Matheus Malaman & Filippo Dellosso

Academia Tech Fit

Limeira – SP  
2025

Sumário

[**1.** **Introdução** 1](#_Toc215663941)

[**2.** **Diagrama** 2](#_Toc215663942)

[2.1 Diagrama de classe 2](#_Toc215663943)

[**3.** **Wireframe** 3](#_Toc215663944)

[3.1 Feito à Mão 3](#_Toc215663945)

[3.2 Feito pelo Figma 3](#_Toc215663946)

[**4.** **Metodologias Ágeis** 4](#_Toc215663947)

[4.1 Kanban 4](#_Toc215663948)

[**5** **Requisitos funcionais e não funcionais** 5](#_Toc215663949)

[5.1 Requisitos Funcionais 6](#_Toc215663950)

[5.2 Requisitos não funcionais 7](#_Toc215663951)

[**6** **Qualidade do Software** 8](#_Toc215663952)

[6.1 Manutenibilidade 8](#_Toc215663953)

[6.2 Segurança 9](#_Toc215663954)

[6.3 Usabilidade 9](#_Toc215663955)

[6.4 Eficiência e Desempenho 10](#_Toc215663956)

[6.5 Confiabilidade 10](#_Toc215663957)

[**7** **Back-End** 11](#_Toc215663958)

[7.1 Agendamentos 11](#_Toc215663959)

[7.2 Funcionários (Admin) 12](#_Toc215663960)

[7.3 Modalidades 12](#_Toc215663961)

[7.4 Clientes 12](#_Toc215663962)

[7.5 Produtos 13](#_Toc215663963)

[7.6 Franquias 13](#_Toc215663964)

[7.7 Autenticação e Sessões 13](#_Toc215663965)

[**8** **Layout do Site** 14](#_Toc215663966)

[8.1 Página de Login 15](#_Toc215663967)

[8.2 Página de cadastro 15](#_Toc215663968)

[8.3 Página do cliente 16](#_Toc215663969)

[8.4 Página do Admin 16](#_Toc215663970)

[**9** **Figma** 17](#_Toc215663971)

[9.1 Layout do cadastro 18](#_Toc215663972)

[9.2 Layout do cliente 18](#_Toc215663973)

[**10** **Conclusão** 19](#_Toc215663974)

# **Introdução**

A "TechFit," uma rede de academias de ginástica com 15 filiais na cidade, busca

modernizar a gestão de suas operações e melhorar a experiência de seus clientes.

Atualmente, a TechFit utiliza um sistema antigo e ineficiente, baseado em planilhas e

controles manuais, o que gera diversos problemas como:

* O **agendamento** de aulas em grupo é feito por telefone ou presencialmente, causando filas, erros e superlotação em algumas turmas, enquanto outras permanecem com vagas ociosas.
* A **identificação** dos alunos é feita por meio de carteirinhas físicas, o que dificulta o controle de frequência, a segurança do acesso e a geração de relatórios de utilização.
* A **comunicação** entre a academia e os alunos é limitada a e-mails e avisos em murais, o que torna difícil o envio de informações importantes, como alterações de horário, promoções e eventos.
* O **registro** e o **acompanhamento** das avaliações físicas dos alunos são feitos em fichas de papel, o que dificulta o acesso aos dados, a análise da evolução dos alunos e a personalização dos treinos.

A TechFit reconhece que esses problemas estão afetando a satisfação dos clientes,

aumentando os custos operacionais e limitando o crescimento da rede.

Nós alunos, desenvolvemos um sistema de software

completo para a TechFit, capaz de solucionar os problemas mencionados e otimizar a

gestão da academia.

# **Diagrama**

Nesta documentação, são apresentados diversos diagramas com o objetivo de ilustrar e facilitar o entendimento da estrutura e do funcionamento do site. Os diagramas utilizados incluem:

## 2.1 Diagrama de classe

Representa a estrutura estática do site, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

# **Wireframe**

## 3.1 Feito à Mão

O wireframe mostra um app chamado TechFit com telas simples de cadastro, menu principal com planos e cartão do usuário, uma área de avaliação física com gráfico e nutrição, e uma tela de contato para enviar mensagens. É um esboço geral do funcionamento básico do aplicativo.

