gerenciamento de mensalidades e planos por aluno.

1.3 JUSTIFICATIVA

Em razão das características do modelo de negócio de estúdios de musculação serem autogerenciáveis, em que normalmente o proprietário do estúdio também faz o controle de matrículas, planos de mensalidade e preparação de rotinas de treino, o profissional pode se ver sobrecarregado o que impacta na qualidade do seu trabalho uma vez que diminui o tempo dedicado a preparação e acompanhamento de alunos.

Com isso, o desenvolvimento de um software que auxilie o profissional no cumprimento de suas rotinas diárias, como gerenciamento de alunos e planos de mensalidades, pode proporcionar ao profissional um tempo maior de dedicação na preparação e aprimoramento de rotinas de treino impactando diretamente no resultado dos treinos e melhorando a retenção de alunos.

Com os dados centralizados e normalizados surge a oportunidade de prover visibilidade dos dados relacionados ao estúdio de forma visual, na forma de gráficos, *dashboards* e linhas temporais auxiliando o profissional a entender o comportamento dos alunos. Com isso ele toma melhores decisões para seu negócio, sabendo qual plano de mensalidade tem maior ou menor adesão, analisa características de seus alunos e defini qual é seu público-alvo direcionando, assim, ações de marketing melhorando a captação de novos alunos.

1.4 ESTRUTURA DO DOCUMENTO

O Capítulo 2 trata da fundamentação teórica, descrevendo conceitos relacionados a criação de rotinas de treino personalizadas e apresentando aplicativos semelhantes.

Já no Capítulo 3 são descritos os materiais e métodos utilizados para o desenvolvimento da aplicação, apresentando a metodologia de engenharia de software, ferramentas e metodologias de gerenciamento aplicadas no desenvolvimento do software.

No Capítulo 4 são apresentadas as funcionalidades do sistema, e por fim, o Capítulo 5 apresenta as considerações finais, resultados e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a revisão da literatura utilizada como base teórica no desenvolvimento do software proposto. São abordados os benefícios de rotinas de treino personalizadas à necessidade do aluno, desistências na prática da musculação e comparação com softwares semelhantes.

2.1 BENEFÍCIOS DO USO DE ROTINAS DE TREINO PERSONALIZADA

Surgiram diversos perfis de alunos buscando objetivos distintos com a popularização da prática da musculação, em que alunos podem ingressar na prática da musculação buscando melhorar sua qualidade de vida, emagrecimento, ganho de massa muscular ou melhoria de performance em algum esporte específico. Com isso ficou clara a diferença dos ganhos proporcionados por rotinas de treino personalizadas em comparação a rotinas de treino genéricas (FILARDO, 2001).

O acompanhamento e refino de uma rotina de treino personalizada tem impacto direto na maximização e consistência na conquista do objetivo. Para efetuar tal refinamento o profissional leva em consideração a qualidade e a intensidade com que o aluno consegue executar os exercícios planejados. Para ajudar o profissional a decidir o que deve ser ajustado na rotina de treino, o profissional leva em consideração métricas que estão alinhadas com o objetivo do aluno, como por exemplo, a diminuição do percentual de gordura em alunos que buscam emagrecimento (GENTIL, 2010).

Como observado por Judge, 2010, a periodicidade com que as rotinas de treino são alternadas geram um impacto grande, uma vez que o corpo tende a se acostumar com os estímulos recebidos durante o treino e com isso o nível de esforço empregado na execução de uma mesma rotina de treino tende a diminuir ao longo das semanas. Logo, o profissional deve definir qual é o momento ideal para que o aluno dê início a uma nova rotina de treinos, alternando grupos musculares ou mudando a ênfase da execução focando em um número maior de repetições ou buscando a maior carga suportada com número baixo de repetições.

2.2 FERRAMENTAS E TECNOLOGIAS

Nesta seção são apresentadas as tecnologias e ferramentas adotadas para a análise e desenvolvimento do software.

2.2.1 Ruby

A linguagem de programação Ruby foi proposta em 1995 por Yukihiro “Matz” Matsumoto com objetivo de ser uma linguagem simples e produtiva, sendo totalmente orientada a objetos, e suportando programação funcional e meta-programação (RUBY, 2022).

Devido a sua simplicidade, o Ruby é utilizado em diversos cenários, de aplicações web a scripts utilizados na automatização de tarefas repetitivas, porém o uso mais comum da linguagem é no desenvolvimento de aplicações web. Apesar de não estar mais entre as linguagens mais utilizadas pelo mercado o Ruby é uma das mais valorizadas resultando em uma maior média salarial (REVELO, 2022).

2.2.2 Ruby on Rails

O Ruby on Rails (ROR) é um *framework* gratuito e de código aberto escrito em Ruby, que busca tornar o desenvolvedor mais produtivo no desenvolvimento de aplicações web. Tendo sido extraído a partir do código dos produtos comercializados pelo Basecamp por David Heinemeier Hansson (DHH) e atualmente mantido pela comunidade que se formou ao seu redor (FUENTES, 2010).

O ROR trouxe uma nova abordagem de implementação de conceitos clássicos como a arquitetura *Model View Controller* (MVC), e novos conceitos que se tornariam padrão na construção de novos frameworks. Seguindo filosofias de *Clean Code* (código limpo), *Don't Repeat Yourself* (DRY), testes e *Convension Over Configuration* (convenção sobre configuração) onde ao invés de arquivos XML contendo as configurações do projeto, o ROR define uma série de convenções para nome e localização de arquivos onde, ao segui-las, o desenvolvedor ganha uma série de funcionalidades economizando tempo de codificação (RUBY ON RAILS, 2022).

2.2.3 Vue.JS

O Vue.js é um *framework* escrito em JavaScript utilizado no desenvolvimento de Single Page Applications (SPA), sendo considerado pela comunidade como um meio termo entre os frameworks Angular5 e React6. Trazendo o poder do *Angular* com a simplicidade do *React* resultando em um framework produtivo e de rápida curva de aprendizagem (VUE, 2022).

2.2.4 Git

É um sistema de controle de versões distribuído gratuito e de código aberto utilizado no gerenciamento de repositórios de código. Atualmente é o padrão de mercado e serve como base para principais plataformas de Cloud utilizadas no compartilhamento de projetos de software como por exemplo *Github* e *GitLab* (GIT, 2022).

2.2.5 Github

O GitHub é uma plataforma cloud que iniciou com a proposta de compartilhar código através de repositórios Git, lançado em 2008. Ele foi adotado por parte da comunidade Open Source hospedando o código de frameworks como Ruby on Rails, e Spring Boot, sistemas operacionais como o Linux e linguagens de programação como Ruby e Elixir por exemplo.

Apesar de sua proposta inicial ser voltada ao compartilhamento de código, a plataforma evoluiu permitindo que sejam executadas rotinas de testes unitários, verificações na qualidade do código e até Build e Deploy de novas versões de um projeto centralizando boa parte da interação com o código em apenas um local (GITHUB. 2022).

2.2.6 Docker

É um conjunto de ferramentas utilizadas na criação e gerenciamento de containers que contém a virtualização de processos, sendo executados em um sistema operacional Linux, tornando possível abstrair por meio de código um ambiente próximo ao ambiente de produção reduzindo possíveis incompatibilidades e facilitando a configuração de ambientes de desenvolvimento (DOCKER, 2022).

2.2.7 PostgreSQL

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) gratuito e de código livre lançado em 1997. É largamente utilizado em projetos que contém bancos de dados relacionais por dispor de uma série de recursos avançados como Materialized Views, Triggers, e definição de tipos customizados (POSTGRESQL, 2022).

2.2.8 Visual Studio Code

O Visual Studio Code é um editor de texto puro com suporte para diversas linguagens de programação e com possibilidade de criação e instalação de plugins de terceiros, adicionando suporte para mais linguagens de programação, ferramentas utilizadas durante o desenvolvimento e até automatização de tarefas como compilação ou execução de testes automatizados (CODE, 2022).

2.2.9 Scrum

Scrum é um *framework* utilizado por equipes enxutas no desenvolvimento e evolução de produtos complexos, sendo utilizado em produtos relacionados à tecnologia uma vez que suas premissas se encaixam com a dinamicidade de produtos que podem mudar seu foco de forma ágil (SCHWABER, 2001).

O *Scrum* disponibiliza uma série de práticas como *sprints*, o hábito de estimar tarefas e algumas cerimônias que ajudam o time a compreender o que está indo bem, o que deve mudar e atualizações diárias sobre o progresso das tarefas. O que resulta em uma melhor visibilidade e clareza aos membros do time sobre o que deve ser feito e qual a prioridade de cada tarefa.

Com o final de cada *sprint* é feita uma reunião em que os membros do time falam o que foi bom, o que foi ruim, e juntos definem ações que serão feitas a partir da próxima *sprint* com objetivo de seguir melhorando o fluxo de trabalho do time (RADIGAN, 2022).

2.2.10 UML

A *UML* (*Unified Modeling Language*) é um padrão utilizado na elaboração de diagramas com objetivo de documentar projetos de *software,* definindo padrões para diagramas estruturais e comportamentais. Os diagramas estruturais apresentam a base do sistema, enquanto os diagramas comportamentais descrevem as ações disponibilizadas pelo sistema (BOOCH, 2006).

A vantagem de se utilizar a UML em projetos de software é o fato de estabelecer um padrão comum de representação de artefatos presentes em um software, e com isso diminuir o tempo para que novos membros possam se iterar em projetos já em andamento, uma vez que todo projeto de software que tenha sua documentação seguindo a UML terá o mesmo padrão.

2.3 SOFTWARES SEMELHANTES

Para embasar o desenvolvimento do sistema proposto buscou-se por sistemas com objetivos semelhantes, dessa forma conseguindo verificar os pontos positivos, negativos e possíveis diferenciais.

Com isso foram analisados os sistemas comerciais Datafitness[1], Technofit[2] e iFitness[3], todos sendo sistemas comerciais voltados no gerenciamento de academias nos moldes de um *Customer Relationship Management* (CRM) podendo ser comparados com o sistema proposto.

O primeiro software analisado foi o Datafitness, que se apresenta como o software para gestão de academias. Ele tem quatro planos disponíveis no momento da avaliação, variando entre planos básicos com limitação de alunos ativos e prescrição de exercícios, por exemplo, a um plano completo sem limitação de alunos e disponibilizando ferramentas de marketing, notificação de alunos via SMS e planos de metas.

Já o iFitness dispõe um aplicativo disponível para as plataformas Android e IOS. Ele apresenta funcionalidades como pagamento recorrente, um módulo de *follow*

---

[1] https://logonsistemas.com.br/
[2] https://www.tecnofit.com.br/
[3] http://ifitness.com.br/

up que auxilia os funcionários a lembrar dos afazeres e integração simples com *hardwares* como catracas e câmeras para reconhecimento facial, esses fornecidos pela mesma empresa que disponibiliza o software.

Por fim, o Tecnofit disponibiliza uma série de módulos que agregam funcionalidades relacionadas ao gerenciamento de alunos, financeiro e marketing. Este está disponível para os sistemas Android e IOS para uso dos alunos e uma aplicação web para o gerenciamento da academia. O Tecnofit tem sua cobrança calculada com base no número de alunos ativos, sendo R$2,18 reais por aluno, o que, segundo o fornecedor, permite um crescimento gradual.

Com isso, pode-se verificar que os principais sistemas de gerenciamento de academias têm como público-alvo academias com grande volume de alunos, demandando de muitos serviços além do gerenciamento de alunos e rotinas de treino. Logo este trabalho propõe o desenvolvimento de um sistema que seja focado em estúdios de musculação com uma proposta mais simples buscando apenas o gerenciamento de alunos, rotinas de treino e organização financeira.

QUADRO 1 - COMPARATIVO ENTRE SOFTWARES SEMELHANTES

	Datafitness	**iFitness**	**Tecnofit**
Possui prescrição de exercícios	X	X	X
Aplicativo para smartphone		X	X
Integração com catracas e câmeras		X	
Ferramenta de marketing	SMS e e-mail	e-mail	e-mail
Tipo de plano	Plano com valor fechado de acordo com o número de alunos ativos.	Plano com valor variável, sendo cobrado de acordo com o número de alunos ativos.	Plano com valor variável, sendo cobrado de acordo com o número de alunos ativos.

FONTE: O autor (2022).

3 MATERIAIS E MÉTODOS

O primeiro passo na elaboração de projetos de software é a escolha da metodologia que será utilizada como base para definir a forma como a equipe irá trabalhar. A escolha da metodologia é feita avaliando o prazo e a forma de entrega do projeto, avaliando se deve ser uma única entrega ou pode ser dividido em entregas menores, bem como os recursos disponíveis e a forma com que o time está acostumado a trabalhar.

As metodologias de desenvolvimento de software podem ser divididas entre métodos ágeis e métodos tradicionais. Os métodos tradicionais buscam fazer um planejamento detalhado no início do projeto, gerando documentação em todos os passos da modelagem e detalhando todas as funcionalidades do sistema como detalhado por Larman (2007).

Em contraponto, os métodos ágeis prezam pela entrega de valor de forma constante, diminuindo drasticamente o tempo dedicado a documentação e planejamento de longo prazo (SCHWABER, 2001). Com isso, é mais simples fazer adaptações mediante alguma mudança no objetivo do produto, e se encaixa muito bem com projetos voltados para aplicações web, uma vez que normalmente este tipo de produto pode ser entregue e melhorado de forma incremental.

3.1 PROCESSO DE ENGENHARIA DE SOFTWARE

O processo de engenharia de software se dá a partir da união de conceitos, metodologias e boas práticas que, juntos, servem como base teórica no desenvolvimento, crescimento e evolução de um projeto de software.

No projeto proposto foram aplicados métodos ágeis de desenvolvimento de software seguindo as práticas propostas no manifesto ágil (BECK, 2009), entregando valor de forma constante em pequenas entregas. O valor citado se dá por meio de entregas de funcionalidades seguido da coleta de feedback. Esse servirá como aprendizado e base para próximas tomadas de decisões em relação ao rumo do projeto e assim, repetir os passos anteriores em iterações pequenas com o cliente.

Em conjunto com o *Scrum,* as práticas de *eXtreme Programing* (XP) auxiliam equipes de pequeno e médio porte a desenvolver softwares de qualidade de forma ágil. Com isso, o time é empoderado para tomar decisões que melhorem o processo

de entrega constante de valor através de utilização de valores como comunicação, feedback, simplicidade, coragem e respeito (BECK, 2004).

É necessário que o código do projeto esteja organizado, seja de simples entendimento e principalmente bem testado para manter o ritmo de codificação consistente. Dessa forma é possível continuar a adição de novas funcionalidades sem que exista um aumento considerável da complexidade do código, o que por consequência, demanda menos tempo e reduz o risco da introdução de novos bugs (MARTIN, 2009).

A organização e planejamento do projeto foi feita seguindo o *framework* ágil *Scrum*, buscando melhor organização e visibilidade referentes ao andamento do projeto. Inicialmente, as funcionalidades planejadas para o sistema foram priorizadas de forma que o sistema pudesse ter sua versão mínima disponível para uso o mais rápido possível.

Uma vez definida a priorização das funcionalidades, estas foram divididas em tarefas menores visando pequenas entregas de valor que juntas compõe uma funcionalidade completa. Para cada uma das tarefas foram escritas Histórias de Usuário (HU), em que foram descritos os requisitos mínimos para a entrega da tarefa permitindo com que fossem atribuídas de forma mais assertiva pontuações referentes ao esforço.

