DOCUMENTAÇÃO DO PROJETO

GymTech

SUMÁRIO

EQUIPE **3**
OBJETIVO ESTRATÉGICO DO PROJETO **3**
RESUMO DO PROJETO **3**
IMERSÃO **4**
PROBLEMAS QUE O PROJETO VISA RESOLVER 4
PESSOAS QUE O PROJETO VISA AJUDAR 5
BENEFÍCIOS DO PROJETO 6
PRODUTOS SEMELHANTES 6
PERSONAS 6
SOLUÇÕES ENCONTRADAS 7
DIFERENCIAL DO PROJETO 7
BACKLOG DO PROJETO 7
CONSTRUÇÃO **8**
TECNOLOGIA UTILIZADA 8
DIAGRAMA DE PROJETO 8
DIAGRAMA DE CLASSES 9
DIAGRAMA DE CASO DE USO 14

EQUIPE

Nome	Papel
Gabriel Barbosa Lucas	Desenvolvedor, Documentador, Designer de Software
Breno Ricardo Ferreira Antunes	Desenvolvedor Back-end, Designer de Software

OBJETIVO ESTRATÉGICO DO PROJETO

O objetivo estratégico do projeto é definido pela seguinte fras

“Criar uma solução cuja proposta de funcionamento que seja expansível e venha auxiliar no gerenciamento em um futuro cenário de expansão”.

RESUMO DO PROJETO

Minimundo

Zezinho é o proprietário da GymTech, uma academia que combina tecnologia e musculação em um ambiente moderno e convidativo, voltado para pessoas que valorizam a saúde e o bem-estar.

Inicialmente, o negócio de Zezinho começou pequeno, com um baixo fluxo de clientes devido ao pouco reconhecimento que o negócio tinha em seu bairro. Porém, com o tempo seu negócio se popularizou, fazendo com que a quantidade de clientes em sua academia aumentasse significativamente. Apesar da GymTech possuir estrutura para comportar essa crescente quantidade de clientes, ela não tem um meio eficiente para gerenciar esse crescimento na clientela. Para resolver esse problema, Zezinho decidiu investir no desenvolvimento de um sistema de gestão para otimizar o controle dos processos na GymTech e proporcionar mais eficiência no atendimento ao cliente e na assistência ao mesmo em seu acompanhamento de treinos e planos de assinatura.

IMERSÃO

A GymTech enfrenta dificuldades em gerenciar o aumento de clientes e oferecer um atendimento eficiente devido à ausência de um sistema organizado para controle de informações e acompanhamento de treinos e planos.

Os clientes poderão realizar cadastro e login, consultar horários de funcionamento e aulas disponíveis, visualizar e assinar planos de maneira autônoma e acompanhar treinos personalizados, permitindo um engajamento direto com seus objetivos.

Do lado administrativo, o sistema permitirá gerenciar perfis de clientes, criar e organizar planos de assinatura, configurar promoções e registrar treinos personalizados para cada cliente. Além disso, incluirá um painel de métricas com informações importantes, como o número de clientes ativos e renovações pendentes, otimizando a tomada de decisões.

PROBLEMAS QUE O PROJETO VISA RESOLVER

- **1. Gerenciamento Ineficiente de Clientes:**
  Dificuldade em organizar e acessar informações sobre os clientes, como dados cadastrais, planos ativos e histórico de treinos.

- **2. Acompanhamento de Treinos Personalizados:**
  Ausência de um método prático para os instrutores criarem e atualizarem treinos personalizados e para os clientes acompanharem seu progresso.

- **3. Controle Manual de Planos de Assinatura:**
  Falta de uma ferramenta centralizada para gerenciar planos, renovações e promoções, dificultando o controle e aumentando o trabalho manual.

- **4. Dificuldade de Consulta para Clientes:**
  Clientes não conseguem acessar facilmente informações como horários de funcionamento, aulas disponíveis e status de seus planos.

- **5. Ausência de Dados para Decisões Estratégicas:**

  Falta de métricas claras sobre a operação da academia, como número de clientes ativos, renovações pendentes e adesão a promoções, limitando a tomada de decisões informadas.

PESSOAS QUE O PROJETO VISA AJUDAR

1. **Clientes Atuais da GymTech:**

   Pessoas que já frequentam a academia e precisam de uma experiência mais prática para acessar seus planos, acompanhar seus treinos e consultar informações importantes, como horários e promoções.