Lousa branca com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

## 3.2 Feito pelo Figma

A imagem mostra uma tela de cadastro do sistema TECHFIT, com campos para inserir nome, email, senha, CEP, rua, número e bairro. Abaixo dos campos há dois botões: Login e Cadastrar, tudo em um layout simples e cinza sobre fundo transparente.

Uma imagem contendo Diagrama

O conteúdo gerado por IA pode estar incorreto.

# **Metodologias Ágeis**

A imagem mostra um quadro **Kanban** do projeto **Academia Tech Fit** no Trello, dividido em três colunas: **A Fazer**, **Em Andamento** e **Concluído**.

* **A Fazer**, estão listadas tarefas pendentes como: criação do sistema de cadastro de alunos, definição dos planos de treino e integração de métodos de pagamento online etc...
* **Em Andamento**, encontram-se atividades em desenvolvimento, como documentação e layout no figma.
* **Concluído**, estão as etapas finalizadas, como criação do logotipo.

## 4.1 Kanban

Esse modelo ágil demonstra que o projeto da Academia Tech Fit está sendo organizado em etapas claras, permitindo acompanhar o progresso de cada tarefa e garantindo que os entregáveis sejam concluídos de forma eficiente e colaborativa.

Tela de computador com fundo azul

O conteúdo gerado por IA pode estar incorreto.

Link: [Trello](https://trello.com/b/JfClboVk/tchfit)

# **Requisitos funcionais e não funcionais**

Os Requisitos Funcionais (RF) definem o que o sistema deve fazer. Eles descrevem as funcionalidades, serviços ou tarefas que o sistema executa para atender diretamente aos objetivos do usuário e à lógica de negócio. Em suma, são as ações do sistema. Por exemplo, em uma aplicação como o TechFit, a capacidade de "Fazer Login" ou "Realizar o CRUD de Produtos" são requisitos funcionais.

Por outro lado, os Requisitos Não Funcionais (RNF) definem como o sistema deve ser. Eles estabelecem critérios de qualidade e restrições que o sistema precisa satisfazer, abrangendo aspectos como desempenho, segurança, usabilidade e manutenibilidade. Embora não descrevam uma função direta, eles são cruciais para a aceitabilidade e eficácia do sistema. No TechFit, a exigência de que a interface seja "Responsiva em dispositivos móveis" ou que o acesso à área administrativa seja "estritamente restrito a perfis 'admin'" são exemplos de requisitos não funcionais.

## 5.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer, ou seja, as ações, comportamentos e processos que ele deve ser capaz de realizar para atender às necessidades dos usuários.

| **Módulo/Área** | **RF ID** | **Requisito Funcional** |
| --- | --- | --- |
| **Autenticação** | **RF001** | O sistema deve permitir que o usuário realize o **Login** com e-mail e senha. (login.php) |
| **Autenticação** | **RF002** | O sistema deve permitir o **Cadastro** de novos clientes, coletando nome, CPF, CEP, data de nascimento, e-mail, endereço, estado e senha. (cadastro\_cliente.php) |
| **Sessão** | **RF003** | O sistema deve permitir o **Logout** seguro, destruindo a sessão e redirecionando para a página de login. (logout.php) |
| **Controle de Acesso** | **RF004** | O sistema deve **diferenciar perfis** de acesso (admin e cliente) e redirecionar para a área correta após o login. (login.php) |
| **Administração (CRUD)** | **RF005** | O administrador deve ser capaz de realizar **CRUD** (Criar, Ler, Atualizar, Deletar) de **Clientes**, incluindo a alteração do perfil\_acesso. (adm\_clientes.php) |
| **Administração (CRUD)** | **RF006** | O administrador deve ser capaz de realizar **CRUD** de **Produtos** (para a loja). (adm\_produtos.php) |
| **Administração (CRUD)** | **RF007** | O administrador deve ser capaz de realizar **CRUD** de **Franquias**. (adm\_franquias.php) |
| **Administração (CRUD)** | **RF008** | O administrador deve ser capaz de realizar **CRUD** de **Modalidades** (aulas), associando-as a um funcionário. (adm\_modalidades.php) |
| **Administração (CRUD)** | **RF009** | O administrador deve ser capaz de realizar **CRUD** de **Agendamentos**, associando data, funcionário e modalidade. (adm\_agendamentos.php) |
| **Busca e Filtro** | **RF010** | O sistema deve permitir que o administrador **filtre/pesquise** registros nas telas de gestão (Clientes, Produtos, Franquias, Agendamentos). (Scripts JS inline nos adm\_\*.php) |
| **E-commerce** | **RF011** | O sistema deve exibir a **vitrine de produtos** (Loja) com opções de **pesquisa e filtro por tipo/categoria**. (loja.html, loja.js) |
| **E-commerce** | **RF012** | O cliente deve poder **adicionar produtos ao carrinho** via interação na vitrine. (loja.js) |
| **Informativo** | **RF013** | O sistema deve **listar as Franquias** disponíveis e exibir seus detalhes (descrição, localização) em um modal. (franquias.html, Franquias.js) |