Após criação das tarefas foi feita a estimativa da velocidade da equipe com base na quantidade de horas de dedicação semanais em relação à quantidade de entregas. Uma vez compreendida a velocidade foi possível definir a duração das *sprints* e quantos pontos de função podem ser alocados por *sprint*, e com isso foi feita a distribuição das tarefas entre as *sprints*.

3.2 PLANO DE SPRINTS

Após avaliação dos recursos disponíveis e das características da equipe para a execução do projeto optou-se por utilizar o *framework Scrum* para auxiliar no gerenciamento do projeto. Com isso, definiu-se que o tempo de duração de cada *sprint* seria de duas semanas, em que uma *sprint* comporta oito pontos de função e cada ponto de função equivale a aproximadamente oito horas de dedicação.

Ao concluir a modelagem das funcionalidades foi feita a priorização de acordo com o que poderia ser entregue e seu valor agregado ao sistema. Uma vez priorizadas

as tarefas foram pontuadas de acordo com uma previsão de sua complexidade e tempo de execução.

Após priorizadas cada uma das funcionalidades foi dividida em tarefas e pontuadas de acordo com o esforço estimado. Em seguida foi feito o planejamento das sprints atribuindo um número de tarefas que totalizassem ou chegassem o mais próximo possível do total de pontos de função comportados por *sprint*, sendo um total de 4 pontos por *sprint* em que 1 ponto de função equivale à 4 horas de dedicação.

No QUADRO 2 estão detalhadas as tarefas atribuídas a cada uma das sprints.

QUADRO 2 - PLANEJAMENTO DE SPRINTS

Sprint	Início	Fim	Tarefas
1	20/02/2022	14/03/2022	Criação e configuração dos ambientes de desenvolvimento referentes ao cliente web e API Início do desenvolvimento das histórias de usuário Autenticação de usuário Barra de navegação para usuário logado
2	14/03/2022	28/03/2022	Cadastrar exercício Listagem de exercícios Cadastrar aluno Logout da plataforma Desenvolvimento do diagrama de casos de uso
3	28/03/2022	11/04/2022	Cadastrar rotina de treino Vincular exercícios a uma rotina de treino Revisar histórias de usuário
4	11/04/2022	25/04/2022	Agendar treino Aplicar treino Finalizar treino em andamento Iniciar desenvolvimento dos diagramas de sequência
5	25/04/2022	09/05/2022	Escrita do primeiro capítulo da monografia Editar rotina de treino Editar exercício Exibir exercício Listar alunos Visualizar agenda de treinos cadastrados Concluir desenvolvimento dos diagramas de sequência
6	09/05/2022	23/05/2022	Cancelar horário de treino Escrita do segundo capítulo da monografia Editar horário de treino Adicionar diagrama de classes

7	23/05/2022	06/06/2022	Editar aluno Escrita do terceiro capítulo da monografia Adicionar atalho na página inicial para treinos em andamento Adicionar modelo do banco de dados
8	06/06/2022	20/06/2022	Listar rotinas de treino Editar série de exercícios relacionada a rotina de treino
9	20/06/2022	04/07/2022	Cadastrar plano Listar planos Editar plano
10	04/07/2022	18/07/2022	Exibição de plano Vincular plano à aluno Visualizar histórico de planos de um aluno
11	05/09/2022	19/09/2022	Visualizar histórico de treinos de aluno Cancelar plano atual de aluno Cadastro de pagamento Exibição de histórico de pagamentos no perfil de aluno
12	19/09/2022	03/10/2022	Escrita do quarto capítulo da monografia Adicionar relatórios financeiros
13	03/10/2022	17/10/2022	Adicionar relatórios com dados de alunos Escrita do quinto capítulo da monografia Revisão dos apêndices

FONTE: O autor (2022)

3.3 HARDWARE UTILIZADO

No QUADRO 3 são apresentados os hardwares utilizados no desenvolvimento do projeto.

QUADRO 3 - HARDWARE UTILIZADO

	Sistema Operacional	Processador	Memória RAM	HD
Desktop Linux	Ubuntu 20.04	I9 9900k 8 núcleos e 16 threads	32GB	2TB SSD
MacBook Pro 2017	MacOS Monterey	Intel Core i5 Dual-Core	8GB	256GB SSD
Raspberry Pi 3 Model B+	Raspbian	Broadcom Quad-Core	4GB	32GB

FONTE: O autor (2022)

3.4 MATERIAIS

No QUADRO 4 são apresentadas as tecnologias e suas respectivas versões aplicadas durante o desenvolvimento do projeto.

QUADRO 4 – MATERIAIS UTILIZADOS

Nome	Versão	Objetivo
Ruby	3.1.2	Linguagem utilizada no desenvolvimento da API.
Ruby on Rails	7.0.3	*Framework* utilizado como base para o desenvolvimento da API
Node	14	Ambiente de execução da aplicação cliente web
Vue.js	2.6.11	*Framework* utilizado como base para o desenvolvimento do cliente web
Docker	20.10.17	Criação de um ambiente de desenvolvimento isolado e independente de sistema operacional
Docker-compose	3	Ferramenta utilizada para orquestração dos containers utilizados pelas aplicações
PostgreSQL	12	Banco de dados relacional

FONTE: O autor (2022)

3.5 DESENVOLVIMENTO DO PROJETO

Durante o desenvolvimento do projeto buscou-se executar o que foi planejado nas *Sprints* (QUADRO 2), desenvolvendo as funcionalidades planejadas e escrevendo a monografia ao longo de treze *Sprints* descritas nas seções a seguir.

3.5.1 Sprint 1

Durante a *Sprint* 1 foi dado início a elaboração das tarefas que compõe as funcionalidades na forma de histórias de usuário (APÊNDICE B), bem como a criação dos projetos e configuração dos ambientes de desenvolvimento referentes a API e cliente web. Por fim, foram implementados a autenticação de usuário e menu lateral de navegação.

3.5.2 Sprint 2

Já na *Sprint* 2 foram implementados o cadastro e listagem de alunos, bem como o *logout* de usuário e a criação do diagrama de casos de uso (APÊNDICE A).

3.5.3 Sprint 3

A *Sprint* 3 teve como objetivo implementar o cadastro de rotina de treino em conjunto com a vinculação de séries de exercício, também foi concluída a divisão das funcionalidades planejadas em tarefas descritas na forma de histórias de usuário (APÊNDICE B).

3.5.4 Sprint 4

Durante a *Sprint* 4 foi implementado o agendamento de treino, bem como as ações de dar início e concluir um treino concluindo as tarefas que compõe o fluxo principal da aplicação. Por fim, foi feita a criação dos diagramas de sequência (APÊNDICE D).

3.5.5 Sprint 5

Já na *Sprint* 5 teve como objetivo concluir as funcionalidades relacionadas com o gerenciamento de exercícios, em que foram implementados a edição e exibição detalhada de exercícios. Também foram implementados a edição de rotina de treino, e a listagem de alunos. Por fim, foi feita a escrita do primeiro capítulo da monografia.

3.5.6 Sprint 6

Durante a *Sprint* 6 foi escrito o segundo capítulo da monografia, em conjunto com a implementação do cancelamento e edição de horário de treino concluindo as funcionalidades básicas relacionadas ao gerenciamento de horários de treino. Por fim, foi feita a criação do diagrama de classes (APÊNDICE C).

3.5.7 Sprint 7

A *Sprint* 7 teve como objetivo concluir as funcionalidades relacionadas ao gerenciamento de alunos, por meio da adição da edição de aluno, bem como a simplificação da aplicação de treinos adicionando um atalho na página inicial que exibe os treinos em andamento. Por fim, foi escrito o terceiro capítulo da monografia bem como a criação do modelo físico do banco de dados (APÊNDICE E).

3.5.8 Sprint 8

Durante a *Sprint* 8 foi implementada listagem de rotina de treino, bem como a edição de séries de exercícios concluindo as funcionalidades relacionadas ao gerenciamento de rotinas de treino.

3.5.9 Sprint 9

A *Sprint* 9 teve como objetivo incluir as primeiras funcionalidades relacionadas ao gerenciamento de planos de mensalidade, permitindo o cadastro, listagem e edição de planos de mensalidade.

3.5.10 Sprint 10

Nesta *Sprint* foram concluídas as funcionalidades relacionadas ao gerenciamento de planos de mensalidade, tendo sido incluídos a exibição detalhada de um plano de mensalidade bem como a vinculação entre um aluno e um plano.

3.5.11 Sprint 11

Durante a *Sprint* 11 foi implementado o cancelamento de um plano de mensalidade vinculado a um aluno, bem como a exibição do histórico de treinos realizados por aluno. Por fim, foram adicionadas funcionalidades que permitem o gerenciamento de pagamentos, incluindo o cadastro e exibição de histórico de pagamentos.

3.5.12 Sprint 12

Nesta *Sprint* foi implementado o *dashboard* com os dados financeiros e deu-se início a escrita do quarto capítulo da monografia.

3.5.13 Sprint 13

Na *Sprint* 13 foi implementado o relatório com dados de alunos concluindo a implementação das funcionalidades planejadas para a aplicação. Por fim, foi concluída a escrita da monografia, encerrando o quarto capítulo, escrevendo o quinto capítulo e revisando os apêndices.

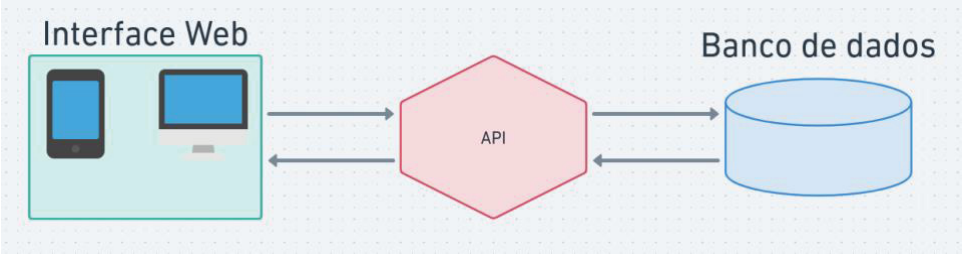
4 APRESENTAÇÃO DOS RESULTADOS

Neste capítulo é apresentada a arquitetura e funcionalidades do software apresentado neste trabalho. O sistema é composto de uma interface web e uma API (*Application Programing Interface*) responsável pelas iterações com o banco de dados.

4.1 ARQUITETURA DO SISTEMA

O sistema segue a arquitetura cliente-servidor, sendo constituído por duas camadas (FIGURA 1), em que uma aplicação do tipo SPA (*Single Page Application*) se comunica com uma API através de chamadas HTTP.

FIGURA 1 – ARQUITETURA DO SISTEMA



FONTE: O autor (2022)

Já o SGBD (Sistema Gerenciador de Banco de Dados) escolhido foi o PostgreSQL, por ser uma implementação gratuita e com amplo suporte da comunidade. Porém não foi utilizada nenhuma funcionalidade específica do PostgreSQL, e com isso é possível mudar o SGBD de forma simples, uma vez que o ORM (*Object-Relational Mapping*) utilizado em aplicações ROR gera código SQL análogo a especificidade de SGBDs.

4.1.1 API

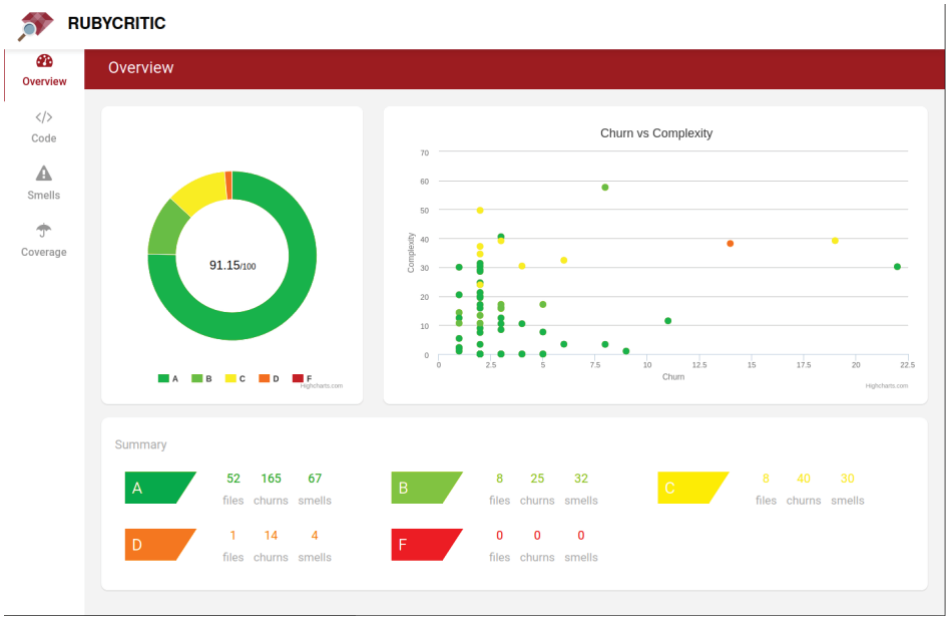
A API responsável por fazer a interface com o banco de dados foi desenvolvida na linguagem Ruby e utilizando o *framework* Ruby on Rails (ROR), uma vez que a arquitetura MVC (*Model View Controller*) e o suporte nativo a requisições

HTTP contendo dados no formato JSON (*JavaScript Object Notation*) simplifica o desenvolvimento deste tipo de aplicação.

Durante o desenvolvimento foram aplicadas técnicas de desenvolvimento de código limpo, buscando reduzir o crescimento de complexidade em manter e adicionar código a aplicação durante o andamento projeto.

Para tal, foram utilizadas algumas ferramentas que auxiliam na visualização da complexidade de código como o RUBYCRITIC (2013), que gera relatórios mostrando a complexidade, código repetido e alguns *smells* conhecidos pela comunidade ROR (FIGURA 2).

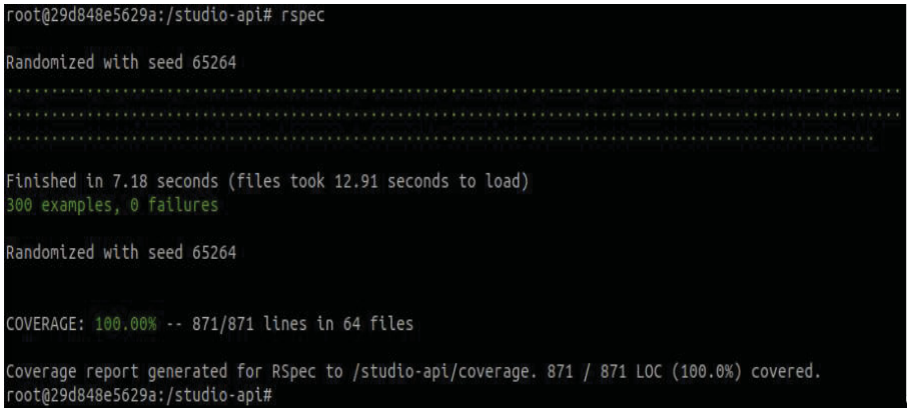
FIGURA 2 - RELATÓRIO DE QUALIDADE DE CÓDIGO



FONTE: O autor (2022)

Foram adicionados testes unitários em todas as classes da aplicação cobrindo um total de 100% das linhas de código para tornar viável a refatoração constante do código do projeto buscando a evolução por meio novas abstrações ou melhorias de legibilidade (FIGURA 3).