2. **Novos Clientes Potenciais:**

   Indivíduos interessados em se matricular na GymTech que buscam informações claras sobre os planos disponíveis, promoções e facilidades oferecidas pela academia.

3. **Equipe Administrativa:**

   Fulano e seus funcionários, responsáveis por gerenciar os processos da academia, como cadastro de clientes, criação de planos, personalização de treinos e monitoramento de métricas.

4. **Professores:**

   Profissionais responsáveis pela criação e acompanhamento de treinos personalizados, que precisam de uma ferramenta para organizar e registrar os treinos dos alunos de forma eficiente.

BENEFÍCIOS DO PROJETO

1. Para a administração da GymTech:

- Maior eficiência no gerenciamento de clientes, planos e treinos.
- Redução de tarefas manuais e economia de tempo.
- Automação de notificações e lembretes para facilitar a comunicação com os clientes.
- Acesso a métricas e relatórios detalhados para decisões estratégicas e crescimento sustentável.

2. Para os clientes:

- Experiência mais prática e moderna, com acesso fácil a informações importantes.
- Ferramentas para acompanhar treinos personalizados e gerenciar planos de forma autônoma.
- Atendimento mais ágil e completo, aumentando a satisfação e o engajamento.

3. Para os instrutores/professores:

- Mais eficiência na assistência aos clientes em seu acompanhamento de treino;
- Praticidade no gerenciamento das fichas de treino;

PRODUTOS SEMELHANTES

Em academias de alcance regional baixo(no contexto do mercado de academias no Brasil), é relativamente comum elas terem seus próprios sistemas de gerenciamento semelhantes à solução proposta por este projeto. Um exemplo de academia é a Engenharia do Corpo (que fica no Shopping Montserrat, Serra, ES, Brasil); que utiliza um sistema para gerenciar a entrada e saída de clientes; bem como a geração de fichas de treino para impressão.

PERSONAS

1. Pessoa jovem-adulta, residente local(que more na região), pratica musculação, possui dispositivo móvel ou PC(Personal Computer)

SOLUÇÕES ENCONTRADAS

- **GymPass:** A proposta do aplicativo se apresenta semelhante no que diz respeito a oferecer planos de assinatura para acesso a academias.

DIFERENCIAL DO PROJETO

Capacidade de interagir com professores remotamente e acessar fichas de treino personalizadas via aplicação, sem a necessidade de pedir pela impressão da mesma no atendimento da academia; ou seja, representa um gasto menor por geração de ficha.

BACKLOG DO PROJETO

ID	História do Usuário	MoSCoW	Importância	RoadMap
001	Como Aluno, Instrutor ou Administrador, eu quero poder fazer cadastro e login no sistema	Must Have	Alta	Primeira entrega (19/11) ~ Segunda entrega
002	Como Professor, quero gerenciar os planos de assinatura dos alunos	Must Have	Alta	Primeira entrega (19/11) ~ Segunda entrega
003	Como Professor, quero gerenciar as fichas de treino dos alunos	Must Have	Alta	Primeira entrega (19/11) ~ Segunda entrega
004	Como Cliente, quero gerenciar meu plano de assinatura	Must Have	Alta	Primeira entrega (19/11) ~ Segunda entrega
005	Como Aluno, quero solicitar a mudança da minha ficha de treino	Should Have	Média	Primeira entrega (19/11) ~ Segunda entrega
006	Como Professor, quero gerenciar os cadastros dos alunos	Must Have	Média	Primeira entrega (19/11) ~ Segunda entrega

007	Como Professor, quero gerenciar os equipamentos da academia	Should Have	Baixa	Primeira entrega (19/11) ~ Segunda entrega

CONSTRUÇÃO

TECNOLOGIA UTILIZADA

Java(Ling. de programação) - O Java foi a linguagem de programação escolhida para este projeto (do lado do servidor), por causa dos seguintes motivos:

1. **"Write Once, Run Anywhere" (WORA)**: O código Java é compilado em bytecode, que é executado pela Java Virtual Machine (JVM). Isso significa que o código pode ser executado em qualquer plataforma que tenha a JVM instalada, independentemente do sistema operacional (Windows, Linux, macOS);

2. **Sintaxe de Orientação a Objetos(O.O):** A sintaxe das estruturas, bem como dos conceitos de Orientação a Objetos aplicados no código, são bastante legíveis e de fácil entendimento. Apesar de ser uma linguagem, no geral, verbosa, possui uma leitura e escrita fácil de assimilar quando o assunto é escrever programas orientados a objetos.