## 5.2 Requisitos não funcionais

Os requisitos não funcionais definem as características de qualidade do sistema, como desempenho, segurança, usabilidade, disponibilidade e escalabilidade. Esses requisitos não dizem respeito diretamente às funções do sistema, mas sim à forma como ele deve se comportar durante o uso.

| **Categoria** | **RNF ID** | **Requisito Não Funcional** |
| --- | --- | --- |
| **Segurança** | **RNF001** | **Controle de Acesso Restrito:** A área administrativa deve ser acessível **apenas** para usuários com o perfil 'admin'. (adm\_\*.php verificam $\_SESSION['user\_perfil']) |
| **Segurança** | **RNF002** | **Proteção contra SQL Injection:** Todas as operações de banco de dados devem utilizar consultas parametrizadas (**PDO Prepared Statements**). (\*DAO.php utiliza prepare e execute) |
| **Usabilidade** | **RNF003** | **Responsividade:** A interface, especialmente as páginas públicas (ex: Loja), deve ser adaptável a diferentes tamanhos de tela (desktop, mobile). (loja.css usa @media queries) |
| **Usabilidade** | **RNF004** | **Feedback ao Usuário:** O sistema deve fornecer alertas de sucesso ou erro após operações (CRUD) no painel administrativo. (Função exibirAlerta nos adm\_\*.php) |
| **Estética** | **RNF005** | **Identidade Visual:** O sistema deve manter um tema escuro e moderno, utilizando a fonte **Orbitron** para uma estética tecnológica. (style.css, adm.css e login\_register.css) |
| **Arquitetura** | **RNF006** | **Padrão de Código:** O código deve seguir o **Padrão DAO (Data Access Object)** para isolar a lógica de acesso ao banco de dados da lógica de controle. (\*DAO.php classes) |
| **Arquitetura** | **RNF007** | **Conexão Única:** A conexão com o banco de dados deve ser gerenciada através do padrão **Singleton** para garantir uma única instância de conexão. (Connection.php) |
| **Manutenibilidade** | **RNF008** | **Código Modular:** O código JavaScript deve ser modularizado e separado por funcionalidade (e.g., loja.js para e-commerce, Franquias.js para listagem de unidades). |

# **Qualidade do Software**

O sistema TechFit apresenta uma qualidade sólida, destacando-se positivamente na Segurança (uso correto de Prepared Statements) e na Usabilidade (design moderno, responsivo e bom feedback ao usuário). A Eficiência e a Confiabilidade são boas devido à arquitetura (Padrão DAO e Singleton) e ao processamento de filtros no lado do cliente. A principal fraqueza reside na Manutenibilidade, devido à prática de misturar código de lógica PHP com o código HTML de apresentação nas páginas de administração.

