

ESCOLA SESI SENAI  
CENTRO EDUCACIONAL 125  
ENSINO MÉDIO INTEGRADO AO TÉCNICO

Diego Gomes Alves  
Geovana Clemente Cruz  
Livia Mayumi Hayashida  
Victor do Vale Souza

**AEZ:** While Play

SALTO  
2025

Diego Gomes Alves  
Geovana Clemente Cruz  
Livia Mayumi Hayashida  
Victor do Vale Souza

**AEZ:** While Play

Trabalho apresentado à disciplina Projetos  
do Ensino médio e técnico da Escola  
SESI/SENAI de Salto, como avaliação  
parcial do 2º semestre/etapa do 2º ano.  
Professor(a): Celso Rodrigo Giusti, Daniel  
Manoel Filho e Marlon Fanger Rodrigues.

SALTO  
2025

## SUMÁRIO

1 INTRODUÇÃO .....	3
2 DESENVOLVIMENTO .....	4
3 LEVANTAMENTO DE REQUISITOS .....	5
3.1 Requisitos funcionais: .....	5
3.2 Requisitos não funcionais:.....	6
4 MATRIZ DE RASTREABILIDADE .....	7
5 DIAGRAMA DE CASO DE USO .....	8
7 PROTOTIPAÇÃO(FIGMA) .....	12
8 BANCO DE DADOS .....	23
8.1 Modelo conceitual .....	23
8.2 Modelo lógico .....	23
9 ANÁLISE DE CUSTOS .....	30
10 ANÁLISE DE RISCOS.....	33
11 ORGANIZAÇÃO COM TRELLO.....	37
12 REPOSITÓRIO GIT/GITHUB .....	47
13 EXPLICAÇÃO DO CÓDIGO.....	48
13.1 Frontend .....	48
13.2 Backend .....	52
13.3 Integração com APIs .....	100
14 TELAS EXECUTADAS E FUNCIONALIDADES .....	101
15 CONCLUSÃO.....	105
REFERÊNCIAS .....	109

## **1 INTRODUÇÃO**

A partir da solicitação dos professores da criação de um site que atingisse o interesse de todos os membros do grupo, nós da equipe While Play, trabalhamos para o desenvolvimento de um site chamado AEZ que visa o reconhecimento e a valorização de produtores de filmes, que não possuem muita visibilidade e oportunidade no mercado de trabalho. Para realizarmos esse projeto utilizamos do nosso conhecimento e da aprendizagem em sala de aula, seguindo assim o passo a passo para a criação de um programa bem desenvolvido, organizado e que atendesse todos os problemas que o usuário possa vir a ter no nosso site.

## 2 DESENVOLVIMENTO

O nosso site AEZ como dito anteriormente, tem o intuito de trazer mais atenção das pessoas para produtores de filmes que não possuem muita notabilidade nesse meio. Portanto nossa equipe teve a ideia de criar um site onde as pessoas poderão publicar suas ideias de roteiros em geral e personagens para empresas ou outras pessoas que estão começando agora na área comprarem a ideia e utilizarem. Logo nosso site poderá subir roteiros exclusivos e o autor que denominara seu preço, terá também uma parte de avaliação, além de uma aba com alguns dos roteiros mais famosos e seus prêmios obtidos , o pagamento do usuário será mensal para deixar o projeto armazenado em nosso programa, terá perfil profissional, também terá as páginas onde o usuário poderá analisar os roteiros lançados e os mais avaliado no site, entretanto para comprar , o cliente deve ter a assinatura citada anteriormente, será permitido no programa que as pessoas peguem ideias já existentes e adaptem como spin-off de filmes ou novos derivados, porém não poderão vender. Nossa aplicativo irá conter plano Free e plano Pago, no Free será possível o usuário ler e visualizar os projetos, porém ele não poderá comprar ideias, já no plano Pago a mensalidade será de 9,90 e o usuário poderá subir seus projetos e deixarem ser armazenados, além de poder comprar ideias.

### 2.1 Ferramentas utilizadas

- Banco de dados (MySQL)
- Prototipação (Figma)
- Diagrama de caso de uso (Miro)
- Modelo conceitual e lógico (BrModelo)

### **3 LEVANTAMENTO DE REQUISITOS**

A primeira etapa que concluímos no nosso projeto e que é de extrema importância para ele, é o levantamento de requisitos, para que assim pudéssemos iniciar o projeto de forma mais direcionada e organizada para o que o sistema deve realizar para atender a todos os objetivos que planejamos no nosso site. Dessa forma, o levantamento de requisitos é como se fosse o escopo do nosso programa, pois define as necessidades que o cliente espera ser solucionado pelo software, ele atribui dessa forma as ações que o sistema e os usuários (também chamados de atores) irão desempenhar e que estão interligadas uma com as outras para o seu funcionamento.

- **Atores:**

- Usuários (compradores, visualizadores e vendedores).
- Sistema

- **Ações:**

- criação de seu projeto, além de editar e excluir.
- poderá editar e criar seu perfil.
- login/cadastro e recuperação de senha.
- navegar entre as páginas.
- enviar feedback ao site ou á perfis.
- Poderá comprar projetos e visualizar eles em sua biblioteca.
- Visualizar roteiros e personagens disponíveis criados por outros usuários.
- aceitar termos e condições.

#### **3.1 Requisitos funcionais:**

Requisitos Funcionais definem o que um sistema irá fazer, descrevendo suas determinadas funcionalidades, ações e comportamentos. Eles especificam como o sistema deve responder a entradas, processar dados e interagir com usuários ou outros sistemas.

- **RF:**

- O usuário pode criar seu projeto, dando um título e uma sinopse dele. Além de poder subir um arquivo de uma foto do projeto, podendo editar e exclui-lo também.
- O usuário após fazer um login poderá editar ou criar seu perfil, como sua foto de perfil e informações de sua escolha.
- Usuário deverá fazer se cadastrar para acessar determinadas páginas e funções, assim podendo fazer o login e caso esqueça sua senha á recupere.
- Com o login feito, poderá dar feedbacks ao site caso encontre um erro ou queira fazer uma dica de melhoria, ou dar feedbacks a projetos de outros usuários.
- Após visualizar, se for do interesse do visualizador poderá comprar o projeto, onde poderá acessar sua biblioteca e ver seus projetos comprados, podendo revendê-los também.

- Certas páginas serão para visualizar os projetos lançados por outros usuários, onde todos poderão ver.
- Para entrar no site e comprar projetos, deverá aceitar os termos do site.

### **3.2 Requisitos não funcionais:**

Requisitos não funcionais especificam os critérios de qualidade, restrições e atribuições do sistema que afetam sua operação, desempenho e usabilidade, sem definir funcionalidades diretas. Além disso, estabelece padrões para confiabilidade, eficiência, segurança e manutenibilidade, garantindo que o sistema atenda a expectativas do usuário e técnica.

- **RNF:**

- O sistema deve ser fácil de usar, sendo bem interativo e deve mostrar todas as funções de forma interativa e fácil.
- Sistema deve ser seguro, não tendo vazamento de dados, além do imput de cartões falsos ou golpes.
- O site deve ser leve em questão visual, visto que a subida de roteiros e personagens deve deixá-lo um pouco mais pesado por natureza.
- Sistema deve funcionar 24 horas por dia.
- Sistema deve ser escalável, suportando um mínimo de usuários usando o site ao mesmo tempo.
- Interações com outros sistemas deverá ser fácil, como por exemplo a subida de arquivo de imagem.

#### 4 MATRIZ DE RASTREABILIDADE

A matriz de rastreabilidade de requisitos é uma tabela que ajuda a entender os requisitos, ditos anteriormente, que estão relacionados ao longo do projeto, sendo útil em gerenciamento, principalmente no desenvolvimento de softwares e sistemas, onde garanti que todos os requisitos no desenvolvimento de softwares e sistemas, onde garanti que todos os requisitos sejam atendidos e qualquer outra alteração também.