Spring + Springboot(framework backend) - O Spring framework (juntamente com o pacote Springboot), foi escolhido pelos seguintes motivos:

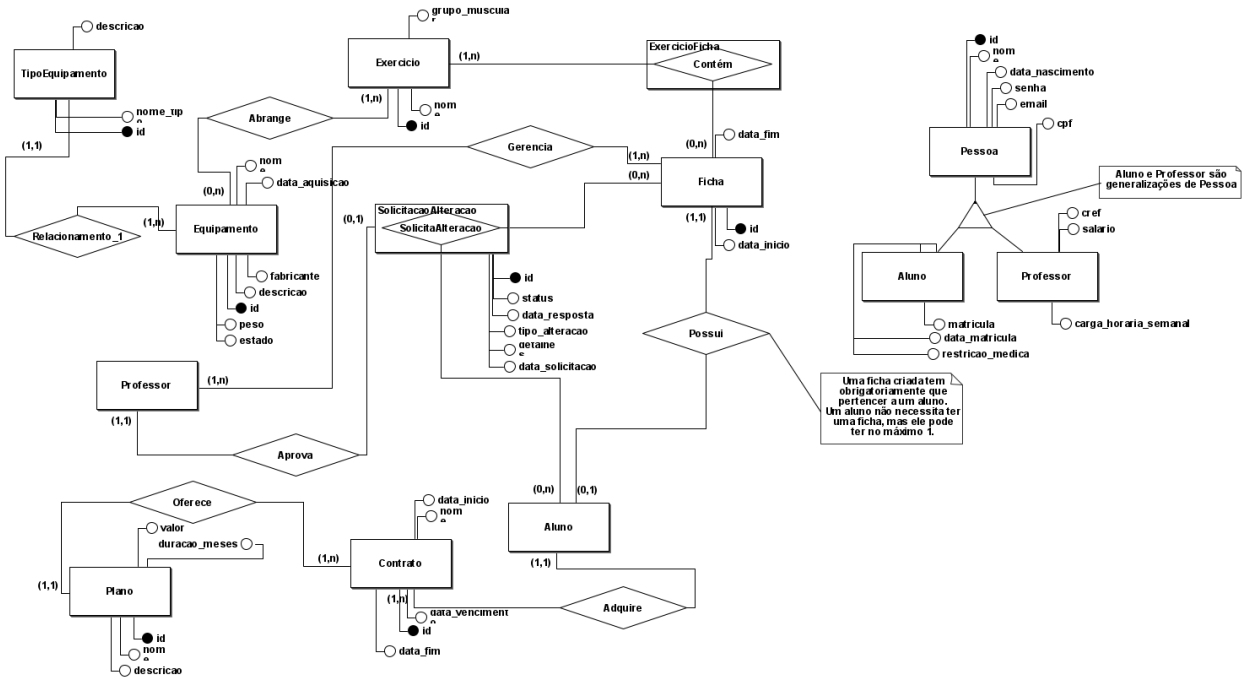
1. **Injeção de Dependência (DI):** O Spring promove uma abordagem de Injeção de Dependência, que facilita o gerenciamento de dependências entre os componentes e melhora a modularidade e testabilidade do código.

2. **Configuração Simplificada:** O Spring permite configurações por meio de anotações ou arquivos XML, e, com o Spring Boot, a configuração automática torna o processo ainda mais simples e rápido.

Framework Front-end: A decidir antes da 2ª entrega do projeto.

DIAGRAMA DE PROJETO

O diagrama abaixo mostra de forma simplificada todas as etapas inerentes ao planejamento e execução deste projeto de software:

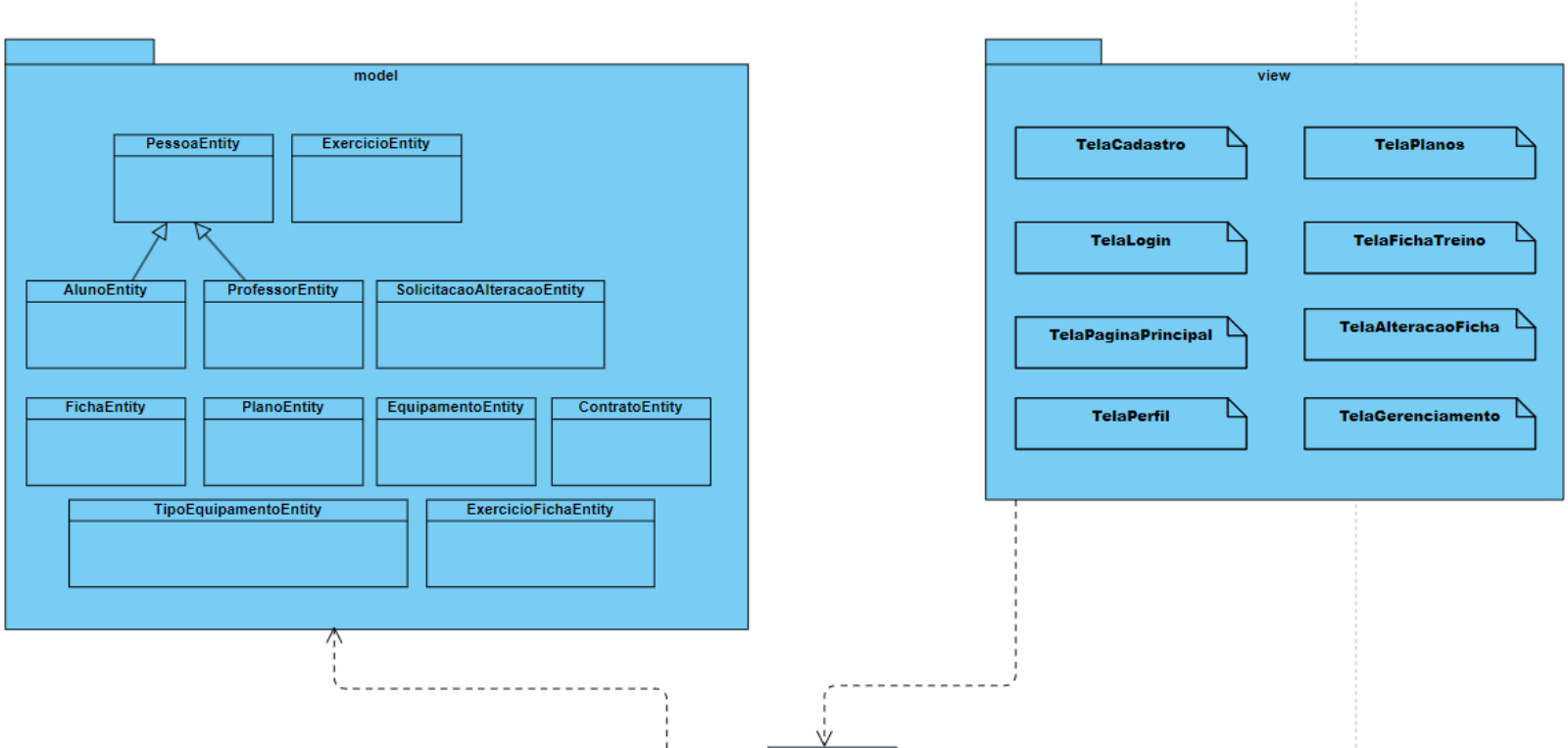


O início do projeto, juntamente com o seu planejamento e análise de requisitos, tem uma projeção de duração de 2 semanas, esta maior parte do tempo sendo usada para definir os principais requisitos funcionais (RFs) não-funcionais (RNFs) e regras de negócio (RN) do projeto. A modelagem e documentação serão feitas de forma simultânea, uma vez que a etapa de documentação não necessariamente depende da modelagem para ser iniciada (durante a modelagem, podem vir a surgir alterações na documentação, como, por exemplo, na mudança de prioridade entre casos de uso registrados no backlog, mudança de tecnologias a serem utilizadas, etc).

Obs.: O commit não é uma etapa de ação única. Serão vários commits realizados por implementação, correção ou atualização de recursos do software.

Obs.: O principal foco para a primeira entrega do projeto na parte do desenvolvimento será o lado do back-end, o front-end será implementado na 2ª versão de entrega do projeto.

DIAGRAMA DE CLASSES



O diagrama acima ilustra um modelo com as principais classes e telas de interface do projeto que são desejadas implementar até o prazo de entrega do mesmo. Pelo diagrama, as telas do pacote “view” dependem das classes do pacote “control”, estas por sua vez dependendo das classes entidade do pacote “model”(consequentemente fazendo do pacote “view” dependente indireto do pacote “model”).