FIGURA 3 - RELATÓRIO DE COBERTURA DE TESTES API



FONTE: O autor (2022)

Em conjunto com testes unitários foram feitos testes de integração com objetivo de garantir que a execução dos fluxos completos esteja acontecendo da forma esperada. Os testes de integração foram adicionados na camada das *controllers*, uma vez que esta é a camada que recebe chamadas exteriores e retorna os dados resultantes do processamento.

Por ser desenvolvida visando o *deploy* em um ambiente de *cloud*, a aplicação foi desenvolvida buscando seguir os 12 fatores propostos por Wiggins (2013), permitindo que a aplicação possa ser escalada sem grandes mudanças em sua arquitetura.

Por fim, para garantir que o código escrito esteja padronizado e não contenha más práticas catalogadas pela comunidade foi utilizada a ferramenta de análise estática de código *Rubocop*, que navega pelo código indicando métodos com muitas linhas, classes contendo mais do que uma responsabilidade, nomenclatura de variáveis entre outras.

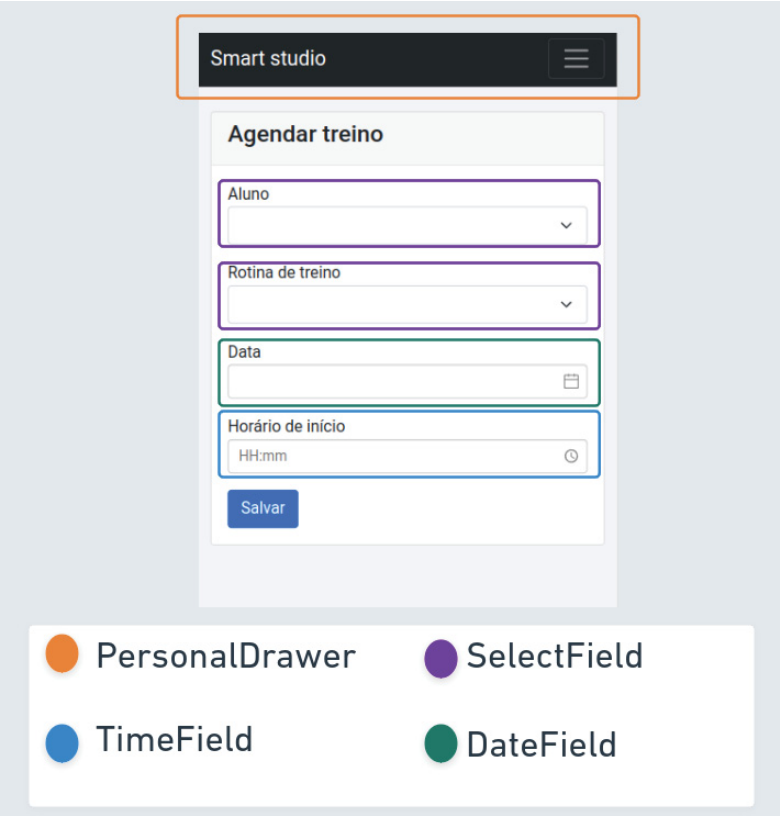
4.1.2 Cliente web

O cliente web foi desenvolvido utilizando o *framework* Vue.js, o que facilita na criação de telas dinâmicas e que não necessitam do recarregamento da tela a cada ação do usuário, tornando sua experiência mais agradável em comparação a aplicações que renderizam HTML no servidor.

O desenvolvimento da aplicação Vue.js seguiu a filosofia de “componentização” buscando dividir telas em componentes isolados, o que resulta em benefícios como a integridade conceitual e de design da aplicação. Todos os

componentes foram estilizados utilizando a mesma biblioteca *Cascading Style Sheets* (CSS), também permitindo que os componentes possam evoluir de forma independente. Logo, tais componentes se tornam concisos uma vez que são projetados para realizar uma única tarefa. Como ilustrado na FIGURA 4 as páginas são resultado da composição de componentes.

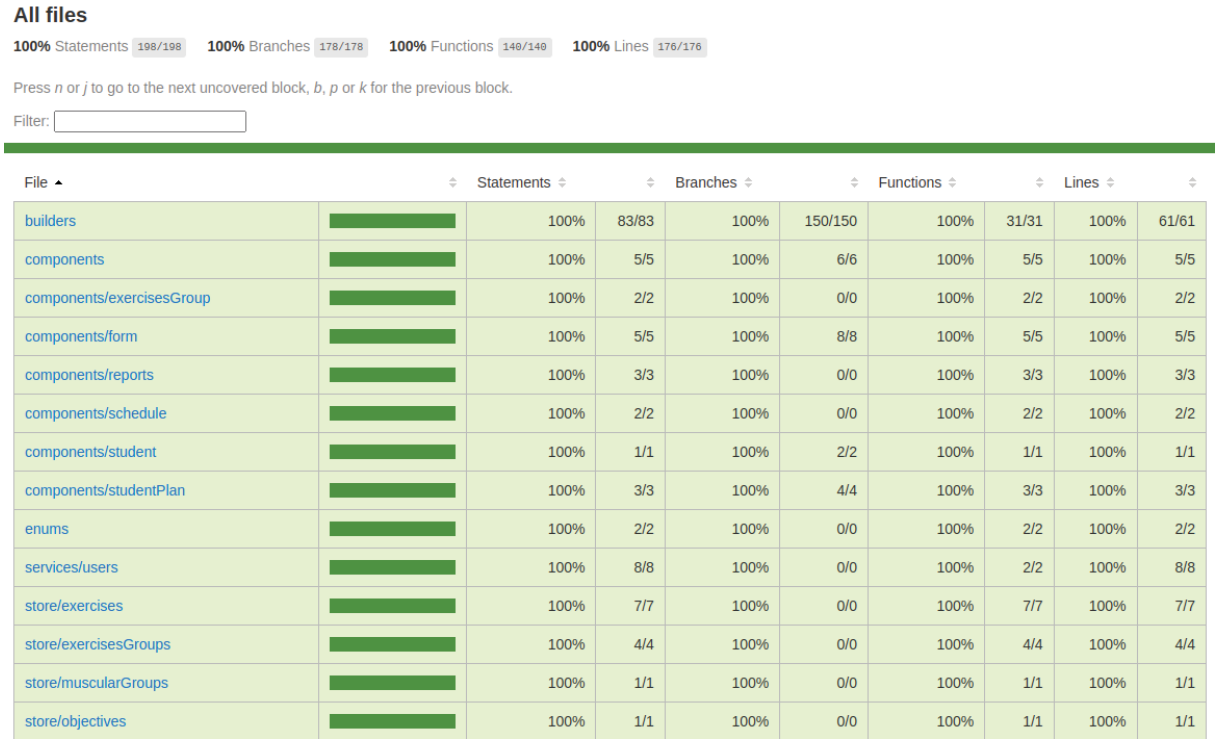
FIGURA 4 - TELA COMPOSTA POR COMPONENTES



FONTE: O autor (2022)

Com objetivo de manter a qualidade do código, a aplicação tem 100% de seu código coberta por testes unitários (FIGURA 5), validando o comportamento, chamadas HTTP realizadas e conteúdo renderizado. Os testes de integração foram desenvolvidos na camada de testes das *views*, em que uma *view* é um grupo de componentes que formam uma tela completa.

FIGURA 5 - RELATÓRIO DE COBERTURA DE TESTES CLIENTE WEB



All files

100% Statements `198/198` **100%** Branches `178/178` **100%** Functions `140/140` **100%** Lines `176/176`

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
builders	100%	83/83	100%	150/150	100%	31/31	100%	61/61
components	100%	5/5	100%	6/6	100%	5/5	100%	5/5
components/exercisesGroup	100%	2/2	100%	0/0	100%	2/2	100%	2/2
components/form	100%	5/5	100%	8/8	100%	5/5	100%	5/5
components/reports	100%	3/3	100%	0/0	100%	3/3	100%	3/3
components/schedule	100%	2/2	100%	0/0	100%	2/2	100%	2/2
components/student	100%	1/1	100%	2/2	100%	1/1	100%	1/1
components/studentPlan	100%	3/3	100%	4/4	100%	3/3	100%	3/3
enums	100%	2/2	100%	0/0	100%	2/2	100%	2/2
services/users	100%	8/8	100%	0/0	100%	2/2	100%	8/8
store/exercises	100%	7/7	100%	0/0	100%	7/7	100%	7/7
store/exercisesGroups	100%	4/4	100%	0/0	100%	4/4	100%	4/4
store/muscularGroups	100%	1/1	100%	0/0	100%	1/1	100%	1/1
store/objectives	100%	1/1	100%	0/0	100%	1/1	100%	1/1

FONTE: O autor (2022)

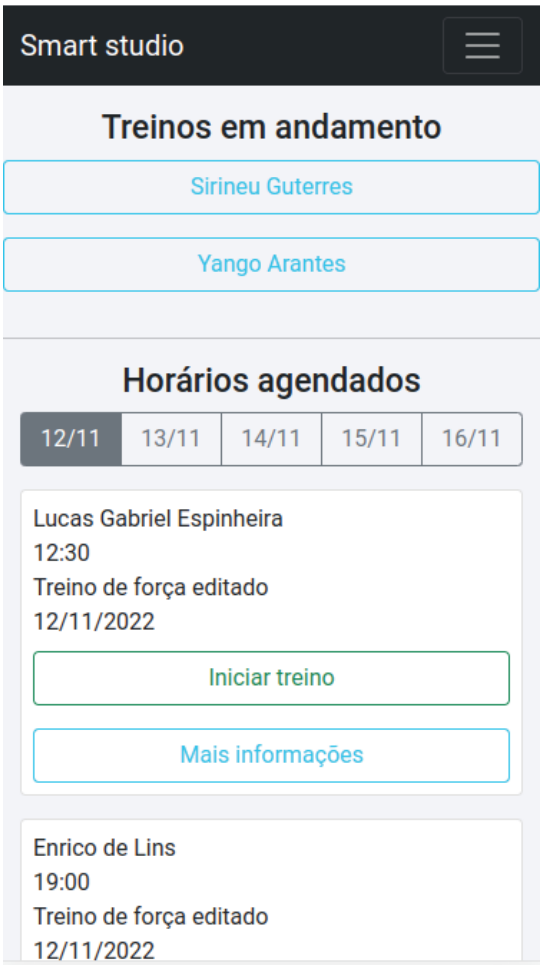
4.2 FUNCIONALIDADES

O Smart Studio é composto por sete módulos, possibilitando gerenciamento de exercícios, alunos, agenda, rotinas de treino, planos de mensalidade, pagamentos e relatórios.

4.2.1 Página inicial

Uma vez autenticado, o usuário é redirecionado para página inicial do sistema, em que é possível visualizar os horários agendados para um total de cinco dias e uma lista com os treinos que estão em andamento. Tais informações são as mais utilizadas durante o dia a dia de um *personal (*FIGURA 6*)*.

FIGURA 6 - TELA INICIAL DE USUÁRIO AUTENTICADO

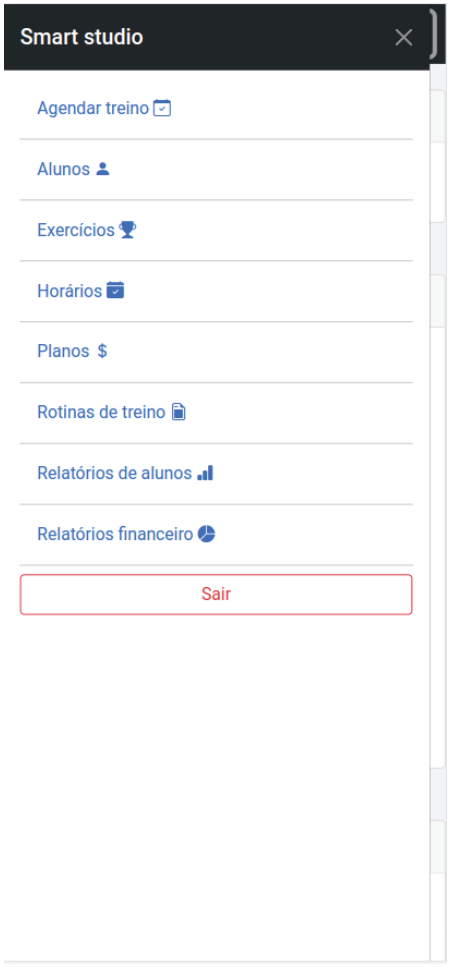


FONTE: O autor (2022)

4.2.2 Navegação

A aplicação disponibiliza um menu lateral com *links* para cada um dos módulos simplificando a navegação do usuário (FIGURA 7) para usuários autenticados.

FIGURA 7 - MENU DE NAVEGAÇÃO

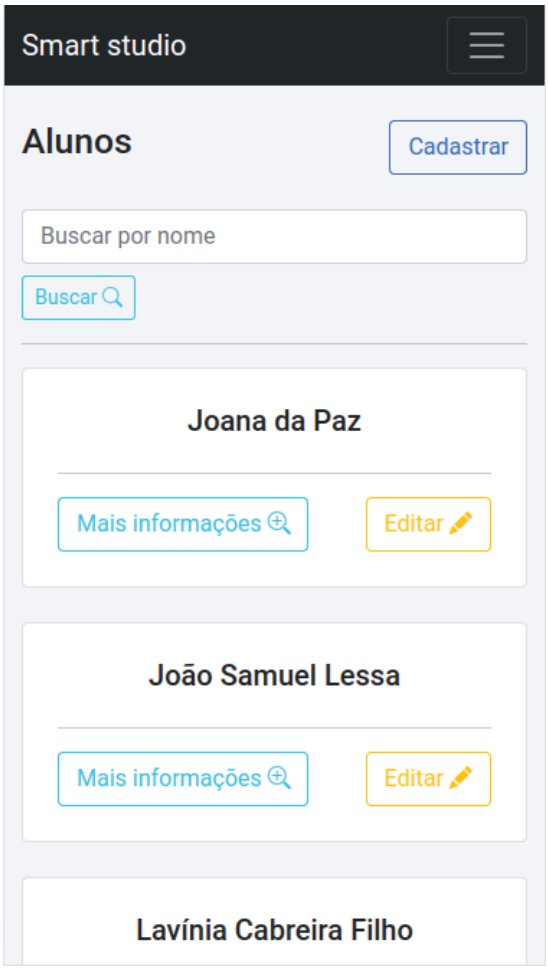


FONTE: O autor (2022)

4.2.3 Módulo alunos

O módulo responsável pelo gerenciamento de alunos permite a criação, edição, listagem permitindo buscas por parte do nome e exibição do perfil de aluno (FIGURA 8).