#### **Matriz de requisitos funcionais:**

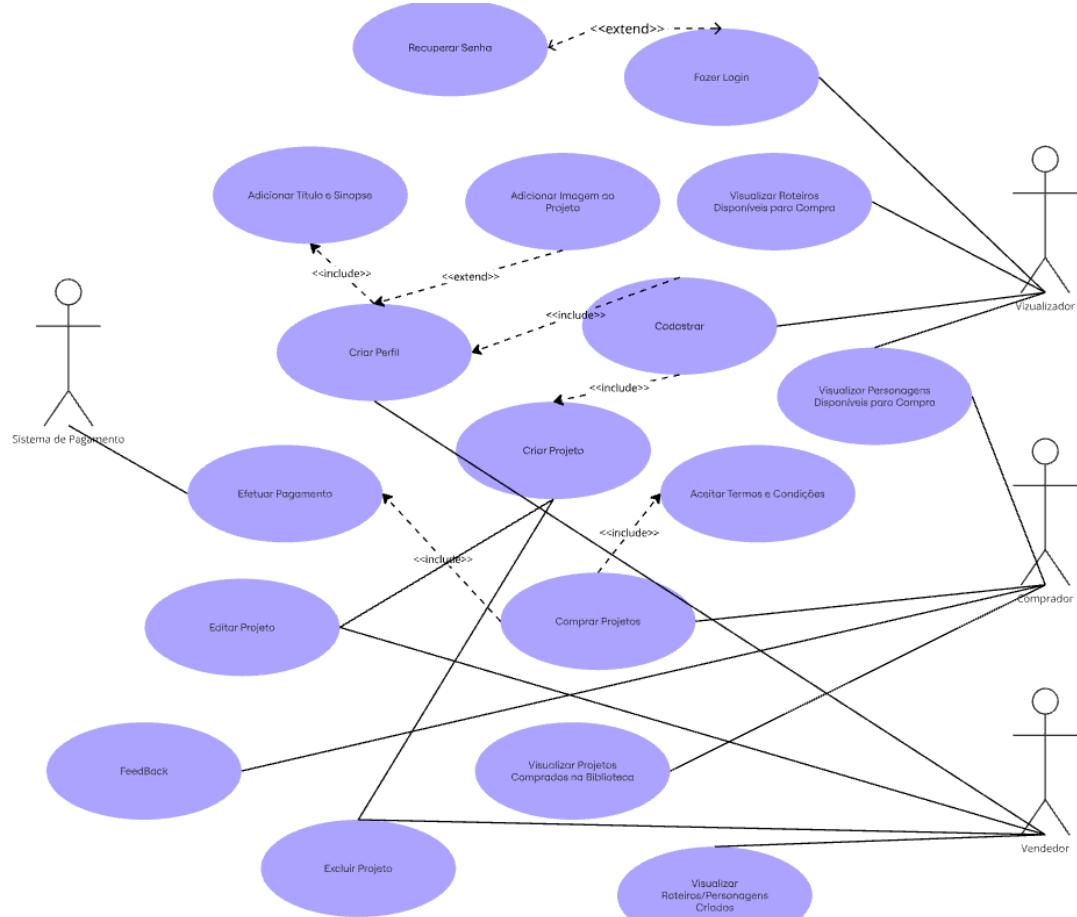
REQUISITOS FUNCIONAIS	RF1	RF2	RF3	RF4	RF5	RF6	RF7
Criar projeto, editar e excluir	X						
Perfil		X					
Login e Cadastro			X				
Feedbacks				X			
Compra e Biblioteca					X		
Visualizar roteiros/personagens						X	
Termos e condições							X

#### **Matriz de requisitos não funcionais:**

REQUISITOS NÃO FUNCIONAIS	RNF1	RNF2	RNF3	RNF4	RNF5	RNF6
Facilidade	X					
Segurança		X				
Leve visualmente			X			
Funcionalidade				X		
Escalabilidade					X	
Outros sistemas						X

## 5 DIAGRAMA DE CASO DE USO

Logo após o levantamento de requisitos, nós fizemos o diagrama de caso de uso que basicamente é uma representação visual dos requisitos estabelecidos anteriormente, que nos permite analisarmos melhor como vai ser essa relação de usuário e sistema no site (os atores). Veja a seguir o diagrama de caso de uso do nosso grupo:



## 6 IHC – UX e UI

### UX

- **Praticidade:**

- Nosso site busca a melhor facilidade visual a todos os usuários, seja na navegação em nosso site, ou em entender suas funcionalidades. Nós garantimos essa facilidade pela explicação em nossa homepage de forma bem direta, onde nosso site funciona como uma forma de apoio à futuros artistas.

Nós buscamos isso de forma que não dificulte a pessoa buscar de ir atrás de seu sonho, pois queremos acabar com a dificuldade de nosso artista ser reconhecido pelas pessoas, ou de possíveis golpes e promessa inacabados. Com nosso site, queremos trazer segurança, esperança e praticidade aos futuros astros do cinema.

- **Navegação:**

- Nosso site é simples e direto em nosso objetivo, que é fazer com que pessoas sejam reconhecidas diferente de como ocorre atualmente. Nosso site busca isso, onde a headbar traz basicamente tudo direcionando nosso usuário a andar pelo site livremente. Logo quando ele quiser trabalhar com nosso projeto, ele deve fazer a assinatura que já traz o que ele poderá fazer ao assinar. Nosso site é básico, sem muitas páginas e objetivo.

- **Acessibilidade:**

- Nosso site infelizmente não é muito acessível a pessoas com deficiência visual, já que nosso usuário deve principalmente visualizar os roteiros e personagens caso queira comprar ou vender. Porém queremos adicionar um dark/light mode, o que pode acabar ajudando e futuramente configurações para daltonismo.

Já para deficiências auditivas ou diversas outras, nosso site trabalha bem, já que não busca dificuldade alguma em navegar ou trabalharem nosso site.

- **Fluxo do usuário:**

- Nossa primeira página será a homepage, onde o usuário deve fazer login ou se cadastrar, assim ele irá se deparar com as páginas de roteiro (de outras pessoas), páginas de prêmios e personagens. Assim para acessar outras páginas, será obrigatório fazer o login. Logo quando ele efetuar, ele conseguirá atualizar e criar seu perfil. Com isso aparecerá a opção de assinatura, o que levará o usuário ao foco principal do site.

Quando ele pagar, irá se deparar com novas funções e que a partir desse momento, ele poderá oficialmente ganhar dinheiro com esse site, publicando seus projetos ou comprando projetos alheios.

### Ui:

- **Cores e visual:**

- Seguimos as cores e as fontes de acordo com nossa prototipação, seguindo um tema mais “black and white”, porém com ressalvas. Buscamos cores neutras já que para leituras e visualizações de texto, algo mais simples deve ser mais simples e fáceis de ver.

Porém nosso projeto tem ressalvas nesse tema de cores, pois em fotos de personagens criados e a foto própria de perfil são o usuário que escolhe e não nós.

- **Layout:**

- Como nosso site há bastante elementos textuais, deixamos como foco as hierarquias visuais em nosso site. Para que as pessoas possam identificar os textos, títulos visíveis e identificáveis, textos fáceis e direto no seu objetivo. Também busquei animações em botões para identificação, mudanças de cursor e entre outras formas de identificação para ajudar o usuário.

- **Consistência:**

- Os botões servem um padrão agradável e fácil de identificar, além de seguirem um padrão visual para não confundir nosso comprador. Todos os sites seguem o mesmo padrão de elementos, em relação a rodapés, headbar e fontes, buscando um agradado visual a todos.

- **Responsividade:**

- Por conta do tempo que tivemos no trabalho, nem todas as páginas estão com responsividade ou alguns elementos podem estar em falta. Porém as páginas que eu (Diego) fiz, em sua maioria estão todas responsivas e apenas a navbar não está completamente responsiva.

## **Melhorias para nosso ui/ux:**

Poderíamos adicionar várias mecânicas, como á já citada “dark mode”, o que facilitaria para pessoas com olhos sensíveis. Também queria abrandar nossas áreas de abrangência, como expandir para composições musicais, ideias para projetos e entre outros. Sobre elementos, eu gostaria de poder trocar as fontes posteriormente dar uma cara só do site, buscar criar elementos que façam lembrar diretamente do seu site e qualquer outro tipo de alteração para expandir a marca.

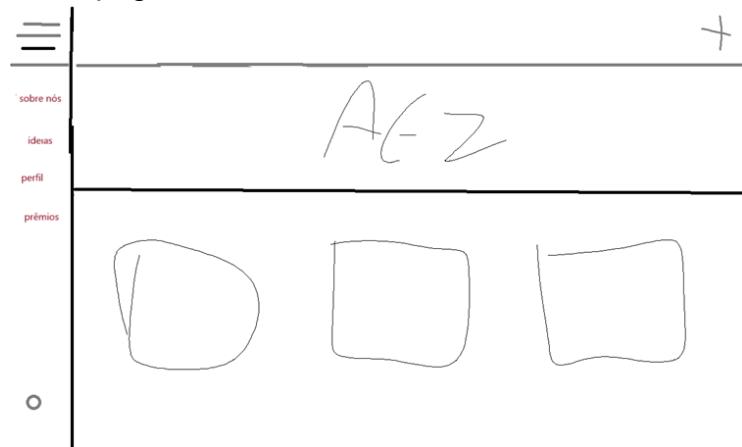
Já de feedbacks que recebemos, gostaram muito da ideia e disseram que é muito nova e muito boa, pois um dos problemas do mundo do cinema é a ignorância em pessoas pequenas, a falta de oportunidades de visibilidade e a dificuldade de emplacar uma carreira, e esse site ajuda muito diretamente nisso.