## 6.1 Manutenibilidade

A Manutenibilidade refere-se à facilidade com que o software pode ser modificado, corrigido e evoluído.

| **Aspecto** | **Análise** | **Avaliação** |
| --- | --- | --- |
| **Arquitetura** | O sistema adota o **Padrão DAO (Data Access Object)**, separando claramente as Entidades (Modelos como Cliente.php) da lógica de acesso ao banco de dados (ClienteDAO.php). Esta separação é um forte indicador de **alta manutenibilidade** e modularidade. | **Alta** |
| **Padrão de Conexão** | O uso do padrão **Singleton** na classe Connection.php centraliza a gestão da conexão com o banco de dados. Isso evita a duplicação de código de conexão e facilita alterações futuras nas credenciais do banco. | **Alta** |
| **Separação de Camadas** | As páginas de visão (ex: adm\_clientes.php) misturam excessivamente código PHP (lógica de controle) com HTML e tags de apresentação. Isso compromete a manutenibilidade, pois dificulta a leitura e alteração separada do *design* e do comportamento. | **Média** |

## 6.2 Segurança

A segurança mede a capacidade do sistema de proteger informações e recursos contra acesso não autorizado.

| **Aspecto** | **Análise** | **Avaliação** |
| --- | --- | --- |
| **Proteção contra SQL Injection** | Todos os métodos nas classes DAO (ClienteDAO.php, AgendamentoDAO.php, etc.) utilizam consistentemente **PDO Prepared Statements** (consultas parametrizadas). Esta é a defesa mais crítica contra ataques de injeção de SQL. | **Alta** |
| **Controle de Acesso (Autorização)** | Todas as páginas administrativas (adm\_\*.php) implementam uma verificação de sessão estrita (if ($\_SESSION['user\_perfil'] !== 'admin')). Isso garante que apenas usuários autorizados tenham acesso à gestão do sistema. | **Alta** |
| **Segurança da Senha** | O login.php chama o método login() do Controller, que geralmente lida com a verificação de hash da senha (embora o hashing em si não esteja visível no código, o padrão sugere sua presença, o que é fundamental). O logout.php destrói a sessão de forma completa, garantindo o fim seguro do acesso. | **Alta (Assumindo Hashing)** |

## 6.3 Usabilidade

A Usabilidade refere-se à facilidade de aprendizado e operação do sistema pelo usuário.

| **Aspecto** | **Análise** | **Avaliação** |
| --- | --- | --- |
| **Identidade Visual e Estética** | Os arquivos CSS (adm.css, style.css, etc.) estabelecem um tema escuro consistente, moderno e de alta tecnologia (fonte Orbitron, acentos roxos). Isso cria uma experiência visual forte e coesa. | **Alta** |
| **Adaptabilidade (Responsividade)** | O arquivo loja.css inclui explicitamente *media queries* (@media (max-width: 980px)), indicando que o design foi planejado para se adaptar a dispositivos móveis, um fator chave na usabilidade moderna. | **Alta** |
| **Interação e Feedback** | O toggle-password.js aumenta a usabilidade ao permitir que o usuário veja a senha digitada. Além disso, as páginas administrativas utilizam a função exibirAlerta para fornecer feedback claro (sucesso/erro) após cada operação de CRUD. | **Alta** |

## 6.4 Eficiência e Desempenho

A eficiência mede a capacidade de resposta do sistema sob determinadas condições de carga.

| **Aspecto** | **Análise** | **Avaliação** |
| --- | --- | --- |
| **Conexão com Banco de Dados** | O padrão **Singleton** (RNF007) na classe Connection otimiza o uso de recursos, garantindo que o sistema estabeleça e reutilize uma única conexão com o banco de dados, minimizando o *overhead* de conexão. | **Alta** |
| **Otimização de Consultas** | As classes DAO utilizam comandos JOIN (como em AgendamentoDAO.php) para obter dados relacionados (ex: nome do funcionário e modalidade) em uma única consulta ao invés de múltiplas consultas separadas, o que melhora o tempo de resposta. | **Alta** |
| **Processamento no Cliente** | As tabelas administrativas e a página de Franquias (Franquias.js, adm\_\*.php com scripts inline) implementam filtros e pesquisas via **JavaScript**. Isso delega o trabalho de filtragem para o navegador do cliente, reduzindo a carga no servidor e melhorando a performance percebida. | **Alta** |