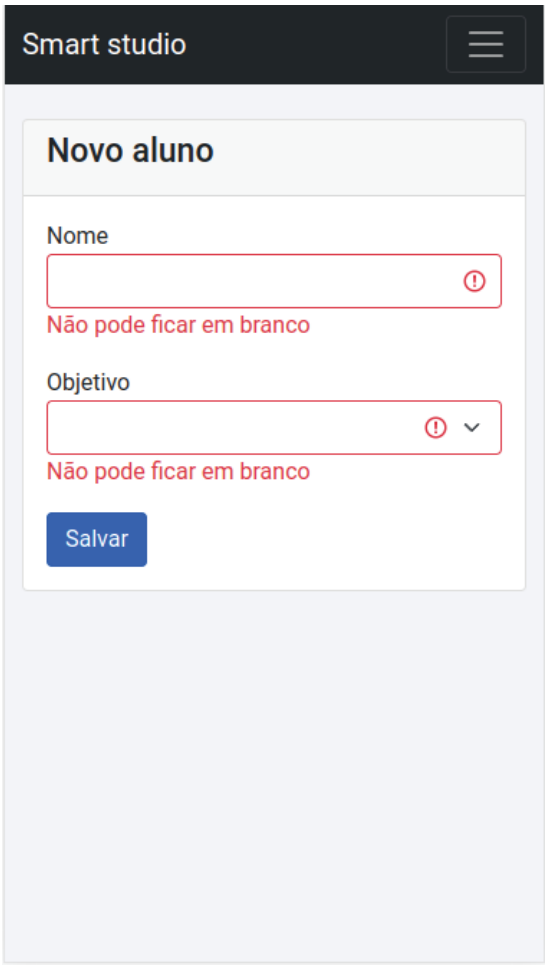
FIGURA 8 - LISTAGEM DE ALUNOS



FONTE: O autor (2022)

Na tela de cadastro de aluno é preciso informar o nome e o objetivo do aluno, ambos os campos sendo obrigatórios e validados no cliente web e na API. O campo nome tem sua unicidade validada apenas pela API (FIGURA 9).

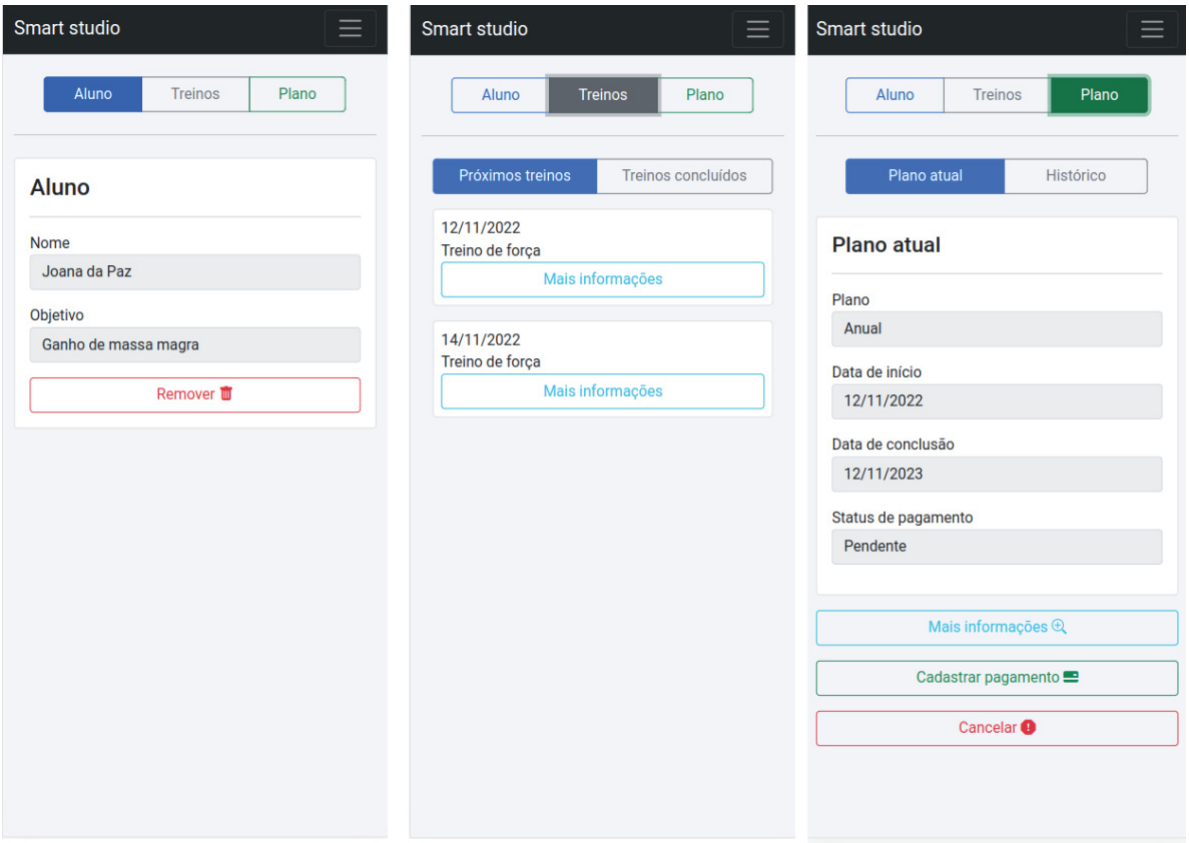
FIGURA 9 - CADASTRO DE ALUNOS



FONTE: O autor (2022)

A exibição do perfil de aluno compila todos os dados relacionados a um aluno em uma única tela, agregando as informações de rotinas de treino, planos e pagamentos realizados (FIGURA 10).

FIGURA 10 - PERFIL DE ALUNO



FONTE - O autor (2022)

4.2.4 Módulo rotina de treino

O módulo responsável por gerenciar as rotinas de treino permite que um usuário autenticado do tipo *personal trainer* possa criar, editar, listar, remover rotinas de treino e relacionar exercícios a uma rotina de treino.

Para acessar as funcionalidades referentes ao gerenciamento de rotinas de treino, o usuário deve clicar no *link* "Rotinas de treino" presente no menu lateral. Com isso, ele é redirecionado para uma tela contendo a listagem paginada das rotinas de treino já cadastradas, um *link* para cadastrar novas rotinas de treino e um campo que permite a busca de rotinas de treino por parte do nome (FIGURA 11).

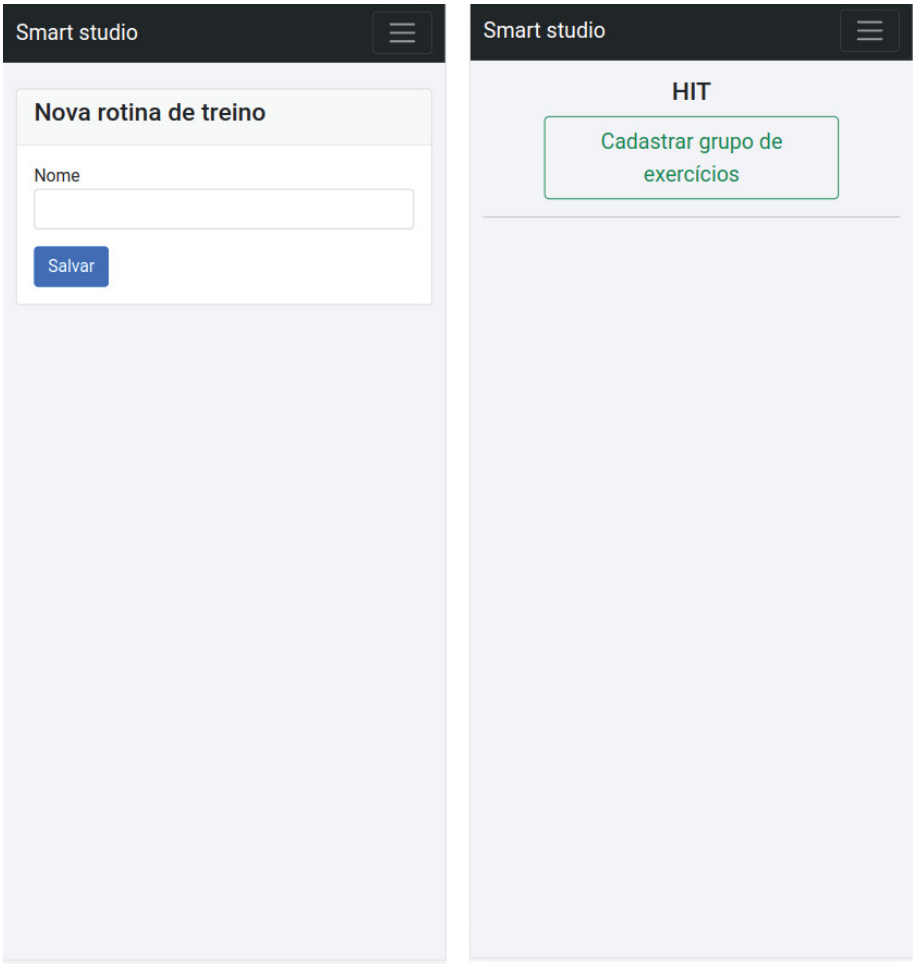
FIGURA 11 – LISTAGEM DE ROTINAS DE TREINO

FONTE - O autor (2022)

O cadastro de uma rotina de treino é feito em duas etapas (FIGURA 12). Inicialmente o usuário define um nome único e obrigatório sendo este tendo sua unicidade validada pela API e a obrigatoriedade validada no cliente web e API.

Em seguida, o usuário deve atribuir as séries de exercícios que compõe a rotina de treino. Isto é feito a partir de uma modal em que o usuário define qual é o exercício, tempo de descanso, número de repetições e a sequência em que este deve ser executado.

FIGURA 12 - CADASTRO DE ROTINA DE TREINO

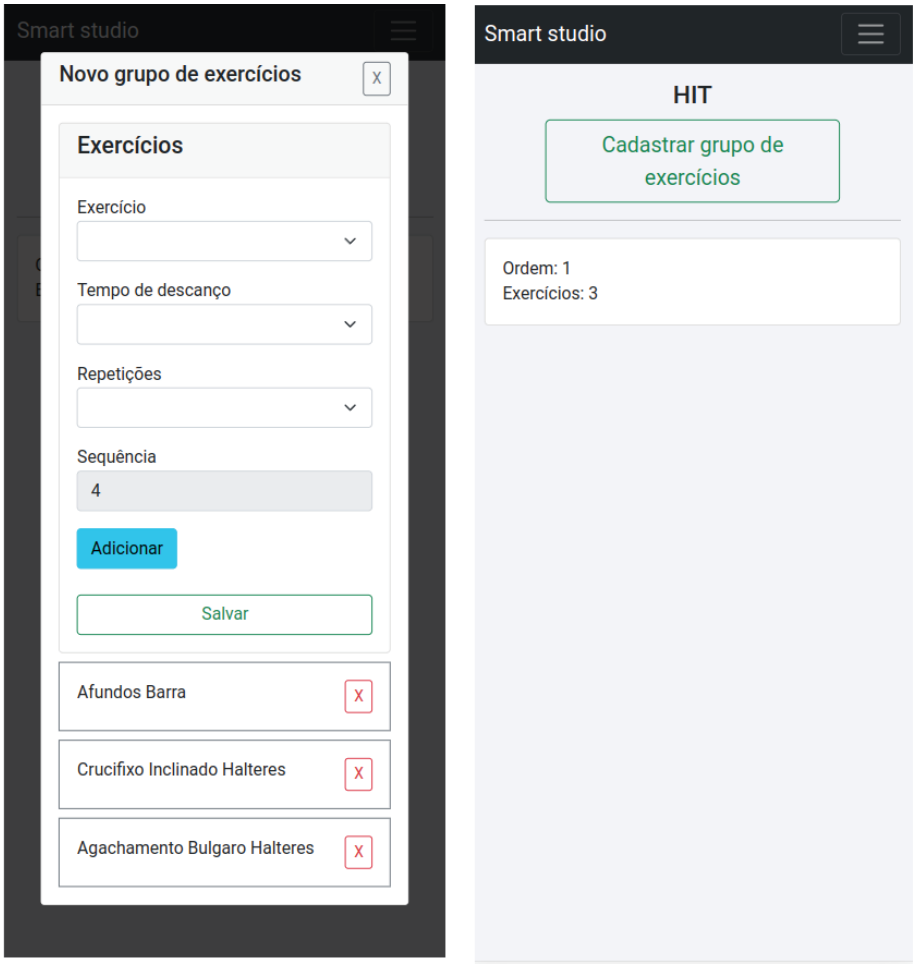


FONTE - O autor (2022)

O usuário pode repetir o processo de adicionar um exercício quantas vezes for necessário, e assim que todos os exercícios forem adicionados o usuário clica no botão de “salvar” para que os dados sejam persistidos no banco de dados.

Como uma rotina de treino é composta por uma ou mais séries de exercícios, o usuário pode clicar no botão "Cadastrar grupo de exercícios", e com isso é iniciado o fluxo para cadastrar uma nova série de exercícios (FIGURA 13).

FIGURA 13 - ATRIBUIÇÃO DE EXERCÍCIO A ROTINA DE TREINO

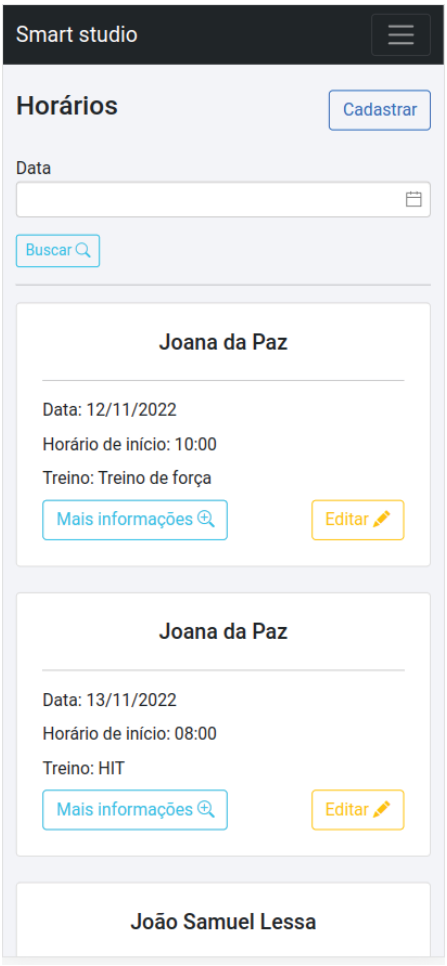


FONTE - O autor (2022)

4.2.5 Módulo horários

O módulo responsável por gerenciar os horários permite a um usuário do tipo *personal trainer* a criação, edição, listagem e visualização de horários. Bem como nos demais módulos ao clicar no respectivo *link*, presente no menu lateral, o usuário é redirecionado para a listagem de horários cadastrados, contendo *links* para cadastrar, editar, visualizar com mais detalhes e buscar por data que filtra apenas os horários agendados para a data buscada (FIGURA 14).

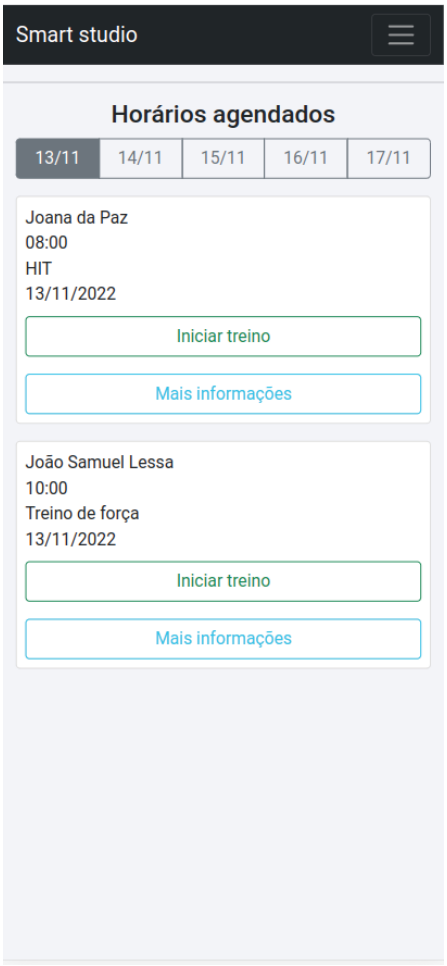
FIGURA 14 - LISTAGEM DE HORÁRIOS



FONTE - O autor (2022)

Por ser uma ação comum no dia a dia de um *personal trainer*, o agendamento e visualização de horários possuem atalhos. Na página inicial existe um componente que exibe os horários agendados para a data atual e os *links* para visualizar os treinos agendados para os próximos quatro dias (FIGURA 15). Também no menu lateral existe um *link* que redireciona diretamente para a página de cadastro de horário, simplificando a navegação para esta página.