As classes do pacote “model” possuem, cada uma, os seguintes atributos:

1. PessoaEntity
    - **id** (private, Long);
    - **nome** (private, String);.
    - **data_nascimento** (private, LocalDateTime);
    - **cpf** (private, String);
    - **email** (private, String);
    - **senha** (private, String);

2. AlunoEntity
    - ***atributos de PessoaEntity***
    - **matricula_aluno** (private, String);
    - **data_matricula** (private, LocalDateTime);

- **restricao_medica** (private, String);

3. ProfessorEntity
    - **atributos de PessoaEntity;**
    - **salario** (private, double);
    - **cref** (private, String);
    - **carga_horaria_semanal** (private, int)

4. FichaEntity
    - **id** (private, Long);
    - **data_inicio** (private, LocalDateTime);
    - **data_fim** (private, LocalDateTime);
    - **aluno** (private, AlunoEntity);
    - **professor** (private, ProfessorEntity)

5. PlanoEntity
    - **id** (private, Long);
    - **nome** (private, String);
    - .**descricao** (private, String);
    - **valor** (private, double);
    - **duracao_meses** (private, int)

6. EquipamentoEntity
    - **idEquipamento** (private, Long);
    - **nome** (private, String);
    - **peso** (private, double);
    - **fabricante** (private, String);
    - **data_aquisicao** (private, LocalDateTime);
    - **estado** (private, String);
    - **descricao** (private, String);

7. ContratoEntity
    - **id** (private, Long);
    - **aluno**.(private, AlunoEntity);
    - **data_inicio** (private, LocalDateTime);
    - **data_fim** (private, LocalDateTime);
    - **data_vencimento** (private, LocalDateTime);
    - **nome** (private, String);
    - **status** (private, String);
    - **valor_pago** (private, double)

8. ExercicioFichaEntity
    - **id_exercicio_ficha** (private, Long);
    - **exercício** (private, ExercicioEntity);
    - **ficha** (private, FichaEntity);
    - **equipamento** (private, EquipamentoEntity);
    - **numero_repeticao** (private, int);
    - **tempo_descanso** (private, int)
    - **peso** (private, int)

9. TipoEquipamentoEntity
    - **idTipo** (private, Long);
    - **nomeTipo** (private, String);
    - **descricao** (private, String)

10. SolicitacaoAlteracaoEntity
    - **idSolicitacao** (private, Long);
    - **idFicha** (private, Long);
    - **tipoAlteracao** (private, String);
    - **detalhes** (private, String);
    - **dataSolicitacao** (private, LocalDateTime);
    - **status** (private, String);
    - **idAprovador** (private, Long);
    - **dataresposta** (private, LocalDateTime)

Da mesma forma, para as classes do pacote “control”, cada uma terá pelo menos 1 dos seguintes métodos:

- criarX();
- editarX();
- deletarX;
- getX();
- listarX();
- atualizarX().

obs.:
X - Nome correspondente à classe no pacote “model” (exemplo: a classe em “control” correspondente a classe FichaEntity em “model” seria Ficha. Então os métodos teeriam os nomes **criarFicha, editarFicha, deletarFicha, getFicha, listarFicha**). Esta informação também se aplica aos parâmetros dos métodos(onde tem XDTO, no exemplo da ficha, seria **FichaDTO**).

Com relação às telas do pacote “view”, cada uma possui uma função:

TelaCadastro - Realizar um novo registro de usuário (seja aluno ou professor. Somente um professor pode cadastrar outro professor. Um aluno cadastra a si próprio ou um professor o cadastra).

TelaLogin - Fazer login na aplicação.

TelaPaginaPrincipal - Página principal da aplicação. A partir dela serão acessadas as outras telas.

TelaPerfil - Vai mostrar as informações de perfil do usuário.

TelaPlanos - Mostra as opções de planos disponíveis para compra(exclusivo para usuários do tipo Aluno).