## 6.5 Confiabilidade

A Confiabilidade refere-se à capacidade do sistema de operar sem falhas durante um período especificado.

| **Aspecto** | **Análise** | **Avaliação** |
| --- | --- | --- |
| **Tratamento de Erros Críticos** | O Connection.php usa blocos try...catch para lidar com falhas de conexão com o banco de dados, interrompendo o processo com uma mensagem de erro (die). Embora básico, garante que o sistema não continue a operar em um estado inconsistente se a conexão falhar. | **Média** |
| **Consistência de Dados** | A arquitetura DAO, em conjunto com o uso de Prepared Statements, contribui indiretamente para a confiabilidade, garantindo que as operações de leitura e escrita no banco de dados sejam feitas de forma previsível e segura. | **Alta** |
| **Gerenciamento de Sessão** | O uso correto de session\_start() e o encerramento seguro de sessão no logout.php (destruindo variáveis e cookies de sessão) garantem que o estado do usuário seja gerenciado de forma confiável. | **Alta** |

# **Back-End**

Este sistema foi desenvolvido utilizando a arquitetura MVC, onde cada parte do código cumpre uma função específica para manter a organização e facilitar futuras manutenções. As páginas PHP são responsáveis por exibir as informações e enviar as ações solicitadas pelos usuários, enquanto os controllers fazem a intermediação entre o front e o banco de dados. Toda a comunicação direta com o banco é feita através dos arquivos DAO, que executam as consultas e retornam os resultados em forma de objetos de modelo. Dessa forma, cada área do sistema possui um fluxo claro e separado, permitindo que o cadastro, edição, exclusão e listagem dos dados aconteçam de forma estruturada. A documentação a seguir descreve como cada parte do backend funciona, explicando quais arquivos são utilizados em cada setor e como os dados são processados internamente, seguindo o padrão de funcionamento aplicado em todo o projeto.

## 7.1 Agendamentos

Na área de agendamentos puxa pelo AgendamentoController.php, que é o responsável por receber todas as ações enviadas pela interface. Esse controller usa o AgendamentoDAO.php, que executa os comandos SQL no banco de dados. Quando a página adm\_agendamentos.php é carregada, ela instancia o controller e chama o método ler(), e esse método solicita ao DAO a listagem completa dos agendamentos, retornando objetos contendo id, data formatada, funcionário relacionado e modalidade vinculada. Quando o usuário clica em cadastrar, o formulário envia acao=salvar, então o controller chama criar(), que repassa as informações ao AgendamentoDAO.php, que insere o registro no banco. Quando o usuário clica em editar, o formulário envia acao=editar, e o controller usa buscarPorId(), que solicita ao DAO que retorne o agendamento específico pelo ID. Ao salvar a edição, acao=atualizar é enviada, o controller chama atualizar() e o DAO executa o update no banco. Quando o usuário clica em excluir, acao=deletar é enviada, o controller chama deletar(), que repassa ao DAO para remover o registro. Além disso, o controller ainda puxa a lista de funcionários através de listarFuncionariosAdmin(), que normalmente vem do FuncionarioDAO.php, e também puxa a lista de modalidades através de listarModalidades(), que utiliza o ModalidadeDAO.php, permitindo que a página exiba as opções corretas nos selects.

## 7.2 Funcionários (Admin)

Na área de funcionários puxa pelo FuncionarioController.php, que recebe as ações de cadastro, edição, atualização e exclusão. Esse controller utiliza o FuncionarioDAO.php, que acessa diretamente o banco. Quando a página de administração de funcionários é carregada, ela executa o método ler(), que retorna todos os funcionários cadastrados. Quando o usuário envia os dados de cadastro, o controller chama criar(), e o FuncionarioDAO.php executa o insert no banco. Ao clicar em editar, o controller chama buscarPorId(), e o DAO retorna o funcionário correspondente. Ao salvar a edição, o controller chama atualizar(), e o DAO modifica o registro no banco. Ao excluir, o controller chama deletar(), que aciona o DAO para remover o funcionário. Esse controller também costuma ser usado como referência na área de agendamentos, já que retorna a lista de possíveis responsáveis por cada atividade.