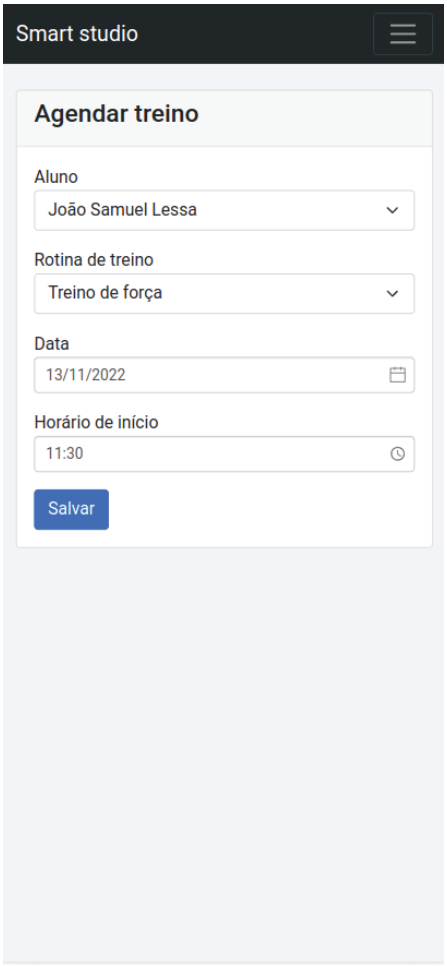
FIGURA 15 - LISTAGEM DE HORÁRIOS NA PÁGINA INICIAL



FONTE - O autor (2022)

O cadastro de horário de treino contém um formulário em que o usuário deve preencher o aluno, a rotina de treino, a data e o horário em que o treino será realizado sendo a obrigatoriedade de todos estes campos validados no cliente web e na API (FIGURA 16).

FIGURA 16 - CADASTRO DE HORÁRIO



FONTE - O autor (2022)

4.2.6 Módulo exercícios

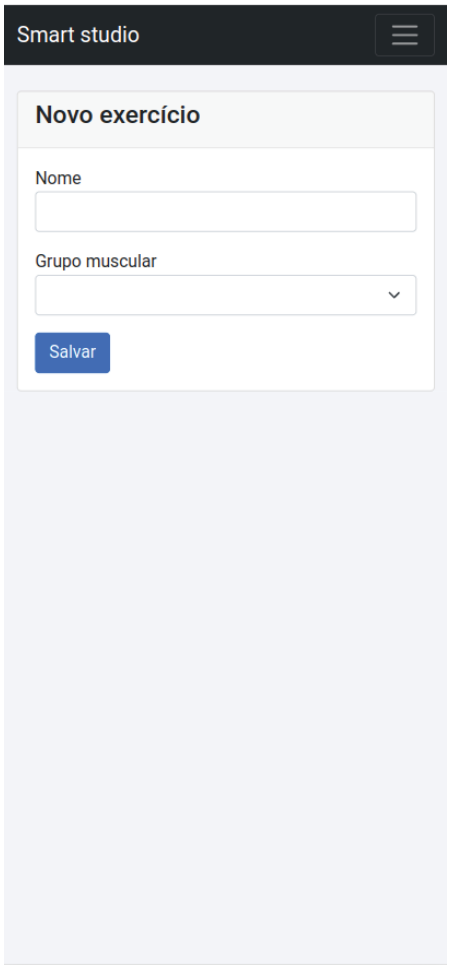
O módulo responsável por gerenciar os exercícios permite que um usuário autenticado do tipo *personal trainer* possa cadastrar, editar, listar, exibir e remover exercícios. Para acessar as funcionalidades referentes aos exercícios, o usuário deve clicar no *link* "Exercícios" presente no menu lateral, e com isso será redirecionado para uma tela com a listagem de usuários, bem como *links* para cadastrar, editar e um campo que filtra exercícios por parte do nome (FIGURA 17).

FIGURA 17 - LISTAGEM DE EXERCÍCIOS

FONTE - O autor (2022)

Para cadastrar um novo exercício o usuário deve preencher o formulário (FIGURA 18) com o nome e selecionar a qual grupo muscular o exercício pertence, ao clicar no botão "Salvar" o próprio cliente web valida a obrigatoriedade dos dados antes de enviá-los para API, além de uma nova validação de obrigatoriedade dos atributos referentes a nome e grupo muscular a unicidade do nome também é validada.

FIGURA 18 - CADASTRO DE EXERCÍCIO



FONTE - O autor (2022)

4.2.7 Módulo planos

O módulo responsável pelo gerenciamento de planos permite que um usuário autenticado do tipo *personal trainer* possa cadastrar, editar, visualizar e listar planos cadastrados aplicando ou não uma filtragem por parte do nome.

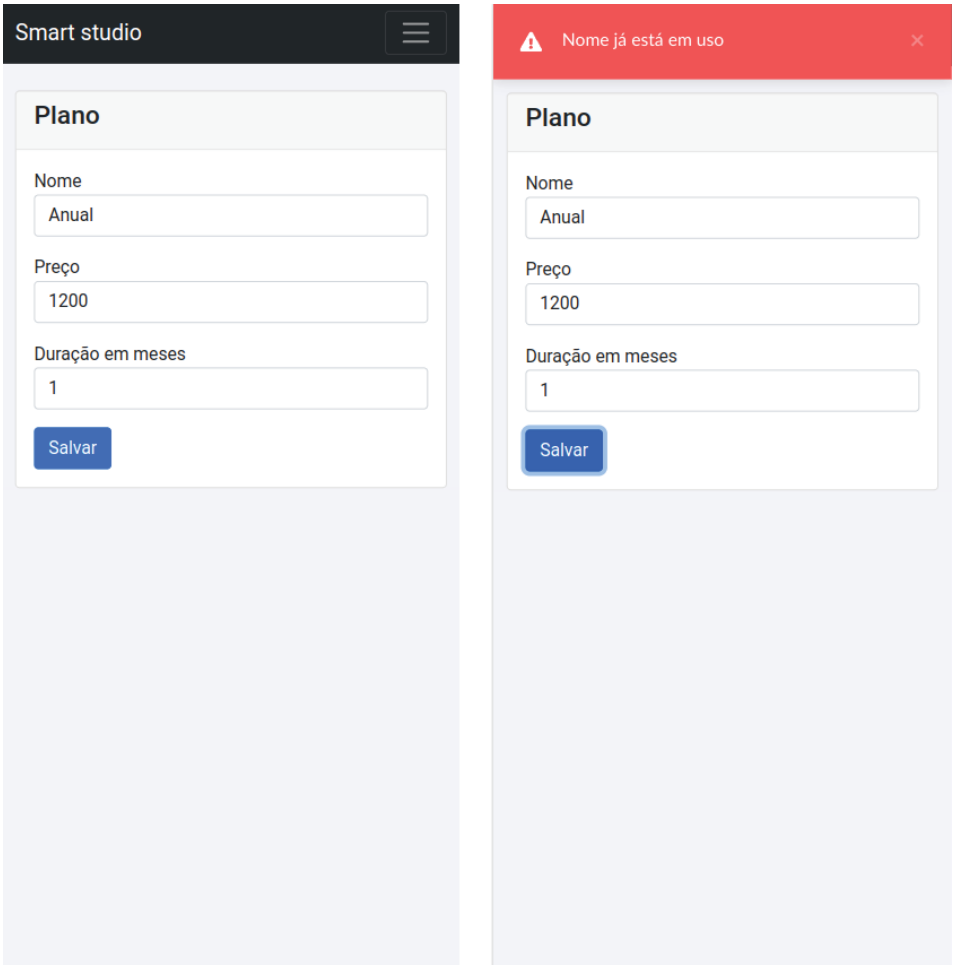
O acesso as funcionalidades relativas ao gerenciamento de planos são feito através do link "Planos" presente no menu lateral. Ao clicar no *link* o usuário é redirecionado para uma tela contendo uma lista com os planos cadastrados e *links* para navegar aos formulários de cadastro e edição de plano bem como *links* para visualizar um plano detalhadamente (FIGURA 19).

FIGURA 19 - LISTAGEM DE PLANOS

FONTE: O autor (2022)

O formulário de cadastro de plano possui os campos nome, preço e duração em meses, sendo todos estes obrigatórios e validados no cliente web e API. A unicidade do nome é validada apenas na API e caso já exista aparece uma mensagem indicando o erro (FIGURA 20).

FIGURA 20 - CADASTRO DE PLANO



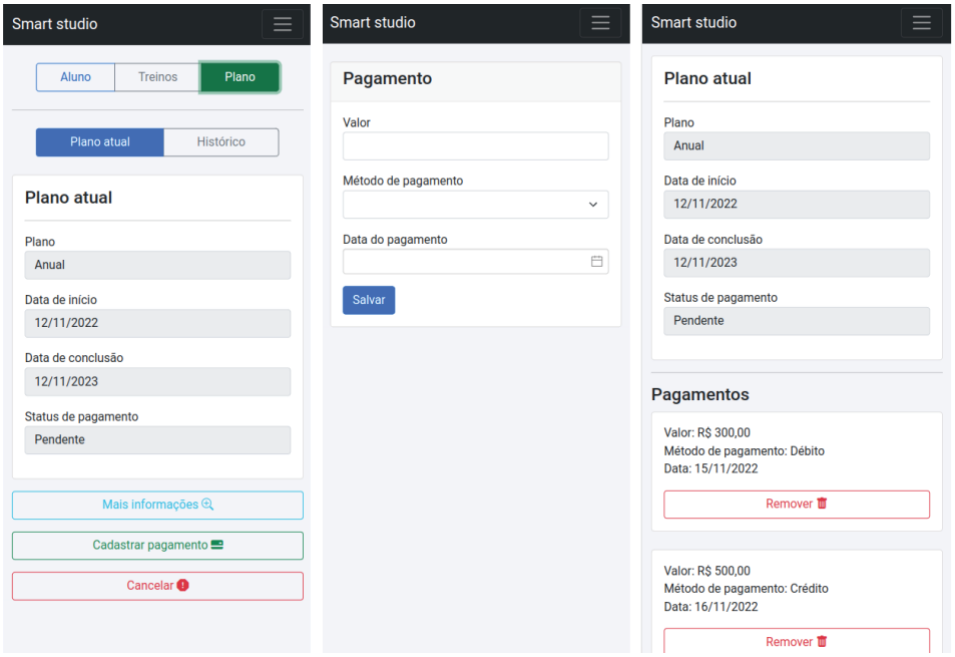
FONTE: O autor (2022)

4.2.8 Módulo pagamentos

O módulo de pagamentos tem como objetivo permitir o cadastro e a visualização de pagamentos recebidos por estudantes. Diferente dos demais módulos as funcionalidades referentes a pagamentos não são acessadas através do menu lateral, mas sim através das telas de exibição de aluno e plano uma vez que estes estão relacionados.

O acesso ao formulário de cadastro de pagamento é feito através da tela de perfil do aluno, ao navegar para sessão plano existe um *link* "Cadastrar pagamento", ao clicar no *link* o usuário é redirecionado para o formulário em que deve preencher o valor, selecionar o método de pagamento e data (FIGURA 21).

FIGURA 21 - CADASTRO DE PAGAMENTO



FONTE: O autor (2022)

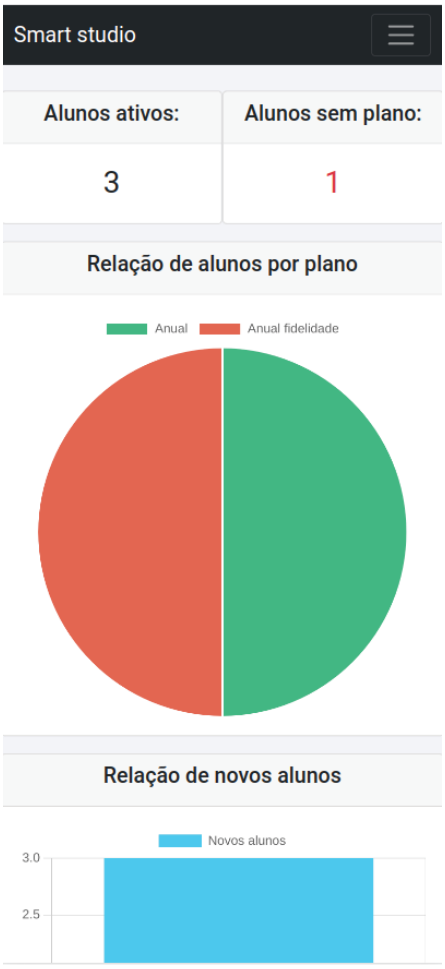
Ao cadastrar um pagamento com sucesso, o usuário é redirecionado para tela em que é exibido o plano atual do aluno bem com os pagamentos referentes a este plano. A outra forma de navegar a página que exibe o plano atual e pagamentos realizados pelo aluno é clicando no *link* "Mais informações" presente na sessão "Plano" no perfil do aluno.

4.2.9 Módulo relatórios

O módulo de relatórios disponibiliza dois *dashboards* em que usuários do tipo *personal trainer* pode visualizar seus dados de forma agrupada, auxiliando na tomada de decisões estratégicas voltadas ao marketing para captação de novos alunos ou melhora da retenção de alunos.

Os *dashboards* são acessados através do menu lateral através dos *links* "Relatórios de alunos" e "Relatórios financeiros". O *dashboard* "Relatório de alunos" (FIGURA 22) exibe o total de alunos cadastrados, o total de alunos que não possuem plano ativo, a relação de alunos por tipo de plano e a relação de novos alunos por mês.

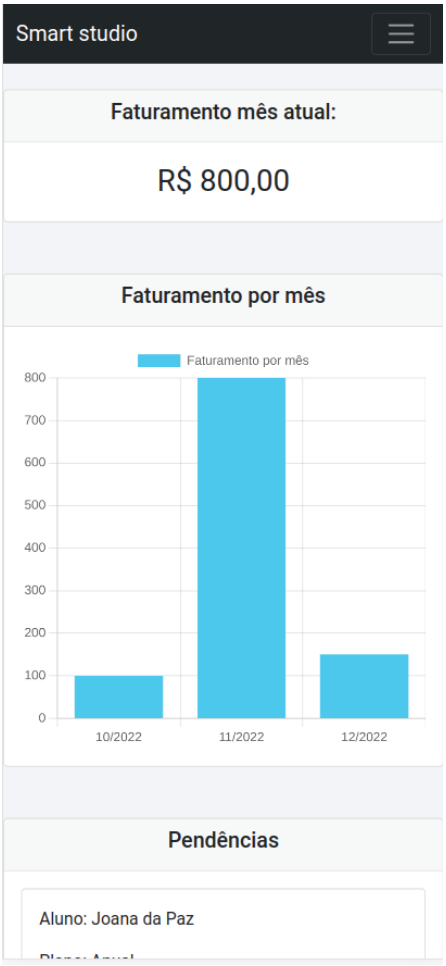
FIGURA 22 - RELATÓRIOS DE ALUNOS



FONTE: O autor (2022)

Já o *dashboard* "Relatórios financeiros" mostra ao usuário o valor total de faturamento no mês atual, a relação faturamento por mês e uma lista com todos os planos que estão em andamento e não estão quitados (FIGURA 23).