## 7 PROTOTIPAÇÃO(FIGMA)

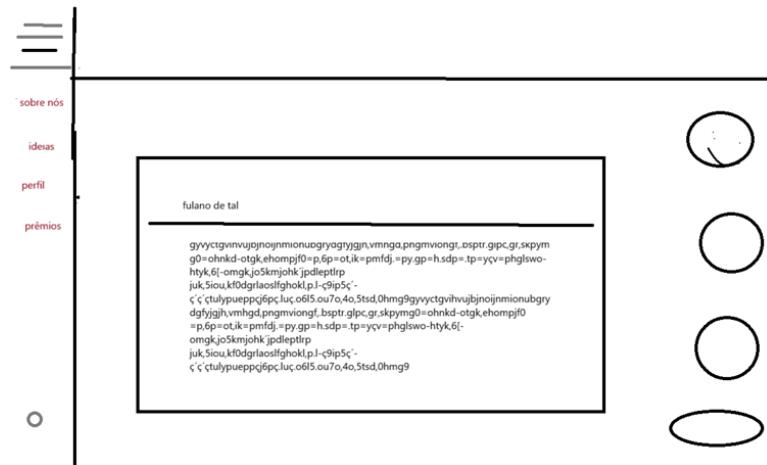
- **Baixa fidelidade:**

Primeiramente antes de fazermos o nosso design de alta fidelidade que basicamente já é o modelo pronto de como queríamos que o nosso site realmente fosse. Nós produzimos alguns rascunhos no papel com desenhos de como imaginávamos o nosso site, para depois já irmos produzir o protótipo de alta fidelidade, preparados com a ideia certa do que desejávamos fazer. Veja a seguir nossos rascunhos:

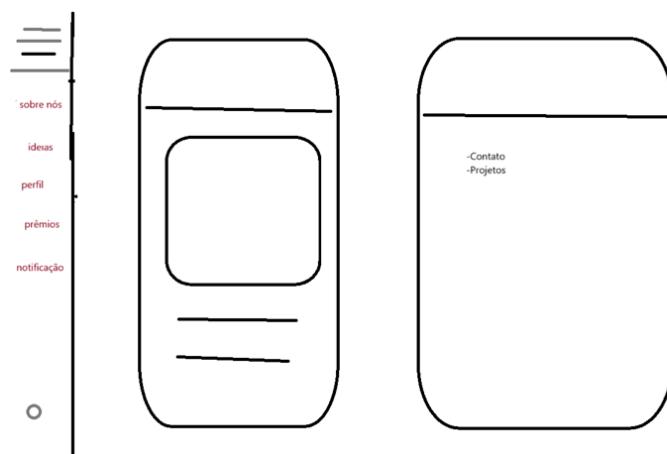
Homepage 1:



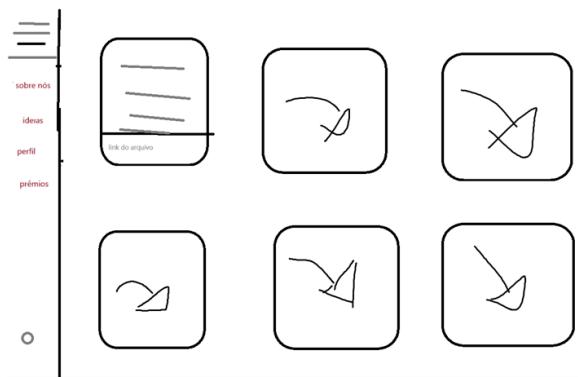
Sobre nós:



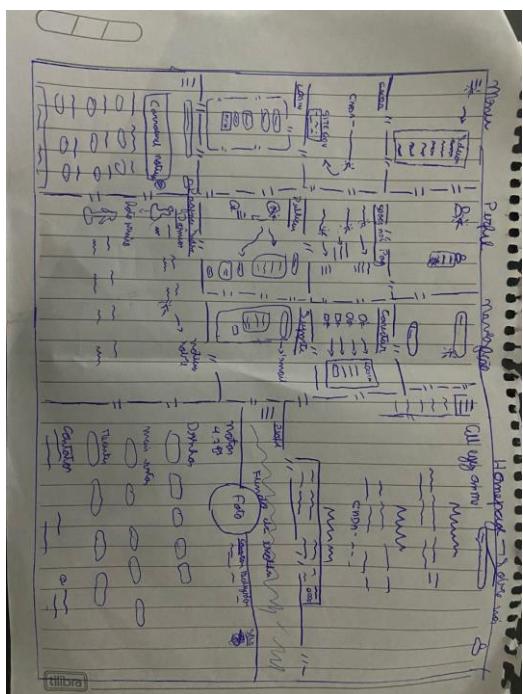
Perfil:

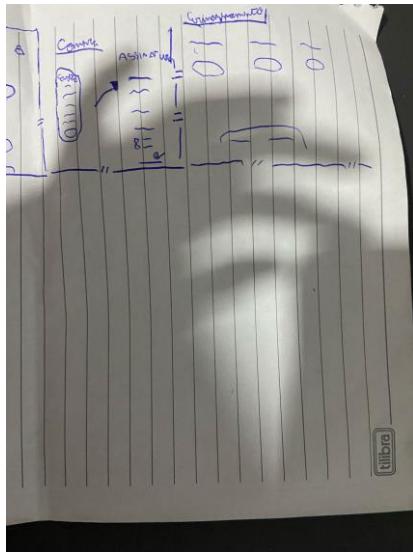


### Biblioteca:



### Rascunhos geral:

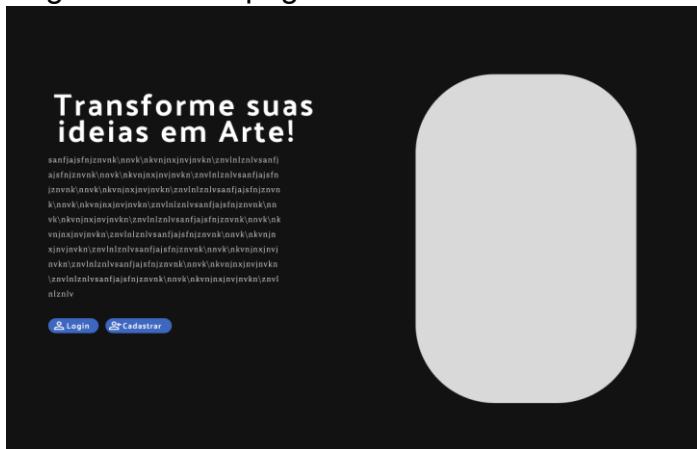




- **Média fidelidade:**

Já a média fidelidade é um modelo mais definido que o rascunho, porém sem interatividade total. Veja a seguir nossas páginas de média fidelidade:

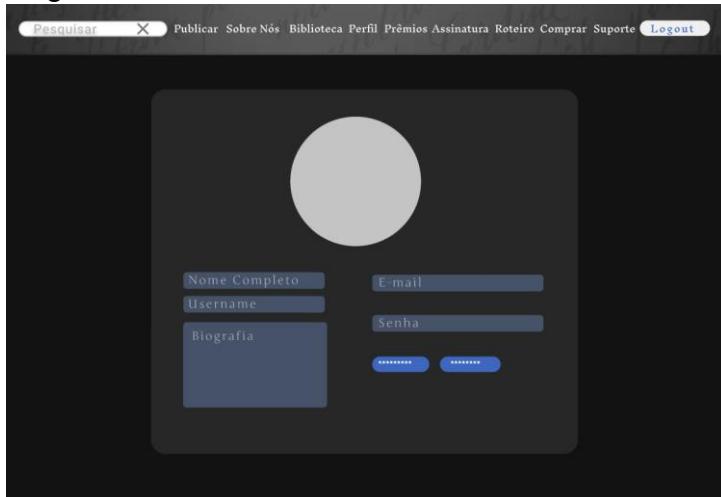
Página de Homepage 1:



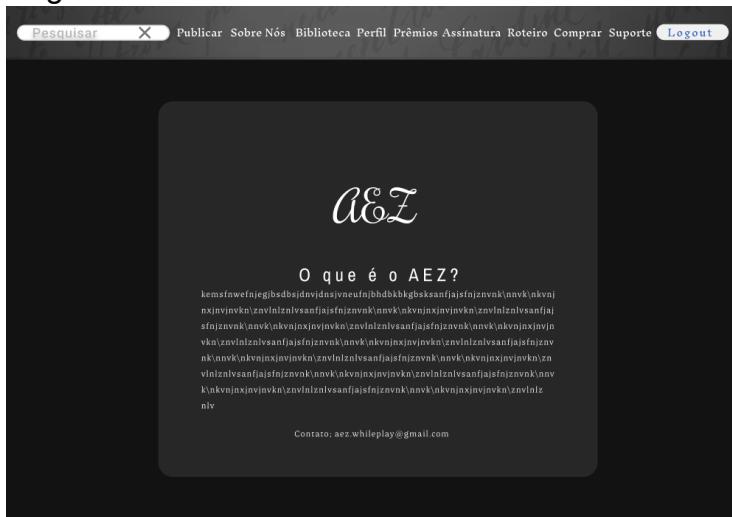
Página de Homepage 2:



### Página de Perfil:

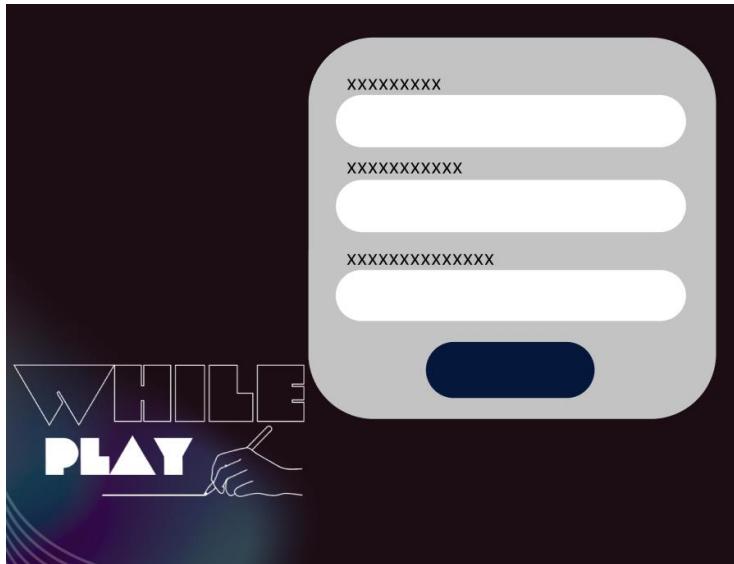


### Página de Sobre Nós:

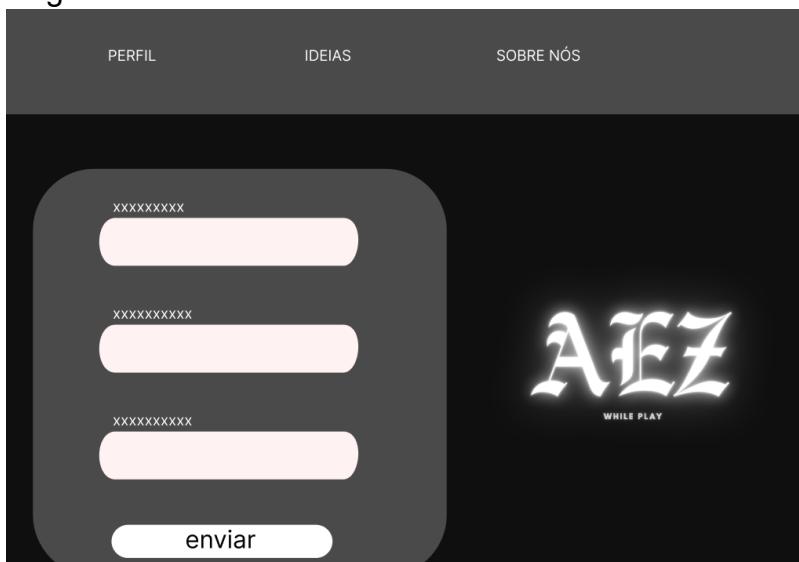


### Página de Login e Recuperar Senha:

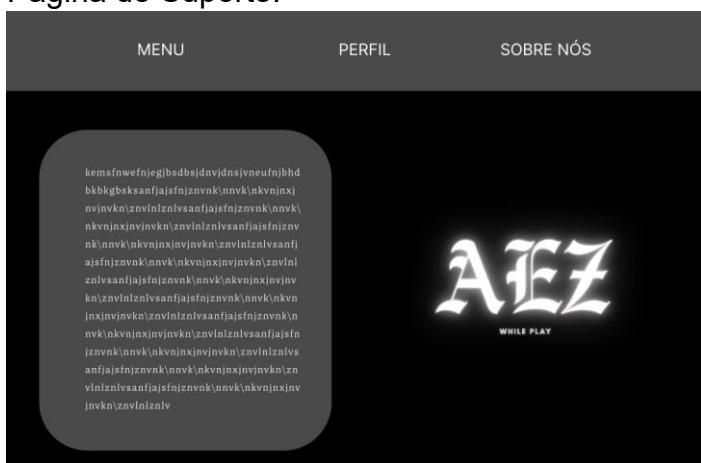




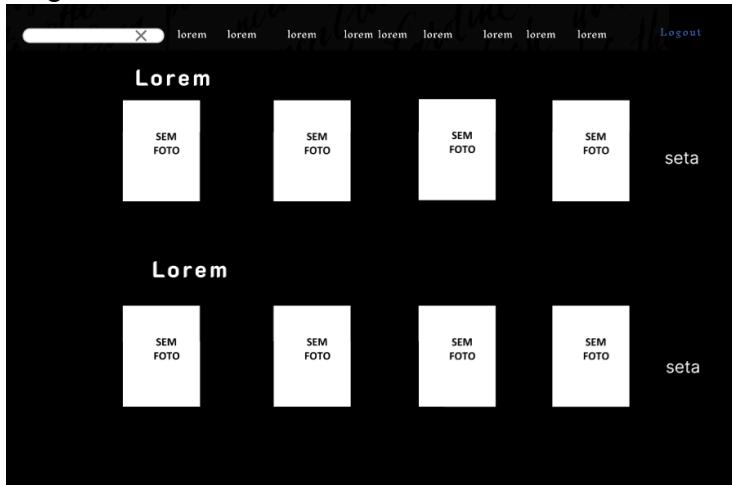
Página de Assinatura:



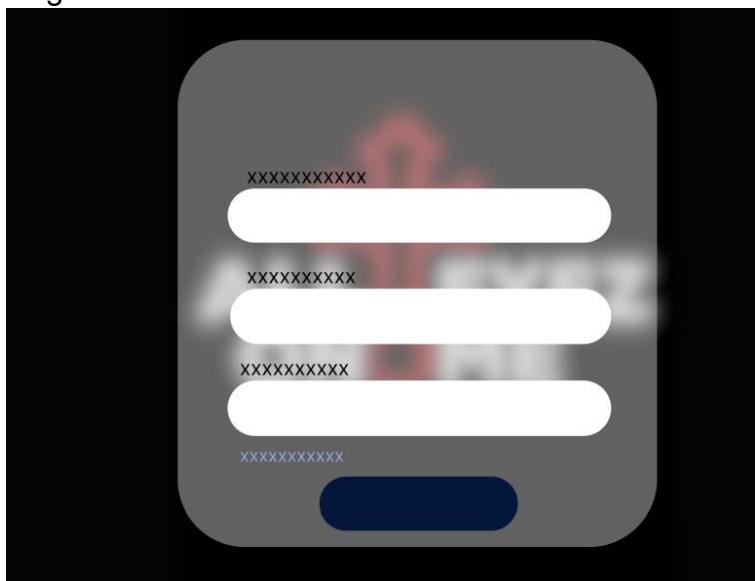
Página de Suporte:



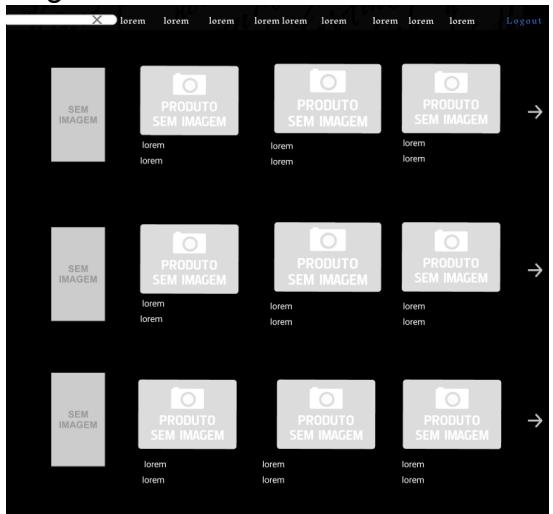
Página de Roteiro:



Página de Cadastro:



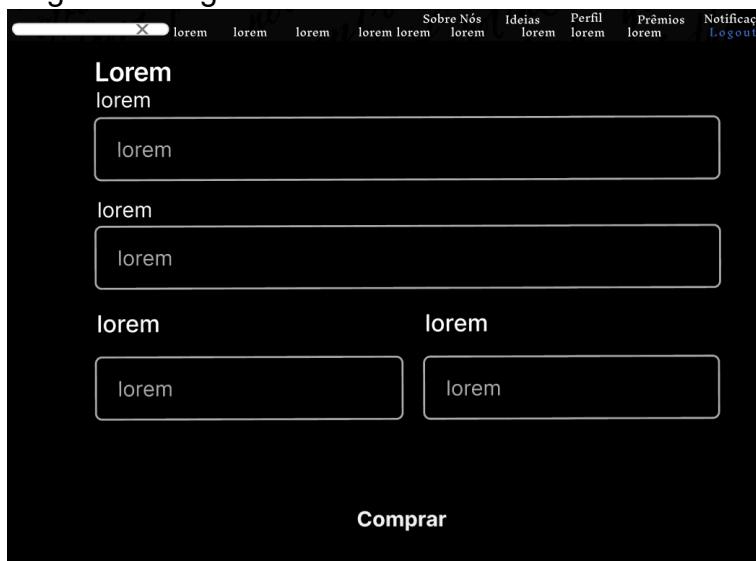
Página de Premios:



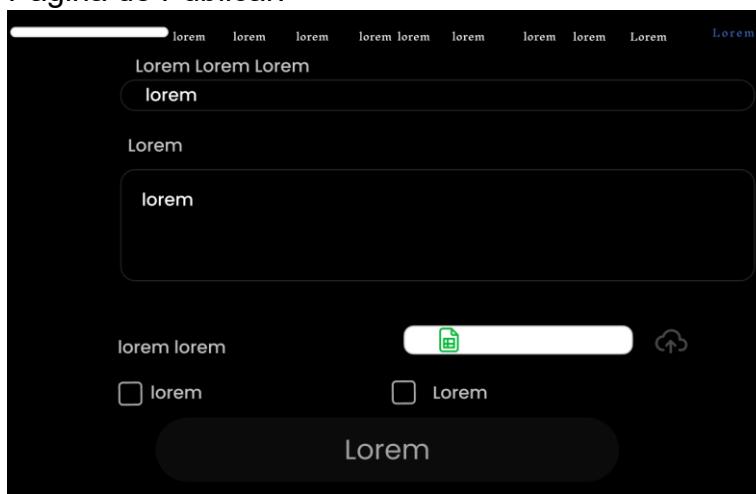
Página de Biblioteca:



Página de Pagamento:



Página de Publicar:



Formulário de inscrição:

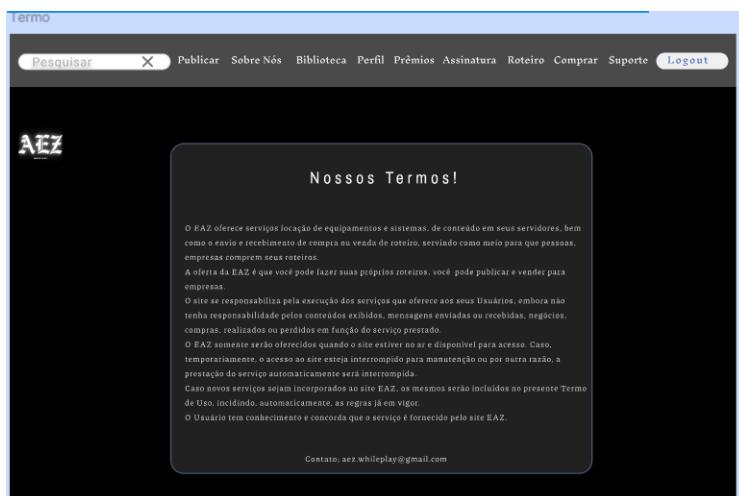
FORMULARIO DE INSCRIÇÃO

XXXXXXXXX  
 XXXXXXXXX  
 ENVIAR

- Alta fidelidade:**

Por fim fizemos o design de alta finalidade que é já é o modelo final de como queríamos que ficasse nosso site.

Página de termos do figma:



Página da homepage 1 no figma:



Página da homepage 2 no figma:

Homepage2

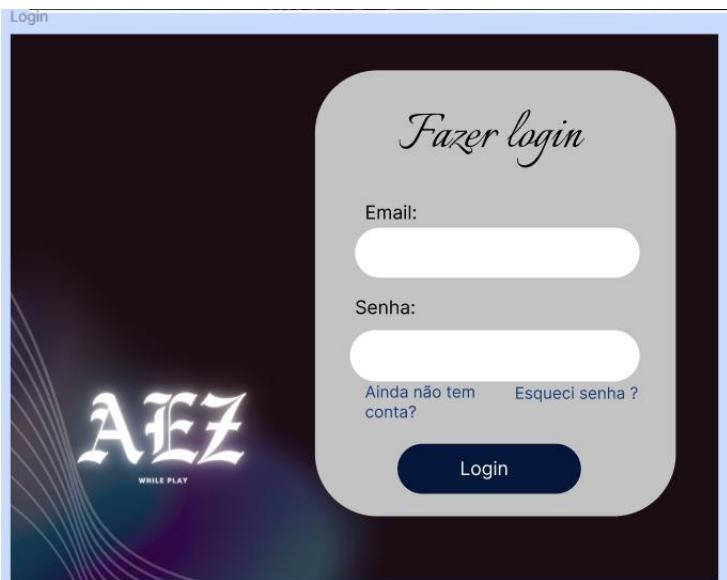


Página da homepage 3 no figma:

Homepage3-Premium

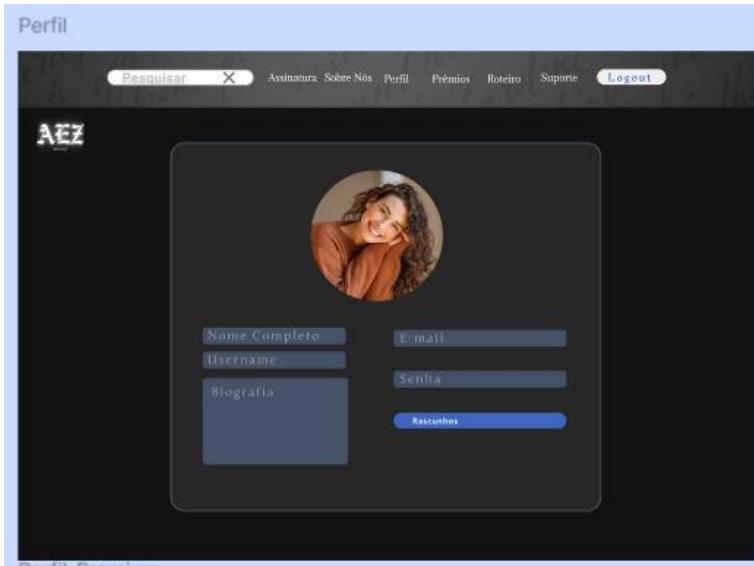


Página de fazer login do figma:



Página de pagamento do figma:

Página de perfil do figma:

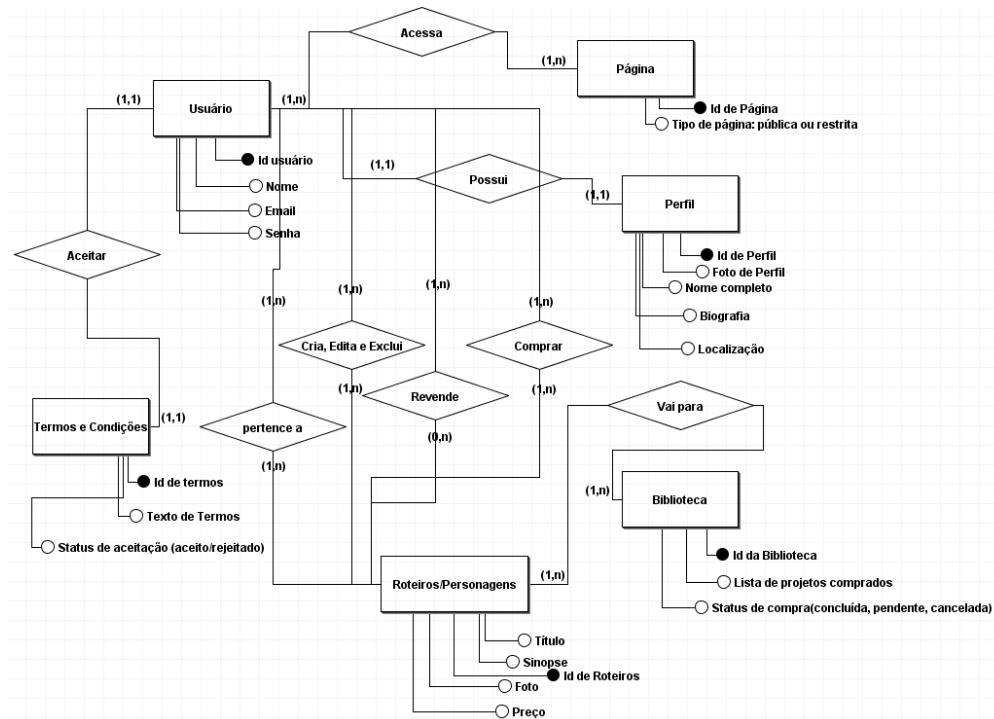


## 8 BANCO DE DADOS

O banco de dados é uma coleção de informações organizadas e estruturadas que geralmente é armazenada eletronicamente em um sistema de computador.

### 8.1 Modelo conceitual

Primeiramente fizemos o modelo conceitual também conhecido como modelo entidade-relacionamento (MER) pois ele é uma representação abstrata do banco de dados que serve como base para os modelos subsequentes. Veja a seguir o nosso modelo MER:

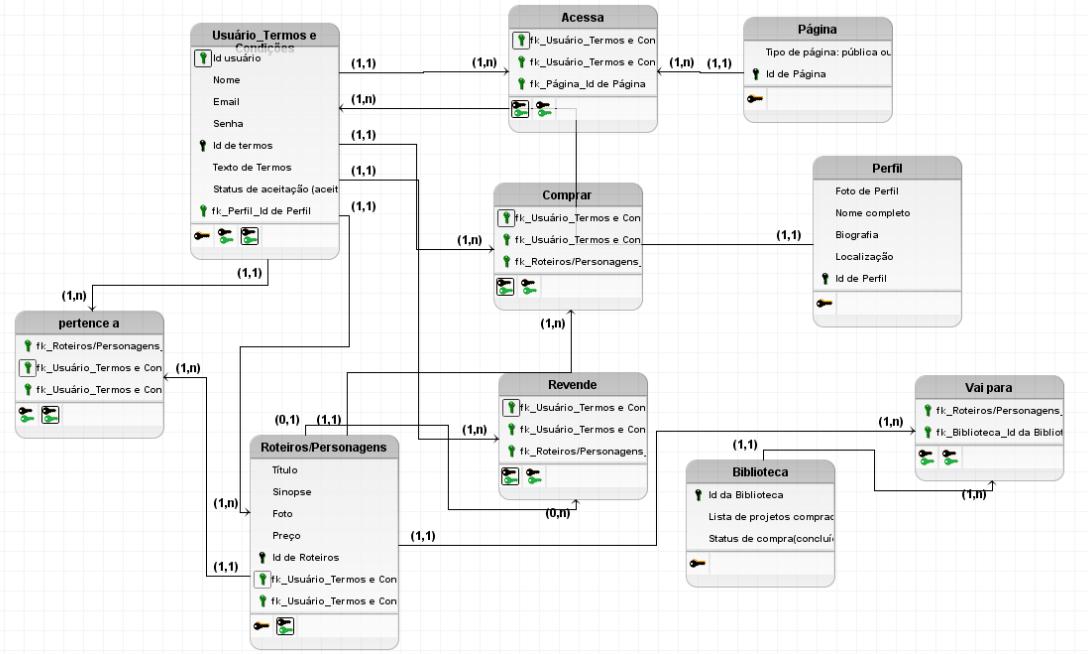


O sistema é composto por algumas entidades principais: Usuário, Roteiros/Personagens, Perfil, Biblioteca, Página e Termos e Condições. O Usuário pode ser comprador, vendedor ou visualizador e deve se cadastrar e realizar login. Ele pode criar, editar ou excluir Roteiros/Personagens, que pertencem ao usuário criador.