## 7.3 Modalidades

Na área de modalidades puxa pelo ModalidadeController.php, que trabalha diretamente com o ModalidadeDAO.php. Esse DAO é responsável pelas consultas SQL de inserir, editar, listar e excluir modalidades. Quando a página administrativa de modalidades é carregada, ela solicita ao controller o método ler(), que aciona o DAO para buscar todas as modalidades. Quando o usuário cadastra uma modalidade nova, envia acao=salvar, o controller chama criar(), e o DAO grava o registro no banco. Ao clicar em editar, acao=editar envia o ID, o controller chama buscarPorId(), e o DAO devolve essa modalidade específica. Ao atualizar, acao=atualizar chama atualizar(), que repassa ao DAO a tarefa de modificar o registro. Ao excluir, acao=deletar aciona o controller, que chama deletar(), e o DAO remove a modalidade. Essa categoria também é usada pelo módulo de agendamentos, pois é o DAO de modalidades que fornece a lista para preencher o select da página.

## 7.4 Clientes

Na área de clientes puxa pelo ClienteController.php, que usa o ClienteDAO.php para manipular o banco de dados. Quando a página de clientes é carregada, o controller chama ler() e o DAO retorna todos os clientes registrados. Quando o usuário cadastra um cliente, o controller chama criar(), e o DAO insere os dados no banco. Ao editar um cliente, acao=editar aciona buscarPorId(), que consulta o DAO. Ao salvar a edição, acao=atualizar usa atualizar(), que chama o DAO para modificar os dados. Quando o usuário clica em excluir, acao=deletar chama o método deletar(), e o DAO remove o cliente. Esse módulo geralmente é usado apenas internamente, mas pode se conectar ao sistema de login caso exista vínculo com autenticação.

## 7.5 Produtos

Na área de produtos puxa pelo ProdutoController.php, que depende do ProdutoDAO.php para persistência no banco. Quando a página é aberta, o controller chama ler(), e o DAO retorna todos os produtos cadastrados. Durante o cadastro, acao=salvar chama criar(), e o DAO executa o insert. No momento da edição, acao=editar chama buscarPorId(), que retorna o produto específico com base no ID. Após editar, acao=atualizar chama atualizar(), e o DAO modifica os dados no banco. Ao excluir, acao=deletar chama deletar(), que solicita ao DAO para remover o produto. Em alguns sistemas, esse módulo pode integrar com estoque ou vendas caso existam.

## 7.6 Franquias

Na área de franquias puxa pelo FranquiaController.php, que acessa o FranquiaDAO.php para qualquer alteração no banco. Ao carregar a página, o método ler() é chamado e o DAO retorna todas as franquias. Durante o cadastro, o controller chama criar(), que envia ao DAO para inserir a nova franquia. Ao editar, buscarPorId() usa o DAO para trazer os dados. Atualizar chama o DAO para salvar a modificação. Deletar remove a franquia pelo DAO. Esse módulo pode servir como base para definir locais de atendimento no sistema.