FIGURA 23 - RELATÓRIOS FINANCEIROS



FONTE: O autor (2022)

5 CONSIDERAÇÕES FINAIS

Este documento apresentou o Smart Studio, uma plataforma criada para facilitar o gerenciamento de estúdios de musculação disponibilizando funcionalidades que buscam simplificar a criação de rotinas de treino, agendamento de horários, controle de planos de mensalidade, cadastro de alunos, pagamentos e visualização dos dados na forma de *dashboards*.

Durante o andamento do projeto fez-se uso de metodologia ágil de desenvolvimento de software, de forma que as funcionalidades planejadas foram divididas em tarefas pequenas possibilitando um processo contínuo de pequenas entregas de forma constante.

O Smart Studio é composto por duas aplicações, sendo estas uma API desenvolvido em *Ruby on Rails* e um cliente web desenvolvido em *Vue.js*, em que ambos os projetos seguiram as boas práticas de desenvolvimento de software buscando entregar um código que seja relativamente simples de manter e evoluir.

Na seção seguinte são apresentadas as recomendações de trabalhos futuros.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Ao longo do desenvolvimento do projeto foram notadas oportunidades de novas funcionalidades e melhorias, que juntas podem melhorar a experiência do usuário e agregar mais valor a aplicação. Em que as oportunidades incluem:

- Acesso utilizando um perfil do tipo “aluno” em que este poderia visualizar seus dados, treinos agendados, plano atual, pendências e pagamentos realizados;
- Cadastrar valores de carga utilizados durante os treinos, dessa forma sendo possível visualizar a progressão do aluno;
- Melhorar a usabilidade aplicando o recurso de animações disponibilizado nativamente pelo Vue.js em etapas de carregamento ou transições entre telas;
- Adaptar a tela para dispositivos maiores do que celulares, uma vez que a aplicação foi desenvolvida priorizando este grupo de dispositivos, já que os profissionais consultados utilizavam apenas *smartfones* durante seu dia a dia.

REFERÊNCIAS

BECK, Kent et al, Manifesto for Agile Software Development, 2001. Disponível em: <https://agilemanifesto.org/>. Acesso em: 5 de nov. de 2022.

BECK, Kent, Programação Extrema (Xp) Explicada. Porto Alegre: Bookman, 2004.

BOOCH, G.; RUMBAUGH, J. UML: guia do usuário. Rio de Janeiro: Elsevier, 2006.

BUSTAMENTE, Fernando. Você sabe a diferença entre academia, personal trainer e estúdio de treinamento. Doctorfir, 2018. Disponível em: <https://www.doctorfit.com.br/blog/voce-sabe-a-diferenca-entre-academia-personal-trainer-e-estudio-de-treinamento/>. Acesso em: 26 de ago. 2022.

CODE. Visual Studio Code in Action. Disponível em: <https://code.visualstudio.com/>. Acesso em: 30/10/2022.

DOCKER. We're excited that you want to learn Docker. Disponível em: <https://docs.docker.com/get-started/>. Acesso em: 30/10/2022.

EVANGELISTA Alexandre, L.; Braz, Tiago V.; TEIXEIRA, Cauê V. S.; RICA, Roberta L.; ALONSO, Angélica C.; BARBOSA, Welmo A.; REIS, Victor M.; BAKER, Julien S.; SCHOENFELD, Brad J.; BOCALINI, Danilo S.; GREVE, Julia M. A. Rotina de treinamento distribuída ou de corpo inteiro: qual a melhor estratégia para aumentar força e hipertrofia muscular. Einstein, 2021. Disponível em: <https://journal.einstein.br/wp-content/uploads/articles_xml/2317-6385-eins-19-eAO5781/2317-6385-eins-19-eAO5781-pt.pdf?x56956>. Acesso em: 14 maio. 2022.

FILARDO, Ronaldo D.; LEITE, Neiva. Perfil dos indivíduos que iniciam programas de exercício em academias, quanto a composição corporal e aos objetivos em relação a faixa etária e sexo. Revista brasileira de medicina do exporte, 2001. Disponível em: <https://www.scielo.br/j/rbme/a/CMQYDcfpHfGbh3hpjjJDRQF/?format=pdf&lang=pt>. Acesso em: 14, maio. 2022

FUENTES, Vinícius Baggio. Ruby On Rails: Coloque sua aplicação web nos trilhos. São Paulo: Casa do Código, 2010.
GIT is a free and open-source distributed version. Disponível em: <https://git-scm.com/>. Acesso em: 30/10/2022.

GITHUB. Let's build from here. Disponível em: <https://github.com/about>. Acesso em: 30/10/2022.

JUDGE, Lawrence; BURKE, Jeanmarie. The Efffect of Recovery Time on Strength Performance Following a High Intensity Bench Press Workout in Males and Females. International journal of sports physiology and performance. 2010, Disponível em: <http://www.researchgate.net/publication/45167587_The_Effect_of_Recovery_Time_on_Strength_Performance_Following_a_High-Intensity_Bench_Press_Workout_in_Males_and_Females>, Acesso em: 14, maio. 2022.

LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo, Porto Alegre: Bookman, 2007.

LIZ, Carla M.; ANDRADE, Alexandre. Análise qualitativa dos motivos e adesão e desistência musculação em academias. Revista Brasileira de Ciência do Esporte. 2015. Disponível em: <https://www.scielo.br/j/rbce/a/gffmp7zVjZgBChtQfnKpGnz/?lang=pt&format=pdf>. Acesso em: 14 maio. 2022

MACEDO, Christiane S. G.; GARAVELLO, João J.; OKU, Elaine C.; MIYAGUSUKU, Fábio H.; AGNOLL, Priscila D.; NOCETTI, Priscila M. Benefícios do exercício físico para qualidade de vida. Revista Brasileira de Atividade Física & Saúde. Disponível em: <https://rbafs.org.br/RBAFS/article/view/875>. Acesso em: 14 maio. 2022.

MARTIN, R. C. Código limpo: Habilidades práticas do Agile Software. Rio de Janeiro: Alta Books, 2009.

PANTA, Regiane; MATHIAS, Ricardo, J; FILHO, José N. S. Efeitos do treinamento resistido personalizado na composição corporal de homens adultos. Revista Brasileira de Obesidade, v. 11, p. 591-597. Disponível em: <https://dialnet.unirioja.es/servlet/articulo?codigo=6301495>. Acesso em: 14 de maio 2022.

PARENTE, Cristiano. Studio ou academia: entenda as diferenças e faça sua escolha. Minhavida fitness, 2022. Disponível em: <https://www.minhavida.com.br/materias/materia-18819>. Acesso em: 26 de ago. 2022.

POSTGRESQL. The World's Most Advanced Open-Source Relational Database. Disponível em: <https://www.postgresql.org/>. Acesso em: 30/10/2022.

RADIGAN, Dan. Quatro cerimônias ágeis desmistificadas. Disponível em: <https://www.atlassian.com/br/agile/scrum/ceremonies >. Acesso em: 30/10/2022.
REVELO. Desenvolvedores: frameworks e linguagens para aprender em 2022. Blog da Revelo, 2022. Disponível em: <https://blog.revelo.com.br/desenvolvedores-linguagens-2022/>. Acesso em: 16/11/2022

RUBY. About Ruby. Disponível em: <https://www.ruby-lang.org/en/about/>. Acesso em 30/10/2022.

RUBYCRITIC. A ruby code quality reporter, 2011. Disponível em: <https://github.com/whitesmith/rubycritic>

RUBY ON RAILS. Compress the complexity of modern web apps. Disponível em: <https://rubyonrails.org/>. Acesso em: 30/10/2022.

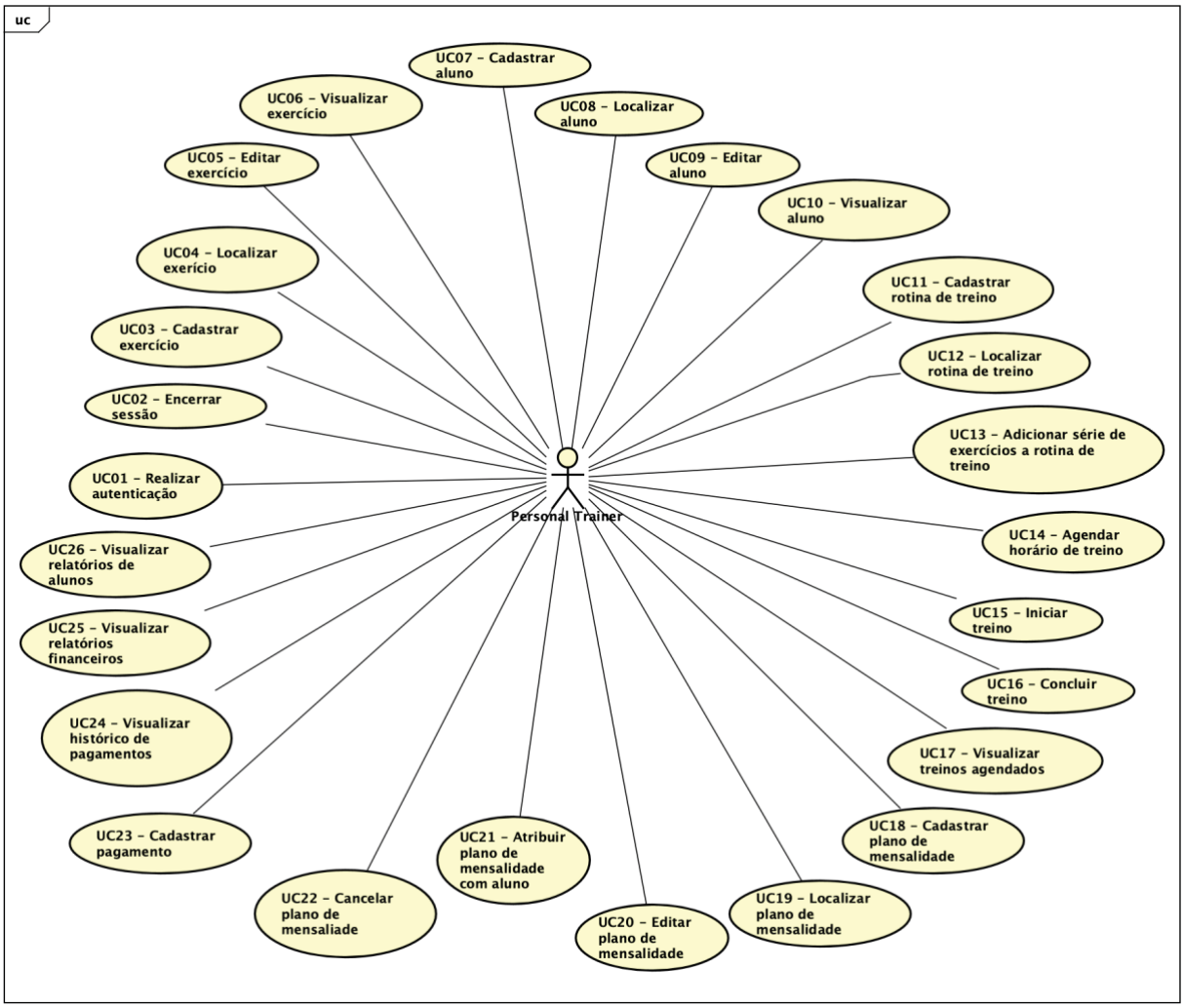
SCHWABER, Ken; BEEDLE, Mike. Agile Software Development with Scrum. São Paulo: Importado Pearson, 2001.

VUE. The progressive JavaScript Framework.Disponível em: <https://vuejs.org/>. Acesso em: 30/10/2022.

WIGGINS, Adam. The Twelve-Factor App, 2013. Disponível em: <https://12factor.net/>. Acesso em: 8 de nov. De 2020.

APÊNDICE A – DIAGRAMA DE CASO DE USO

FIGURA 24 - DIAGRAMA DE CASOS DE USO



FONTE: O autor (2022)

APÊNDICE B – HISTÓRIAS DE USUÁRIO

UC01 - Realizar autenticação	**SENDO** um *personal trainer* **QUERO** acessar o sistema **PARA** gerenciar um estúdio de musculação **CRITÉRIOS DE ACEITAÇÃO** 1. Deve permitir inserir usuário e senha; 2. Não acessar o sistema caso e-mail e senha não pertençam a um usuário; 3. O usuário deve ser redirecionado para página inicial em caso de sucesso.
UC02 - Encerrar sessão	**SENDO** um *personal trainer* **QUERO** sair do sistema **PARA** que deixe de utilizar o sistema **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve ser redirecionado para página de acesso; 2. O usuário deve ser impossibilitado de acessar ou realizar ações até que se autentique novamente.
UC03 - Cadastrar exercício	**SENDO** um *personal trainer* **QUERO** cadastrar um novo exercício **PARA** vinculá-lo a uma rotina de treino **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, e selecionar o grupo muscular trabalhado; 3. Não deve permitir que exercícios com mesmo nome e que pertençam ao mesmo *personal trainer* sejam cadastrados; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 6. Deve redirecionar o usuário a página que exibe os dados do novo exercício.

UC04 - Localizar exercício	**SENDO** um *personal trainer* **QUERO** localizar um exercício **PARA** poder visualizar ou editar seu registro **CRITÉRIOS DE ACEITAÇÃO** 1. Deve permitir busca por parte do nome; 2. Deve exibir uma lista paginada; 3. Deve exibir apenas os exercícios pertencentes ao usuário atual; 4. Deve exibir *link* para navegar à página que edita um exercício; 5. Deve exibir *link* para navegar à página que exibe os dados de um exercício; 6. O usuário deve estar autenticado.
UC05 - Editar exercício	**SENDO** um *personal trainer* **QUERO** editar um exercício **PARA** corrigir um exercício cadastrado com dados válidos, porém incorretos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, e selecionar o grupo muscular; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. O exercício deve pertencer ao usuário atual; 6. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 7. Deve redirecionar o usuário a página que exibe os dados do exercício.
UC06 - Visualizar exercício	**SENDO** um *personal trainer* **QUERO** visualizar os dados de um exercício **PARA** verificar se estão corretos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O exercício deve pertencer ao usuário atual; 3. Deve exibir uma mensagem de erro caso o id informado não pertença a nenhum exercício cadastrado; 4. Deve exibir uma mensagem de erro caso o id informado pertença a um exercício que não pertence ao usuário dono da sessão atual.

UC07 - Cadastrar aluno	**SENDO** um *personal trainer* **QUERO** cadastrar um novo aluno **PARA** agendar treinos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, e selecionar objetivo; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 6. Deve redirecionar o usuário a página que exibe os dados do novo aluno.
UC08 - Localizar aluno	**SENDO** um *personal trainer* **QUERO** localizar o perfil de um aluno **PARA** verificar seus dados, treinos agendados e históricos de planos de mensalidade e pagamentos **CRITÉRIOS DE ACEITAÇÃO** 1. Deve permitir busca por parte do nome; 2. Deve exibir uma lista paginada; 3. Deve exibir apenas os alunos pertencentes ao usuário atual; 4. Deve exibir *link* para navegar à página que edita um aluno; 5. Deve exibir *link* para navegar à página que exibe os dados de um aluno; 6. O usuário deve estar autenticado.
UC09 – Editar aluno	**SENDO** um *personal trainer* **QUERO** editar os dados de um aluno **PARA** corrigir os dados de um aluno cadastrado com dados válidos, porém incorretos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, e selecionar objetivo; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. O aluno deve pertencer ao usuário atual; 6. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 7. Deve redirecionar o usuário a página que exibe os dados do aluno.