Cada Usuário tem um Perfil que pode editar, com dados como nome, foto e biografia. O usuário pode comprar Roteiros/Personagens, que são armazenados na Biblioteca e podem ser revendidos. Para comprar projetos, o Usuário precisa aceitar os Termos e Condições.

### 8.2 Modelo lógico

O segundo passo do meu grupo foi converter o modelo conceitual para o lógico que basicamente é uma representação mais abstrata ainda que o modelo conceitual do banco de dados, mas ele não se preocupa com detalhes de implementação. Veja a seguir nosso modelo lógico:



### 8.3 Modelo físico

Esse script cria um banco de dados chamado while\_play e define várias tabelas que, formam a base de dados necessária para a nossa plataforma de roteiros, então foram criadas as tabelas de login e cadastro onde vai conter todos os dados do usuário no nosso site, a tabela de biblioteca onde vai ser armazenado os roteiros adquiridos pelo usuário, a tabela de publicar onde vai armazenar os projetos publicados pelo usuário e a parte de pagamento onde irá ter mais informações do usuário como o cpf.

#### Script SQL da criação de tabelas:

Após algumas conversas com os professores foram feitas algumas alterações no banco e tiramos algumas tabelas que não eram necessárias:

```
create database while_play;
use while_play;
```

```
CREATE TABLE suportes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT,
    mensagem TEXT,
    data_envio DATETIME DEFAULT CURRENT_TIMESTAMP,
    status ENUM ('aberto', 'respondido', 'fechado') default 'aberto',
    foreign key (usuario_id) references perfil(id)
);
```

```
drop table suportes;
select * from suportes;
```

```
CREATE TABLE perfil (
```

```

id INT AUTO_INCREMENT PRIMARY KEY,
nome_completo VARCHAR (255) NOT NULL,
username VARCHAR (100) UNIQUE NOT NULL,
email VARCHAR (255) UNIQUE NOT NULL,
senha VARCHAR (255) NOT NULL, -- Armazene a senha criptografada (ex: bcrypt)
biografia TEXT,
foto_url VARCHAR (255), -- URL ou caminho do arquivo da imagem de perfil
data_criacao DATETIME DEFAULT CURRENT_TIMESTAMP
);
select * from perfil;

```

```

CREATE TABLE assinaturas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id int,
    nome VARCHAR (100),
    cidade VARCHAR (100),
    endereco VARCHAR (255),
    cep VARCHAR (20),
    cpf VARCHAR (14),
    data_assinatura DATETIME DEFAULT CURRENT_TIMESTAMP,
    foreign key (usuario_id) references perfil(id)
);

```

```
drop table assinaturas;
```

```

ALTER TABLE assinaturas
ADD constraint fk_assinaturas_perfil
FOREIGN KEY (usuario_id) references perfil(id);

```

```

CREATE TABLE roteiros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR (100) NOT NULL,
    categoria ENUM ('Mais bem avaliados', 'Lançados recentemente') NOT NULL,
    caminho_imagem VARCHAR (255),
    visualizacoes INT DEFAULT 0,
    assinatura_id INT,
    FOREIGN KEY (assinatura_id) REFERENCES assinaturas(id)
);

```

```
drop table roteiros;
```

```

CREATE TABLE publicar (
    id INT AUTO_INCREMENT PRIMARY KEY,

```

```

usuario_id INT,
titulo VARCHAR (255) NOT NULL,
sinopse TEXT,
tipo ENUM ('roteiro', 'personagem') NOT NULL,
arquivo_url VARCHAR (255),
data_criacao DATETIME DEFAULT CURRENT_TIMESTAMP,
publicado BOOLEAN DEFAULT FALSE,
FOREIGN KEY (usuario_id) REFERENCES perfil(id)
);

drop table publicar;

select * from publicar;
drop table personagens;
CREATE TABLE personagens (
    id_sobre INT AUTO_INCREMENT PRIMARY KEY,
    mais_bem_avaliados VARCHAR (100),
    lançados_recentemente VARCHAR (100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
INSERT INTO personagens (mais_bem_avaliados, lançados_recentemente)
VALUES
('Geralt de Rivia', 'Miles Morales'),
('Jon Snow', 'Loki'),
('Darth Vader', 'Wanda Maximoff'),
('Harry Potter', 'Stranger Things'),
('Batman', 'The Mandalorian');

select * from personagens;

CREATE TABLE pagamento (
    id_pagamento INT AUTO_INCREMENT PRIMARY KEY,
    nome_do_cartao VARCHAR (300),
    numero_do_cartao varchar(150),
    data_de_vencimento date,
    codigo decimal (4),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
insert into pagamento (nome_do_cartao, numero_do_cartao, data_de_vencimento, codigo ) values
('Livia Mayumi hayashida', '12345678909', '2001-08-09', '0987'),
('Diego gomes', '12864712547', '1980-09-12', '1234'),
('Geovanna clemente', '2314567890', '1999-12-21', '3242'),
('Victor do Vale', '9876543210', '2009-07-29', '6785');
select * from pagamento;

```

Banco montado com registros:

Tabela perfil:

	<b>id</b>	<b>nome_completo</b>	<b>username</b>	<b>email</b>	<b>senha</b>	<b>biografia</b>	<b>foto_url</b>	<b>data_criacao</b>
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela pagamento:

	<b>id_pagamento</b>	<b>nome_do_cartao</b>	<b>numero_do_cartao</b>	<b>data_de_vencimento</b>	<b>codigo</b>	<b>created_at</b>
1	Livia Mayumi hayashida	12345678909	2001-08-09	987	2025-06-18 14:47:59	
2	Diego gomes	12864712547	1980-09-12	1234	2025-06-18 14:47:59	
3	Geovanna clemente	2314567890	1999-12-21	3242	2025-06-18 14:47:59	
4	Victor do Vale	9876543210	2009-07-29	6785	2025-06-18 14:47:59	
	NULL	NULL	NULL	NULL	NULL	NULL

Tabela assinatura:

	<b>id</b>	<b>usuario_id</b>	<b>nome</b>	<b>cidade</b>	<b>endereco</b>	<b>cep</b>	<b>cpf</b>	<b>data_assinatura</b>
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela roteiros:

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content:						
	id	titulo	categoria	caminho_imagem	visualizacoes	assinatura_id
*	NULL	NULL	NULL	NULL	NULL	NULL

Tabela suporte:

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content:					
	id	usuario_id	mensagem	data_envio	status
*	NULL	NULL	NULL	NULL	NULL

Tabela publicar:

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content:								
	id	usuario_id	titulo	sinopse	tipo	arquivo_url	data_criacao	publicado
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela personagens:

	id_sobre	mais_bem_avaliados	lançados_recentemente	created_at
▶	1	Geralt de Rivia	Miles Morales	2025-06-06 13:21:04
	2	Jon Snow	Loki	2025-06-06 13:21:04
	3	Darth Vader	Wanda Maximoff	2025-06-06 13:21:04
	4	Harry Potter	Stranger Things	2025-06-06 13:21:04
	5	Batman	The Mandalorian	2025-06-06 13:21:04
*	NULL	NULL	NULL	NULL

## 9 ANÁLISE DE CUSTOS

- o Licenças de software
- o APIs pagas
- o Hospedagem e domínio
- o Materiais

A análise de custos é uma etapa muito importante para a gestão da empresa e para a tomada de decisões estratégicas pois ajuda a analisar os gastos, determinar o preço de um produto, estimar quanto custará produzir um item e avaliar a rentabilidade de produtos e serviços.