## 7.7 Autenticação e Sessões

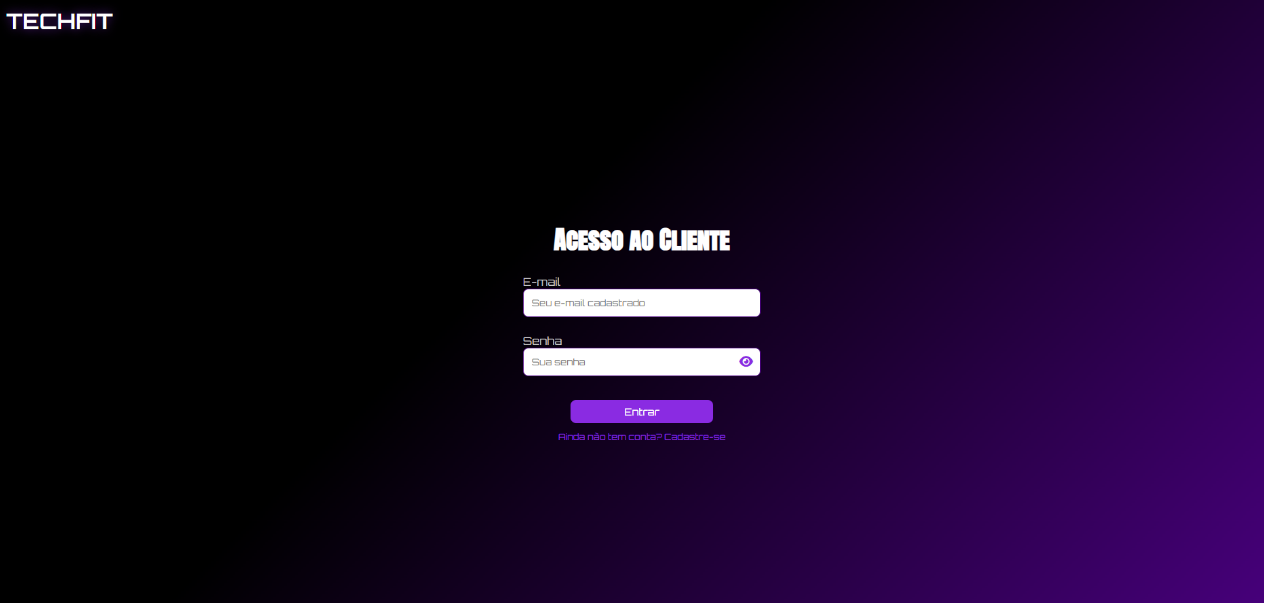
Na área de login puxa pelo LoginController.php, que geralmente utiliza UsuarioDAO.php ou FuncionarioDAO.php, dependendo de como o sistema foi estruturado. Quando o usuário envia os dados de login, o controller chama validarLogin(), que solicita ao DAO que busque o usuário no banco pelo email e senha. Se os dados estiverem corretos, o controller cria a sessão com o perfil do usuário. A verificação de permissão ocorre diretamente nas páginas, como em adm\_agendamentos.php, que checa se a sessão existe e se o perfil do usuário é admin. Caso contrário, redireciona para login.php. O logout destrói a sessão e retorna para o login.

# **Layout do Site**

Os layouts apresentados compõem a identidade visual completa do ecossistema Tech Fit, unindo telas de acesso, cadastro, área institucional e painel administrativo. Cada página segue uma estética moderna com predominância de preto e tons de roxo, transmitindo tecnologia, performance e profissionalismo. Desde as interfaces de login e cadastro do cliente, passando pela página inicial com forte apelo motivacional, até o painel administrativo voltado à gestão interna, todos os elementos visuais mantêm consistência no estilo, tipografia e usabilidade. Juntos, esses layouts formam uma experiência sólida, organizada e voltada tanto para o usuário final quanto para os administradores da plataforma.

## 8.1 Página de Login

A imagem mostra uma tela de login com fundo em degradê preto para roxo, o logo “TECHFIT” no canto superior esquerdo e, ao centro, o título “Acesso ao Cliente” com campos de e-mail e senha, um botão roxo de “Entrar” e um link abaixo para cadastro.



## 8.2 Página de cadastro

A imagem mostra uma tela de cadastro com fundo em degradê do preto ao roxo. No topo esquerdo aparece o logo “TECHFIT”. No centro, o título “Cadastro” está em branco e destacado. Abaixo dele há um formulário dividido em duas colunas com campos como nome, e-mail, CPF, senha, data de nascimento, CEP, endereço e estado. Também há uma caixa para aceitar os termos de uso e, ao final, um botão roxo escrito “Cadastrar”, seguido por um link de “Login”.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

## 8.3 Página do cliente

A imagem mostra a página inicial da TECHFIT com um menu escuro no topo exibindo opções como Franquias, Planos, Modalidades e Loja, além de um ícone de perfil. O fundo apresenta uma academia moderna iluminada por luzes roxas, com esteiras alinhadas. À direita, uma mulher segurando uma corda de treino sorri enquanto olha para o lado, usando um top com a marca TECHFIT. À esquerda aparece a frase em destaque “TRANSFORME SUOR EM RESULTADO”.