UC10 - Visualizar aluno	**SENDO** um *personal trainer* **QUERO** visualizar os dados de um aluno **PARA** acompanhar os dados de perfil, visualizar históricos de treinos agendados, planos de mensalidades e pagamentos realizados **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O aluno deve pertencer ao usuário atual; 3. Deve exibir uma mensagem de erro caso o id informado não pertença a nenhum aluno cadastrado; 4. Deve exibir uma mensagem de erro caso o id informado pertença a um aluno que não pertence ao usuário dono da sessão atual; 5. Deve exibir os dados de perfil do aluno; 6. Deve exibir o histórico de treinos agendados e realizados; 7. Deve exibir histórico de planos de mensalidade; 8. Deve exibir histórico de pagamentos realizados.
UC11 - Cadastrar rotina de treino	**SENDO** um *personal trainer* **QUERO** cadastrar uma nova rotina de treino **PARA** que possa agendar treinos com alunos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 6. Deve redirecionar o usuário a página que contém o *link* para cadastrar séries de exercícios.

UC12 - Localizar rotina de treino	**SENDO** um *personal trainer* **QUERO** localizar uma rotina de treino **PARA** vincular novas séries de exercícios ou editar seus dados **CRITÉRIOS DE ACEITAÇÃO** 1. Deve permitir busca por parte do nome; 2. Deve exibir uma lista paginada; 3. Deve exibir apenas rotinas de treino pertencentes ao usuário atual; 4. Deve exibir *link* para navegar à página que edita uma rotina de treino; 5. Deve exibir *link* para navegar à página que exibe os dados de uma rotina de treino; 6. O usuário deve estar autenticado.
UC13 - Adicionar série de exercício a rotina de treino	**SENDO** um *personal trainer* **QUERO** adicionar séries de exercícios a uma rotina de treino **PARA** planejar rotinas de treino e aplicá-las em sessões de treino **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O formulário deve ser renderizado em uma modal; 3. Deve permitir selecionar exercício, tempo de descanso e número de repetições; 4. Deve exibir a ordem de execução que será atribuída ao exercício; 5. Deve gerar o número referente a ordem de execução automaticamente; 6. Deve permitir cadastrar um grupo de exercícios; 7. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 8. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 9. Deve fechar a modal com o formulário em caso de sucesso.

UC14 - Agendar horário de treino	**SENDO** um *personal trainer* **QUERO** agendar um horário de treino **PARA** realizar um treino com um aluno **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir selecionar aluno, rotina de treino, data e horário de início; 3. Deve listar apenas alunos que pertençam ao usuário atual; 4. Deve listar apenas rotinas de treino que pertençam ao usuário atual; 5. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 6. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 7. Deve redirecionar o usuário a página inicial.
UC15 - Iniciar treino	**SENDO** um *personal trainer* **QUERO** alterar o status de um treino agendado para em andamento **PARA** visualizar os exercícios planejados para o treino **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O horário de treino deve pertencer ao usuário atual; 3. Deve alterar o status do horário de treino para “em progresso”; 4. Deve exibir uma mensagem de sucesso caso a ação tenha sucesso; 5. Deve redirecionar o usuário à tela que exibe as séries de exercícios pertencentes ao treino iniciado.
UC16 - Concluir treino	**SENDO** um *personal trainer* **QUERO** concluir um treino em andamento **PARA** que o *link* para o treino concluído deixe de aparecer na página inicial **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O horário de treino deve pertencer ao usuário atual; 3. Deve exibir uma mensagem de sucesso caso a ação tenha sucesso; 4. O treino selecionado deve conter status “em progresso”; 5. O status do horário de treino deve ser alterado para “concluído”; 6. O usuário deve ser redirecionado para página inicial caso a ação tenha sucesso.

UC17 - Visualizar treinos agendados	**SENDO** um *personal trainer* **QUERO** visualizar meus treinos agendados **PARA** organizar como irei ministrar os treinos durante o dia **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Os treinos listados devem pertencer ao usuário atual; 3. Devem ser listados treinos agendados para data atual; 4. O usuário deve poder visualizar treinos agendados até cinco dias após a data atual.
UC18 - Cadastrar plano de mensalidade	**SENDO** um *personal trainer* **QUERO** cadastrar plano de mensalidade **PARA** organizar o faturamento do meu estúdio de musculação **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, preço e duração em meses; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. Deve exibir uma mensagem de sucesso caso o registro tenha sido criado com sucesso; 6. Deve redirecionar o usuário a página que exibe os dados do novo plano.
UC19 - Localizar plano de mensalidade	**SENDO** um *personal trainer* **QUERO** localizar um plano de mensalidade **PARA** organizar os planos de mensalidade oferecidos atualmente **CRITÉRIOS DE ACEITAÇÃO** 1. Deve exibir uma lista paginada; 2. Deve exibir apenas planos de mensalidade de treino pertencentes ao usuário atual; 3. Deve exibir *link* para navegar à página que edita plano de mensalidade; 4. Deve exibir *link* para navegar à página que exibe os dados de plano de mensalidade; 5. O usuário deve estar autenticado.

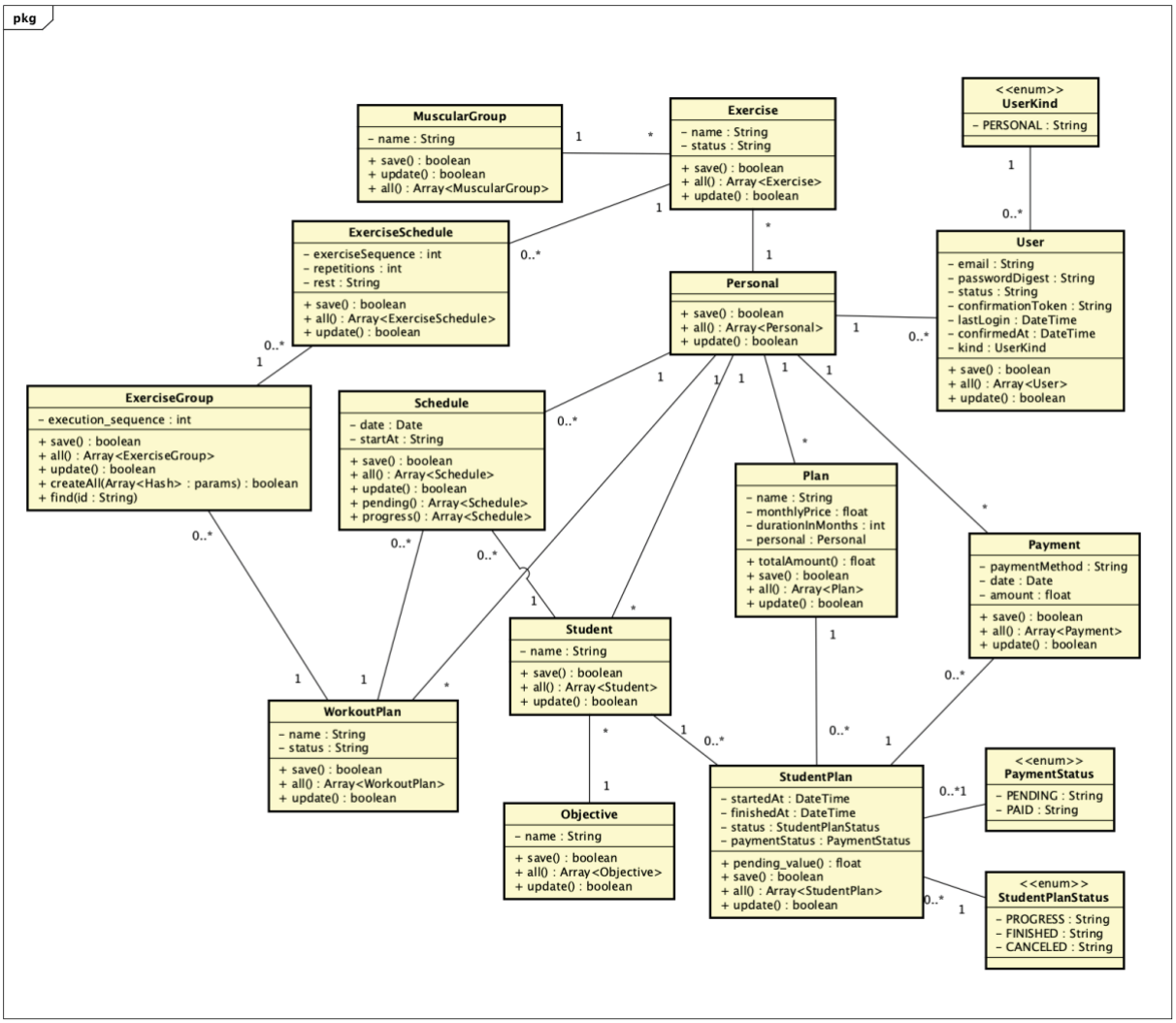
UC20 - Editar plano de mensalidade	**SENDO** um *personal trainer* **QUERO** editar um plano de mensalidade **PARA** reajustar o preço **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o nome, preço e duração em meses; 3. Deve validar unicidade do atributo nome; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. O aluno deve pertencer ao usuário atual; 6. Deve exibir uma mensagem de sucesso caso o registro tenha sido editado com sucesso; Deve redirecionar o usuário a página que exibe os dados de plano.
UC21 - Atribuir plano de mensalidade com aluno	**SENDO** um *personal trainer* **QUERO** atribuir um plano de mensalidade à um aluno **PARA** controlar o faturamento de meu estúdio **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir selecionar um plano de mensalidade e data de início do plano; 3. Um aluno não pode possuir mais de um plano ativo; 4. Deve exibir uma mensagem de erro em caso de tentativa de envio do formulário sem o preenchimento dos dados obrigatórios; 5. O aluno deve pertencer ao usuário atual; 6. Os planos de mensalidade listados devem pertencer ao usuário atual; 7. Deve exibir uma mensagem de sucesso caso o registro tenha sido editado com sucesso; 8. Deve redirecionar o usuário a página que exibe os dados de plano.

UC22 - Cancelar plano de mensalidade	**SENDO** um *personal trainer* **QUERO** cancelar o plano atual de um aluno **PARA** atribuir um novo plano ao aluno **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O aluno deve pertencer ao usuário atual; 3. O aluno deve pertencer ao usuário atual; 4. Deve exibir uma mensagem de sucesso caso o registro tenha sido editado com sucesso; 5. Deve redirecionar o usuário a página que vincula um aluno a um plano.
UC23 - Cadastrar pagamento	**SENDO** um *personal trainer* **QUERO** cadastrar um pagamento **PARA** ter controle do faturamento do estúdio e das pendências de alunos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Deve permitir preencher o valor, data do pagamento e método de pagamento; 3. O aluno deve pertencer ao usuário atual; 4. O plano não pode estar com status de pagamento “pago”; 5. Deve exibir uma mensagem de sucesso caso o registro tenha sido editado com sucesso; 6. Deve redirecionar o usuário a página que exibe o histórico de pagamentos relacionados ao plano atual do aluno.
UC24 - Visualizar histórico de pagamentos	**SENDO** um *personal trainer* **QUERO** visualizar o histórico de pagamentos referente ao plano de um aluno **PARA** verificar se existe alguma pendência **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. O aluno deve pertencer ao usuário atual; 3. Deve exibir uma lista com os pagamentos vinculados ao plano exibido na tela atual;

UC25 - Visualizar relatórios financeiros	**SENDO** um *personal trainer* **QUERO** visualizar dados financeiros agregados **PARA** avaliar como está a saúde financeira de meu estúdio **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Os dados utilizados na geração dos relatórios devem pertencer ao usuário atual; 3. Deve exibir o valor total de faturamento no mês atual; 4. Deve exibir o faturamento por mês para os meses pertencentes ao ano atual; 5. Deve exibir uma lista com os alunos que possuem planos em andamento, porém que não estejam quitados.
UC26 - Visualizar relatórios de alunos	**SENDO** um *personal trainer* **QUERO** visualizar os dados de alunos agregados **PARA** avaliar o estado atual e visualizar oportunidades de crescimento do número de alunos **CRITÉRIOS DE ACEITAÇÃO** 1. O usuário deve estar autenticado; 2. Os dados utilizados na geração dos relatórios devem pertencer ao usuário atual; 3. Deve exibir o número total de alunos cadastrados; 4. Deve exibir o número total de alunos que não possuem um plano ativo; 5. Deve exibir a relação de alunos por plano de mensalidade; 6. Deve exibir a relação de novos alunos por mês, para os meses pertencentes ao ano atual.

APÊNDICE C – DIAGRAMA DE CLASSES

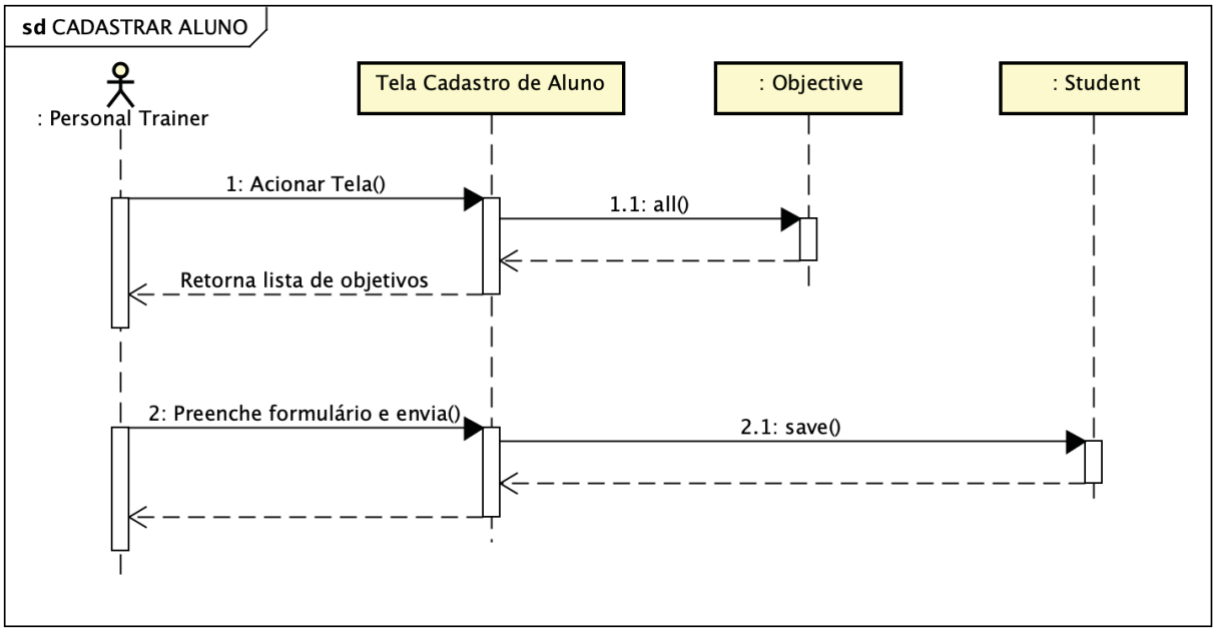
FIGURA 25 - DIAGRAMA DE CLASSES



FONTE: O autor (2022)