A tabela utilizada de base para atribuir os números foi a seguinte:

Entradas de Usuário: (Simples):

Cadastro: 3;  
Login: 3;  
Recuperar senha: 3;  
Assinatura: 3;  
Pagamento: 3;

Total: 15;

Saídas de usuário: (Médio):

Publicar: 5;  
Barra de pesquisa: 5;

Total: 10;

Arquivos Lógicos: (Complexo):

Roteiro: 15  
Biblioteca: 15  
Total: 30;

Consultar: (Médio):  
Suporte: 4;  
Perfil: 4;  
Sobre nós: 4;  
Prêmios: 4;

Total: 16;

Contagem total:  $15 + 10 + 30 + 16 = 71$

Soma (Fi):

Tabela:

Fator	Descrição	Pontuação de 0 a 5
F1	O sistema requer backup e recuperação confiável?	5
F2	São exigidas comunicação de Dados?	5
F3	Há funções de processamento distribuídas?	3
F4	O desempenho é crítico?	5
F5	O sistema funcionará num ambiente operacional existente, intensivamente utilizado?	4
F6	O sistema requer entrada de dados on-line?	3
F7	A entrada de dados on-line exige que a transação de entrada seja elaborada em múltiplas telas ou operações?	3
F8	Os arquivos mestres são atualizados on-line?	4
F9	A entrada, saída, arquivos ou consultas são complexos?	5
F10	O processamento interno é complexo?	3
F11	O código é projetado de forma a ser reusável?	5
F12	A conversão e a instalação estão incluídas no projeto?	5
F13	O sistema é projetado para múltiplas instalações em	5

	diferentes organizações?	
F14	A aplicação é5 projetada de forma a facilitar mudanças e o uso pelo usuário?	
Total		59

Function Points (Pontos de Função):

$$FP = \text{Contagem Total (tabela de ponderação)} \times [0,65 + 0,01 \times \text{Soma (F)}]$$

$$FP = 71 \times [0,66 \times 59]$$

$$FP = 71 \times 38,94$$

$$FP = 2.764,74$$

Tempo do Projeto:

$$\text{Tempo Total (horas)} = \text{Pontos de Função (PF)} \times \text{Horas por PF}$$

$$\text{Tempo Total} = 2.764,74 \times 9 \text{ horas}$$

$$\text{Tempo Total} = 24.882,66$$

Custo mão de obra:

$$\text{Custo mão de obra} = \text{Tempo total do projeto} * \text{Valor por hora funcionário}$$

$$\text{Custo mão de obra} = 24.882,66 \times 50,00$$

$$\text{Custo mão de obra} = 1.244.133$$

Custo por ponto de função:

$$\text{Valor por ponto de função} = \text{custo total de mão de obra} / \text{número de pontos de função (PF)}$$

$$\text{Valor por ponto de função} = 1.244.133 / 2.764,74$$

$$\text{Valor por ponto de função:} 450$$

## 10 ANÁLISE DE RISCOS

A análise de riscos é uma etapa crítica no desenvolvimento de sistemas, pois serve para antecipar problemas que possam ocorrer e minimiza impactos negativos. Segue abaixo o projeto completo com identificação, avaliação, tratamento e classificação de riscos, seguido de exemplos reais.

### 1. Identificação dos Riscos e seu impacto:

- **Ataque:** Podemos enfrentar por falha de segurança algum ataque (como SQL injection ou DDo's), além de um possível vazamento de dados, hacks ou implementação de vírus. Isso pode acarretar problemas com clientes, atrasamento e denúncias judiciais.
- **Grupo:** Podemos enfrentar atrasos, alguém pode faltar em dia de produção ou qualquer motivo pessoal, como brigas e entre outros. Isso atrasaria a entrega, resultaria em adiamento ou atrapalharia a atenção ao cliente.
- **Monitoramento:** Podemos ter problemas de monitoramento de dados, assinaturas e produtos criminosos, além da falsificação de cartões etc. Isso poderia causar golpes, problemas judiciais e uma atenção direta à causa.
- **Incompatibilidade:** Podemos encontrar dificuldades de entregar um sistema com outro ou usar uma tecnologia instável, como também o cartão incompatível.
- **Armazenamento:** Podemos ter problemas em armazenar logins, projetos lidos e comprados, como também as assinaturas. Isso poderia acabar fazendo o cliente escolher outro fornecedor, sair do site ou perder seus projetos.
- **Escalabilidade:** Podemos ter muitas compras e pessoas de pessoas acessando, podendo ter imprevistos. Isso poderia desestimular clientes, ou a possível desistência.
- **Cópia:** Podemos ter confrontamentos de cópias, assim gerando concorrências. Isso poderia nos fazer mudar algo ou depositar mais atenção e dinheiro.
- **Regulamentações:** Problemas a atender às leis de proteção de dados e dos direitos do produto de cada cliente. Isso daria problemas judiciais, além da impossibilidade de compras no site e vendas.
- **Humano:** Falhas no atendimento para o cliente. Isso chatearia o cliente, podendo causar más avaliações, a perda de clientes ou um possível processo.

Riscos	Chance	Impacto	Perigo
ATAQUE	BAIXA	ALTO	ALTO
GRUPO	BAIXA	BAIXA	BAIXA

MONITORA	MÉDIA	MÉDIA	ALTA
ARMAZENA	MÉDIA	BAIXA	BAIXA
ESCALA	ALTA	MÉDIA	BAIXA
CÓPIAS	MÉDIA	BAIXO	ALTO
REGULA	BAIXA	MÉDIA	ALTA
HUMANO	BAIXA	BAIXA	MÉDIO

O que fazer caso o erro ocorra?

- **ATAQUE:**
  - Poderíamos contratar cyber seguranças para a proteção e atualização de segurança do site, como o cloudfare(mitiga o DDo's). Podemos também adicionar etapas para dificultar e minimizar os acessos estranhos ou com coisas suspeitas.
  -
- **GRUPO e HUMANOS:**
  - Podíamos adiar um pouco a entrega até nós se ajeitarem, podíamos também repor as pessoas caso haja a desistência de algum colega e procura de uma ajudante como uma psicóloga ou terapeuta. Também devemos repassar as normas de respeitos com clientes, como cursos e consenso.
  -
- **MONITORAMENTO:**
  - Podemos antecipar isso dando regulamentos a serem feitos antes da compra, algo para ler e já identificar de onde vem o cartão.
  -
- **ARMAZENAMENTO E ESCALABILIDADE:**
  - Devemos revisar o nosso site antes, inclusive a parte de armazenamento e acessos, onde devemos rever se há problemas ao armazenar grandes quantidades de dados ou se está apto a bugs.

- **CÓPIAS:**
  - Devemos legalizar nosso site, para não haver a possibilidade de cópia utilizando uma falha nossa. Também devemos estar abertos para ideias dadas ao nosso projeto e fazer talvez uma parceria
  -
- **REGULAMENTO:**
  - Devemos rever todas as possibilidades de falhas na legislação do nosso projeto antes de lançá-lo ao ar.
  - 
  - **Quais ações preventivas podem ser tomadas para evitar que o risco aconteça?**
  -
- **ATAQUE:**
  - Devemos fechar o site momentaneamente até que o problema seja resolvido, para evitar vazamento de dados e fishing. Assim contrataríamos profissionais para que elimina o projeto e nos de um norte de como melhorar.
  -
- **GRUPO E HUMANOS:**
  - Devemos comunicar as pessoas que consumem nosso site e ir atrás de novas contratantes caso haja uma demissão. Também devemos contatar um(a) advogado(a) que cuide de possíveis denúncias ou problemas envolvendo brigas, separação e entre outras causas.
  -
- **MONITORAMENTO:**
  - Chamar um especialista para que possas nos ajudar a manter o asseguramento desses dados e nos ajudar a identificar onde está o erro ou se falta algo dessa verificação no site, além de fazer ou utilizar programas (Clearsale) para que que barre possíveis falsificações. Também devemos pedir desculpas e devolver o dinheiro caso haja um golpe com alguém.
  -
- **ARMAZENAMENTO E ESCALABILIDADE:**
  - Criar opção de backup dos arquivos que quem perdeu algo, arrumar o sistema escalável do site (como o AWS) e a quantidade de arquivos que podem conter cada cliente. Também seria importante comunicar os usuários e recompensá-los caso não conseguimos recuperar sua perda.
  -
- **CÓPIAS:**
  - Podemos caso haja um projeto que nos adaptou com uma ideia diferente, podemos fazer uma colab ou comprá-los (caso haja verba).
  -
- **REGULAMENTO:**
  - Devemos ir atrás de um profissional para que nos ajude a regulamentar tudo que ainda esteja pendente, além de ir atrás de algo perdido.
  -