Tela de celular com imagem de uma pessoa

O conteúdo gerado por IA pode estar incorreto.

## 8.4 Página do Admin

A imagem mostra o painel administrativo da Tech Fit com menu no topo, o usuário Matheus logado como admin e um formulário para cadastrar novos clientes ou funcionários, com campos de dados pessoais e um botão roxo de cadastro. Abaixo há uma seção para visualizar e buscar funcionários, tudo em um layout preto e roxo.

Tela de computador com texto preto sobre fundo escuro

O conteúdo gerado por IA pode estar incorreto.

# **Figma**

A TECHFIT apresenta uma identidade visual moderna e voltada para o ambiente de academia, usando tons escuros contrastados com elementos claros para destacar as informações importantes. Os layouts seguem uma linha tecnológica e objetiva, facilitando a navegação do usuário tanto no processo de cadastro quanto na área principal do site. As telas mostram uma combinação de praticidade e estética, com campos bem organizados, imagens motivacionais e um visual que reforça o conceito de desempenho, energia e transformação física. Essa apresentação visual cria uma experiência clara e intuitiva, acompanhando a proposta da plataforma de oferecer ao usuário um acesso simples, direto e eficiente aos serviços da TECHFIT.

## 9.1 Layout do cadastro

A imagem mostra uma tela de cadastro da TECHFIT com fundo escuro em tons de roxo. No centro aparecem campos brancos arredondados para preencher nome, e-mail, CPF, endereço e data de nascimento, e abaixo há um botão também branco escrito “Cadastrar”. O design é simples, contrastante e com estilo tecnológico.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

## 9.2 Layout do cliente

A imagem mostra a página inicial da TECHFIT com menu superior e, ao fundo, uma mulher treinando na academia. Em destaque, aparece a frase motivacional “TRANSFORME SUOR EM RESULTADO”.

Interface gráfica do usuário, Site

O conteúdo gerado por IA pode estar incorreto.

Link: [Figma](https://www.figma.com/design/Apu8LvuTfGSlPOcW8Wf2bM/Tech-Fit?node-id=0-1&p=f&t=waN47B8iKavq5jK0-0)

# **Conclusão**

A plataforma Tech Fit, analisada como um todo, apresenta um ecossistema digital bem estruturado, onde design, funcionalidade e lógica interna trabalham em harmonia para entregar uma experiência profissional tanto ao usuário final quanto à equipe administrativa. Os layouts demonstram uma identidade visual moderna, com forte presença de preto e roxo, reforçando a estética tecnológica da marca. As telas de login, cadastro, área inicial motivacional e painel administrativo refletem uma preocupação com clareza, organização dos elementos e facilidade de navegação, contribuindo para uma jornada intuitiva do usuário.

Por trás dessa interface, o back-end cumpre um papel essencial, garantindo que cada ação executada dentro do sistema aconteça de maneira segura, eficiente e consistente. É ele que valida informações, processa cadastros, controla permissões, gerencia sessões e realiza toda a comunicação com o banco de dados. Assim, tanto o cliente quanto o administrador podem interagir com o sistema com confiança, sabendo que seus dados são tratados com regras bem definidas e fluxos automatizados.

Dessa forma, o conjunto final revela uma aplicação sólida, bem planejada e alinhada com a proposta de modernidade da Tech Fit. Cada página, cada formulário e cada rotina interna contribui para formar um ambiente digital coeso, confiável e preparado para crescer e receber novas funcionalidades no futuro. A integração entre visual, usabilidade e lógica de funcionamento evidencia um projeto pensado para entregar resultados, manter organização operacional e oferecer uma experiência de alta qualidade aos seus usuários.