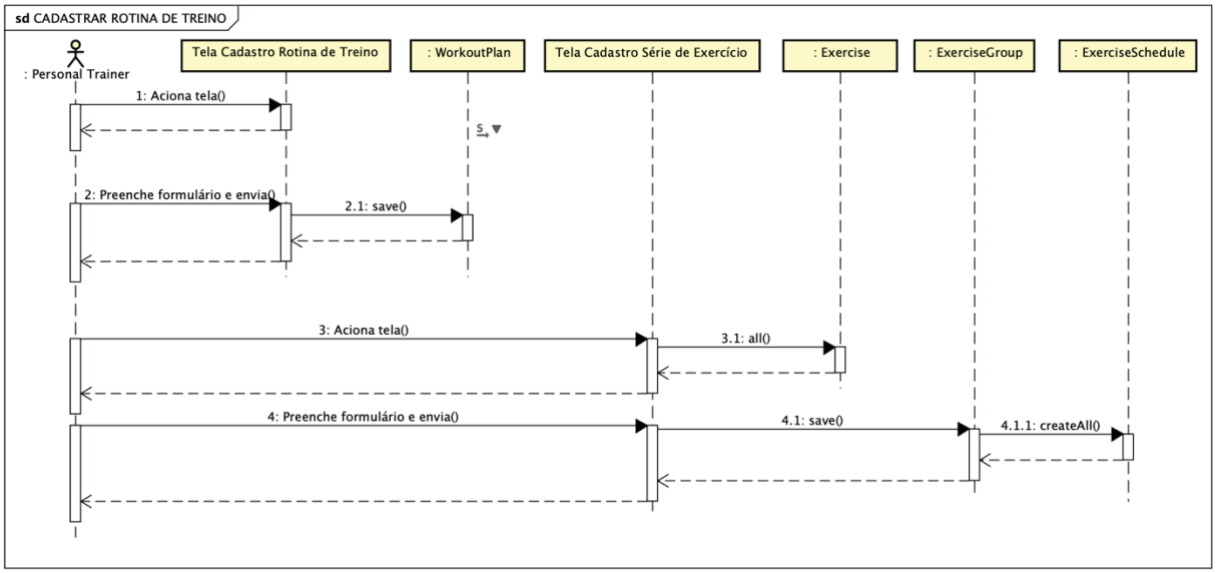
APÊNDICE D – DIAGRAMAS DE SEQUÊNCIA

FIGURA 26 - DIAGRAMA DE SEQUÊNCIA CADASTRAR ALUNO



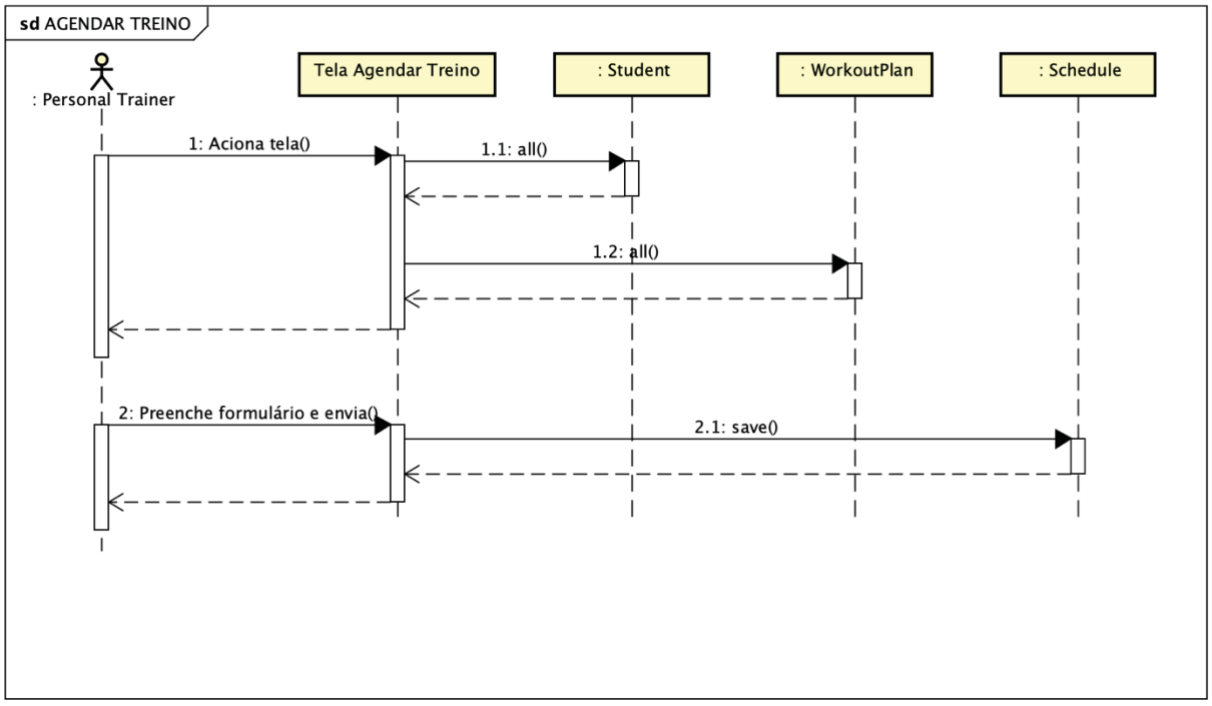
FONTE: O autor (2022)

FIGURA 27 - DIAGRAMA DE SEQUÊNCIA CADASTRAR ROTINA DE TREINO



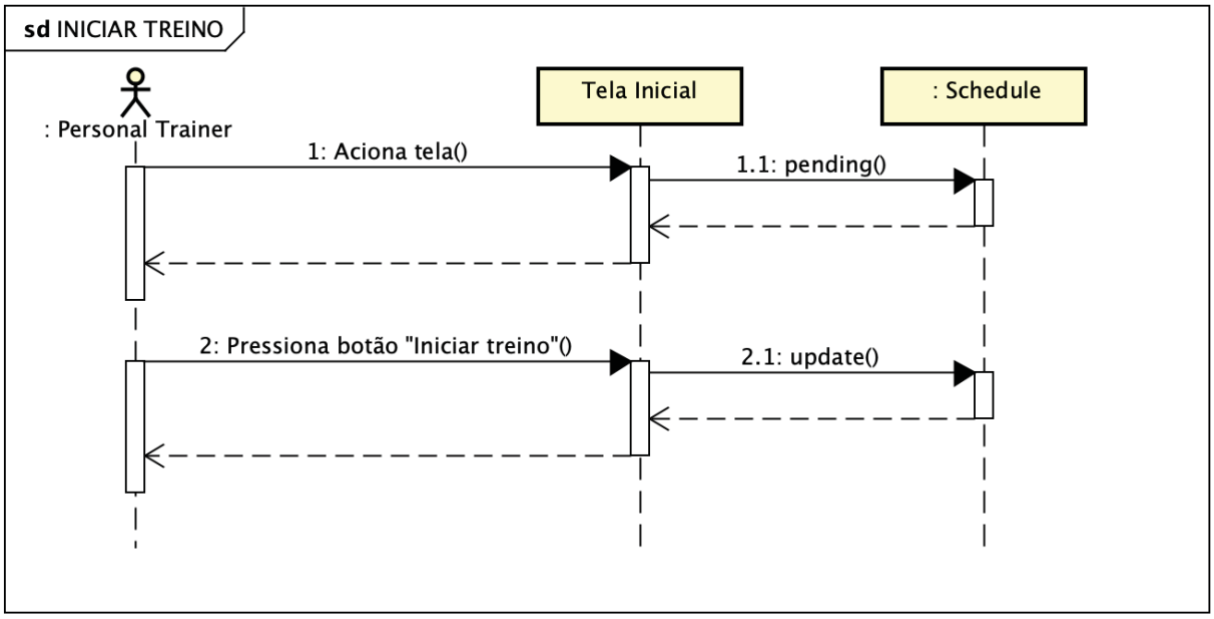
FONTE: O autor (2022)

FIGURA 28 - DIAGRAMA DE SEQUÊNCIA AGENDAR TREINO



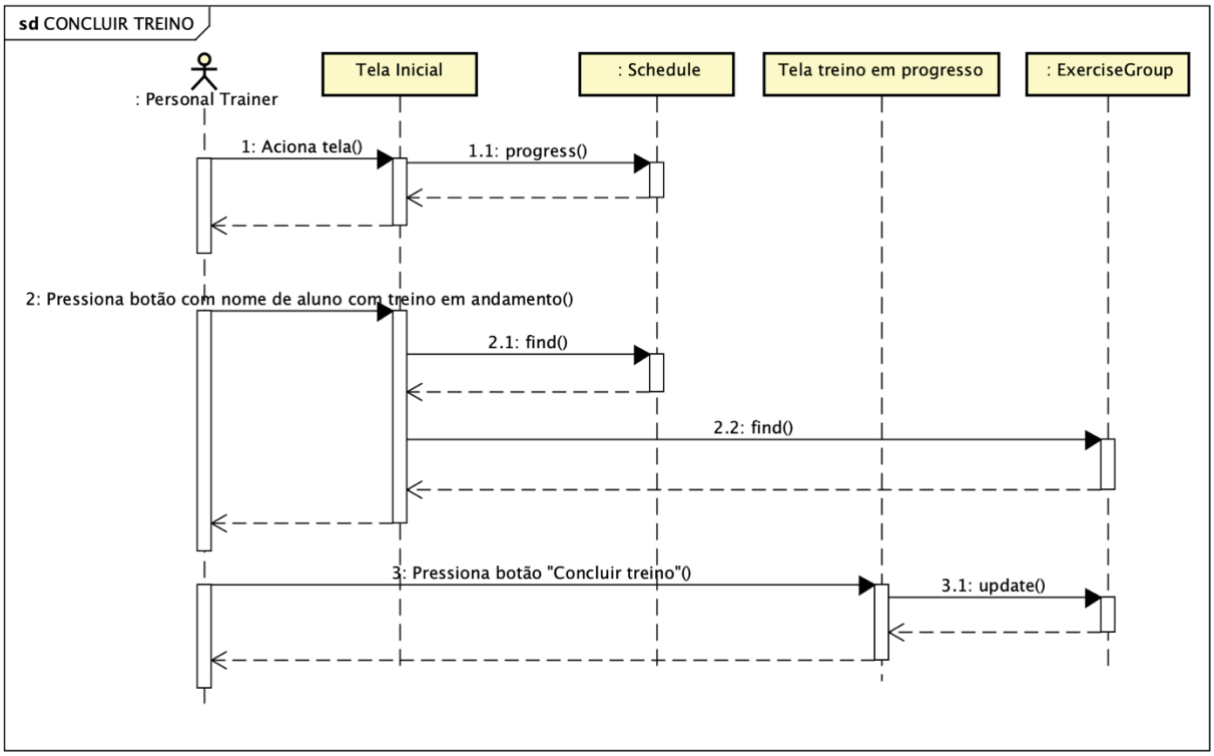
FONTE: O autor (2022)

FIGURA 29 - DIAGRAMA DE SEQUÊNCIA INICIAR TREINO



FONTE: O autor (2022)

FIGURA 30 - DIAGRAMA DE SEQUÊNCIA CONCLUIR TREINO

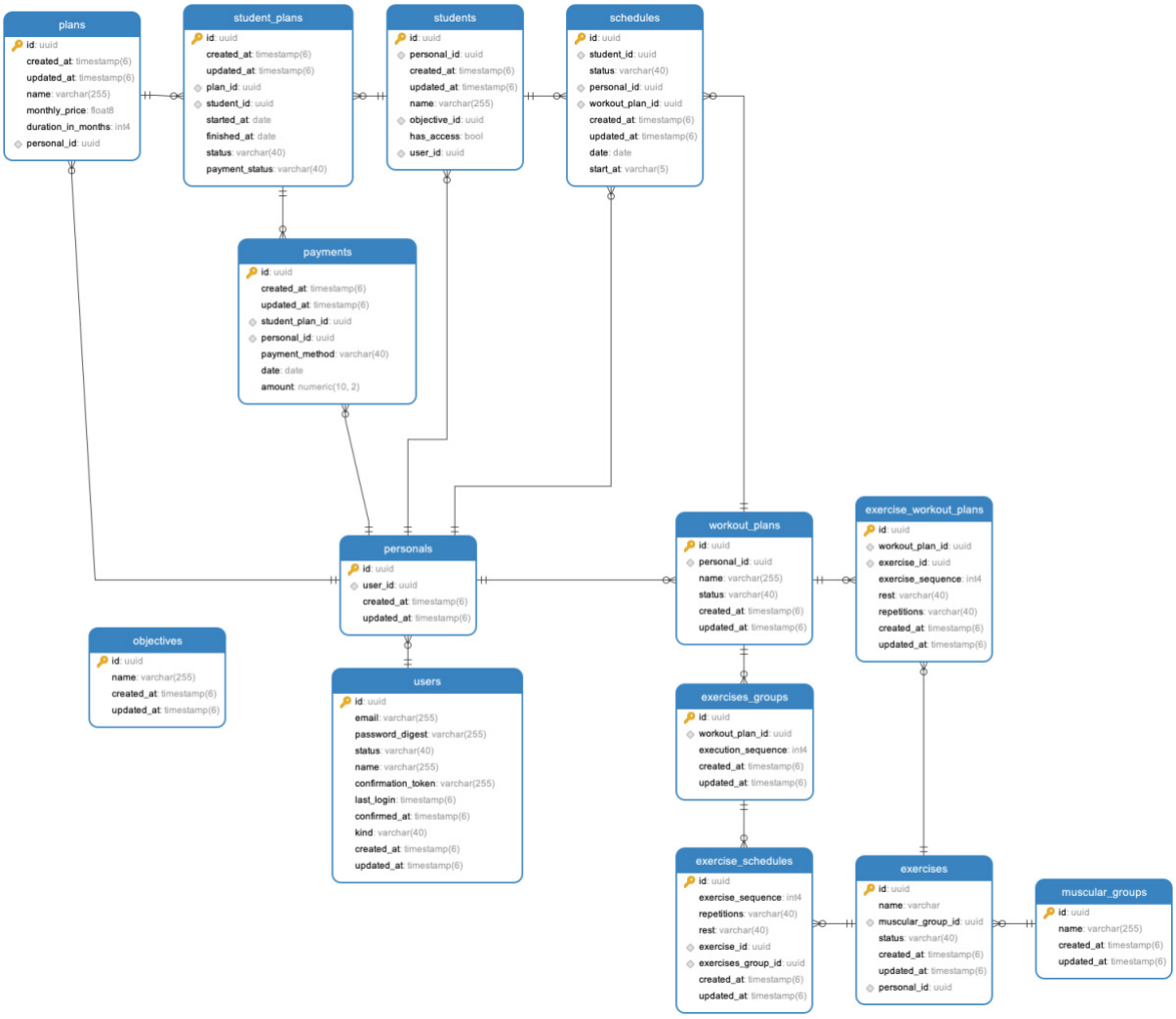


FONTE: O autor (2022)

APÊNDICE E – MODELO FÍSICO DO BANCO DE DADOS

FIGURA 31 - MODELO FÍSICO DE BANCO DE DADOS

-



FONTE: O autor (2022)