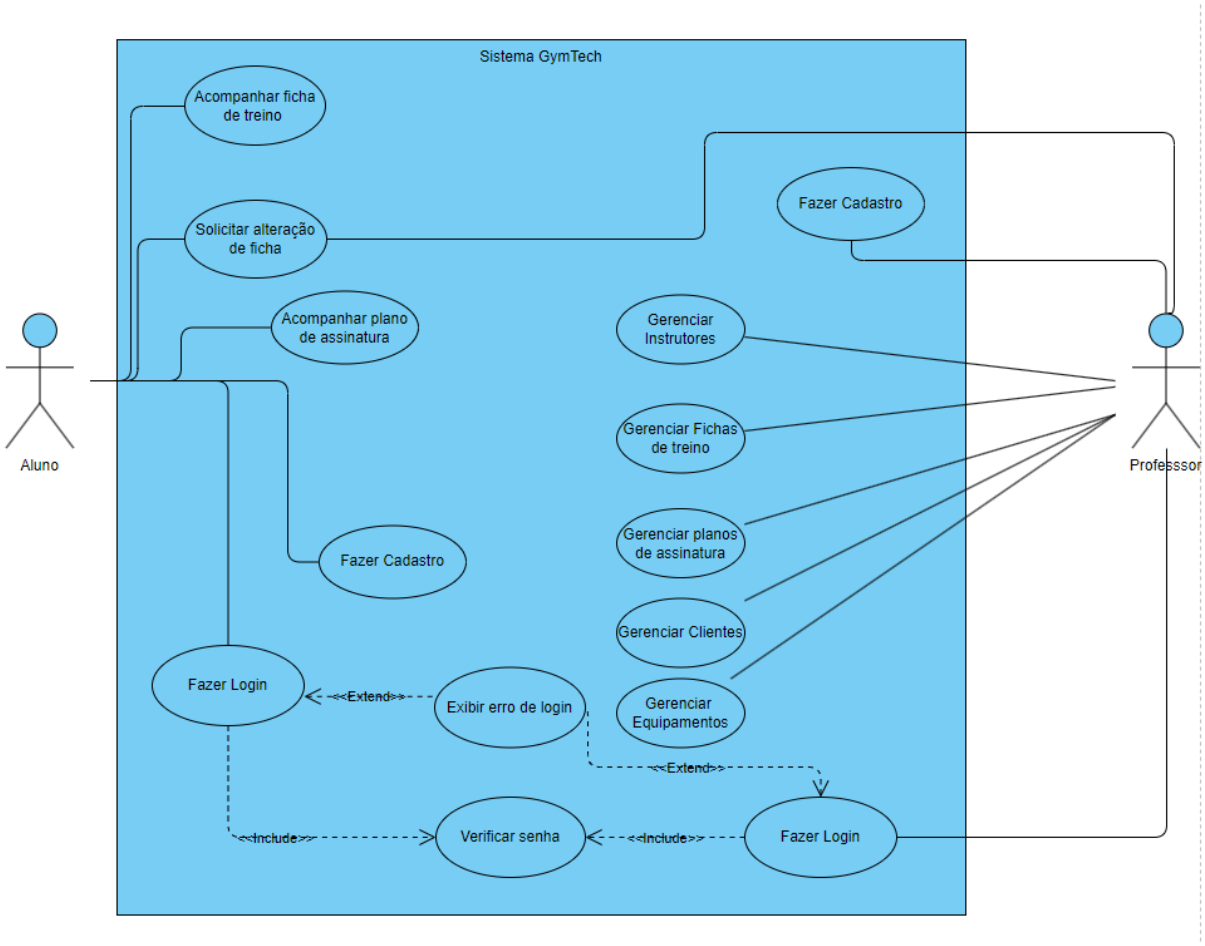
TelaFichaTreino - Layout que mostrará as informações da ficha de treino personalizada do aluno registrado.

TelaAlteracaoFicha - Layout com formulário para abertura de solicitação de alteração de ficha (Esse layout, para um usuário do tipo Professor, aparecerá como uma listagem de solicitações).

TelaGerenciamento - Layout voltado para usuários do tipo Professor. A partir deste layout, o usuário será capaz de gerenciar todos os recursos registrados da academia.

DIAGRAMA DE CASO DE USO

O diagrama de casos de uso abaixo demonstra as interações entre os agentes Aluno e Professor. O Professor, no contexto da GymTech, atua também como uma espécie de administrador da aplicação; podendo não somente gerenciar as fichas, mas também os cadastros de alunos, professores, gerenciar os planos de assinatura e os equipamentos da academia.


Sistema GymTech
Acompanhar ficha de treino
Solicitar alteração de ficha
Acompanhar plano de assinatura
Fazer Cadastro
Fazer Login
Exibir erro de login
Verificar senha
Fazer Login
Gerenciar Instrutores
Gerenciar Fichas de treino
Gerenciar planos de assinatura
Gerenciar Clientes
Gerenciar Equipamentos
Fazer Cadastro
<<Extend>>
<<Include>>
<<Include>>
<<Extend>>
Aluno
Professor