**Plano de Contingência:**

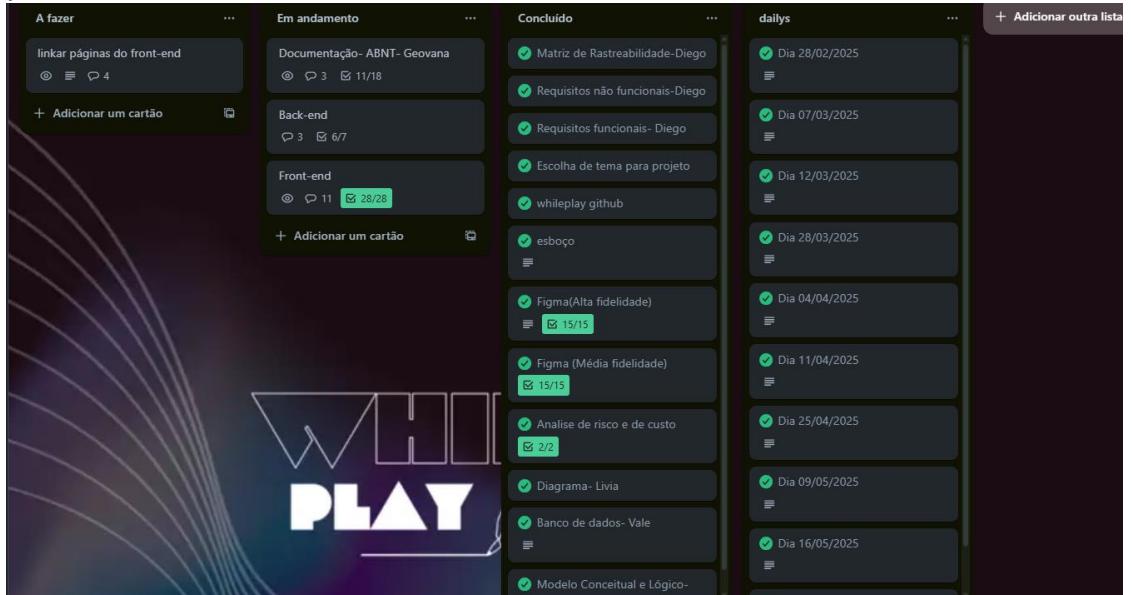
Devemos conversar entre o grupo primeiro caso haja alguns desses erros apareça em nosso caminho, ver o que devemos fazer primeiro e resolvêrmos o método de acabar com o problema. Logo devemos ir atrás de pessoas qualificadas na parte do erro e pedir uma revisão de tudo que possa ainda estar dando certo perigo ao site.

Também devemos manter os clientes informados de cada passo do site, além de também dar oportunidades de adaptações caso haja vários pedidos. Devemos garantir que todas as exigências judiciais estejam corretas e na validade, pois isso daria um enorme problema e acarretaria toda atenção nessa causa, nos distraindo do site.

Por fim devemos sempre atualizar o site para melhor segurança ou até para garantir que mais clientes entrem em nosso site, pois o mundo não para de se atualizar, então entrar em alguma moda na internet ou parcerias e aumento da proporção da nossa visibilidade dentro do cinema seria ideal para garantia de renda.

## 11 ORGANIZAÇÃO COM TRELLO

- Link do quadro Trello:



- Nosso trello foi usado em nosso projeto como forma de organizar o que cada membro deveria fazer para ajudar na conclusão de nosso projeto. Assim separamos em forma de tarefas para cada pessoa, assim quando o membro terminava sua tarefa, dava o concluir no trello e assim conseguíamos saber o que já estava pronto e o que falta. Também é valido dizer que há comentários nas postagens, onde conversamos e atualizamos nossos companheiros.

- **A fazer ou em andamento:**

- Estamos atualmente concluindo a documentação e o back-end, além de estarmos finalizando os links entre as páginas. Vale ressaltar que cada um desse itens estão separados por pessoas. As páginas que estão em produção ou quase finalizadas, diferentes das que estão como á fazer (links front-end) , essas já estão começadas e são necessárias para agora.

The screenshot shows a digital workspace interface. On the left, there is a checklist titled "Documentação- ABNT- Geovana" with various items marked as completed (checkmarks). On the right, there is a "Comentários e atividade" (Comments and activity) section with several comments from users "Diego Gomes Alves" and "Geovana Clemente Cruz". The comments discuss the documentation process and specific requirements.

A documentação foi separada para o que cada pessoa fez em seus trabalhos documentar no word, enquanto uma pessoa ajeita a documentação para seguir as normas ABNT e ser mais bonito.

The screenshot shows a digital workspace interface. On the left, there is a checklist titled "Back-end" with various items marked as completed (checkmarks). On the right, there is a "Comentários e atividade" (Comments and activity) section with comments from users "Geovana Clemente Cruz", "victor.v.souza", and "Geovana Clemente Cruz". The comments discuss the status of the project and specific tasks.

Separado por nomes o que cada pessoa deve fazer em seu trabalho, o que nesse caso é back-end. Isso se dá para termos controle de que cada pessoa está fazendo e seu trabalho.

A fazer

- linkar páginas do front-end

+ Adicionar    Ⓛ Etiquetas    ⏰ Datas    ☑ Checklist    📥 Membros

Descrição

Fazer os links corretos na página, desde os botões até os inputs.

Editar

Comentários e atividade

Mostrar Detalhes

Escrever um comentário...

Diego Gomes Alves 4 de jun. de 2025, 08:50  
Terminei todos os links que eu posso fazer por enquanto.  
+ Editar • Excluir

Diego Gomes Alves 30 de mai. de 2025, 15:23  
fiz alguns links nas páginas de logins e na sem login está tudo completo.  
+ Editar • Excluir

Diego Gomes Alves 30 de mai. de 2025, 12:40 (editado)  
Estarei começando a linkar entre as minhas páginas, quando todas estiver disponível eu farei.  
+ Editar • Excluir

Diego Gomes Alves 21 de mai. de 2025, 12:11  
Vou apenas terminar a ultima página e vou fazer os links em todas.  
+ Editar • Excluir

Diego Gomes Alves adicionou este cartão a A fazer  
30 de abr. de 2025, 10:12

Isso se vale para esse card, porém como não começamos não há ainda o que cada pessoa deve fazer.

## Concluído:

The screenshot displays a project management interface with two main sections: 'Front-end' and 'Back-end'.

**Front-end Task:**

- Owner:** Livia(HTML e CSS)
- Status:** Concluído (Completed)
- Description:** Adicione uma descrição mais detalhada...
- Progress:** 100% (green bar)
- Items:**
  - Homepage+
  - Homepage 2
  - Homepage 2-premium
  - Rodapé
  - sobre-nós-premium
  - sobre-nós
  - suporte
  - suporte-premium
  - perfil
  - perfil premium
  - termo
- Actions:** + Adicionar, Etiquetas, Datas, Checklist, Membros

**Activity Stream (Left):**

- Livia Mayumi Hayashida (LH) - 4 de jun. de 2025, 14:06: Organizando os links das páginas e o menu
- Diego Gomes Alves - 30 de mai. de 2025, 15:23: Terminei as páginas tudo de front, estou fazendo os links com as páginas disponíveis.

**Back-end Task:**

- Owner:** Diego(HTML e CSS)
- Status:** Concluído (Completed)
- Description:** Adicione parte do código na ABNT
- Progress:** 100% (green bar)
- Items:**
  - Cabeçalho
  - Login
  - Roteiro
  - Cadastro
  - Recuperar-senha
  - Personagens
  - Prêmios
  - Pagamento
  - ASSINATURA
  - publicar
  - cabeçalhos login
  - cabeçalhos assinatura
  - loading page diego gomes alves
  - pagamento confirmado
  - todas as páginas com assinatura
  - Todas as páginas com login
  - Página de carregamento
- Actions:** + Adicionar, Ocultar itens marcados, Excluir

**Activity Stream (Right):**

- Livia Mayumi Hayashida (LH) - há 2 minutos: Colocando parte do código na ABNT
- Livia Mayumi Hayashida (LH) - 4 de jun. de 2025, 14:06: Organizando os links das páginas e o menu
- Diego Gomes Alves - 30 de mai. de 2025, 15:23: Terminei as páginas tudo de front, estou fazendo os links com as páginas disponíveis.
- Diego Gomes Alves - 30 de mai. de 2025, 12:28: Estarei adicionando os rodapés nas minhas págs!!
- Diego Gomes Alves - 30 de mai. de 2025, 12:12: Além disso, estarei terminando as páginas hj front minha de assinatura e colocando os rodapés e links que são possíveis.
- Diego Gomes Alves - 30 de mai. de 2025, 12:11: Adicionei os rodapés as minhas páginas, agora irei apenas arrumar as imagens e começar a linkar!!!!

Começando pelos concluídos com as páginas front end, onde separamos por pessoas os trabalhos, como está no print em cima. Também é notável os comentários entre os integrantes.

- Matriz de Rastreabilidade-Diego
- Requisitos não funcionais-Diego
- Requisitos funcionais- Diego
- Escolha de tema para projeto
- whileplay github
- esboço
- Diagrama- Livia
- Banco de dados- Vale
- Modelo Conceitual e Lógico- Vale e Livia

Já essa etapa foi separada entre os integrantes igualmente para os membros antes de começarmos nossa programação. Desde o início para a escolha do nome, até os requisitos.

**Concluído**

**Figma(Alta fidelidade)** **+ Adicionar**

**Descrição** **Editar**

A separação das páginas que cada um vai fazer no figma ficou da seguinte forma:

**Livia** **Ocultar itens marcados** **Excluir**

100% 

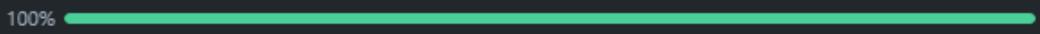
- homepage-1
- homepage-2
- sobre-nós
- perfil

**Geovana** **Ocultar itens marcados** **Excluir**

100% 

- Biblioteca
- Cadastro
- login
- Recuperar-senha

**Vale** **Ocultar itens marcados** **Excluir**

100% 

- Assinatura
- Páginas rodapé
- Suporte

**Diego** **Ocultar itens marcados** **Excluir**

100% 

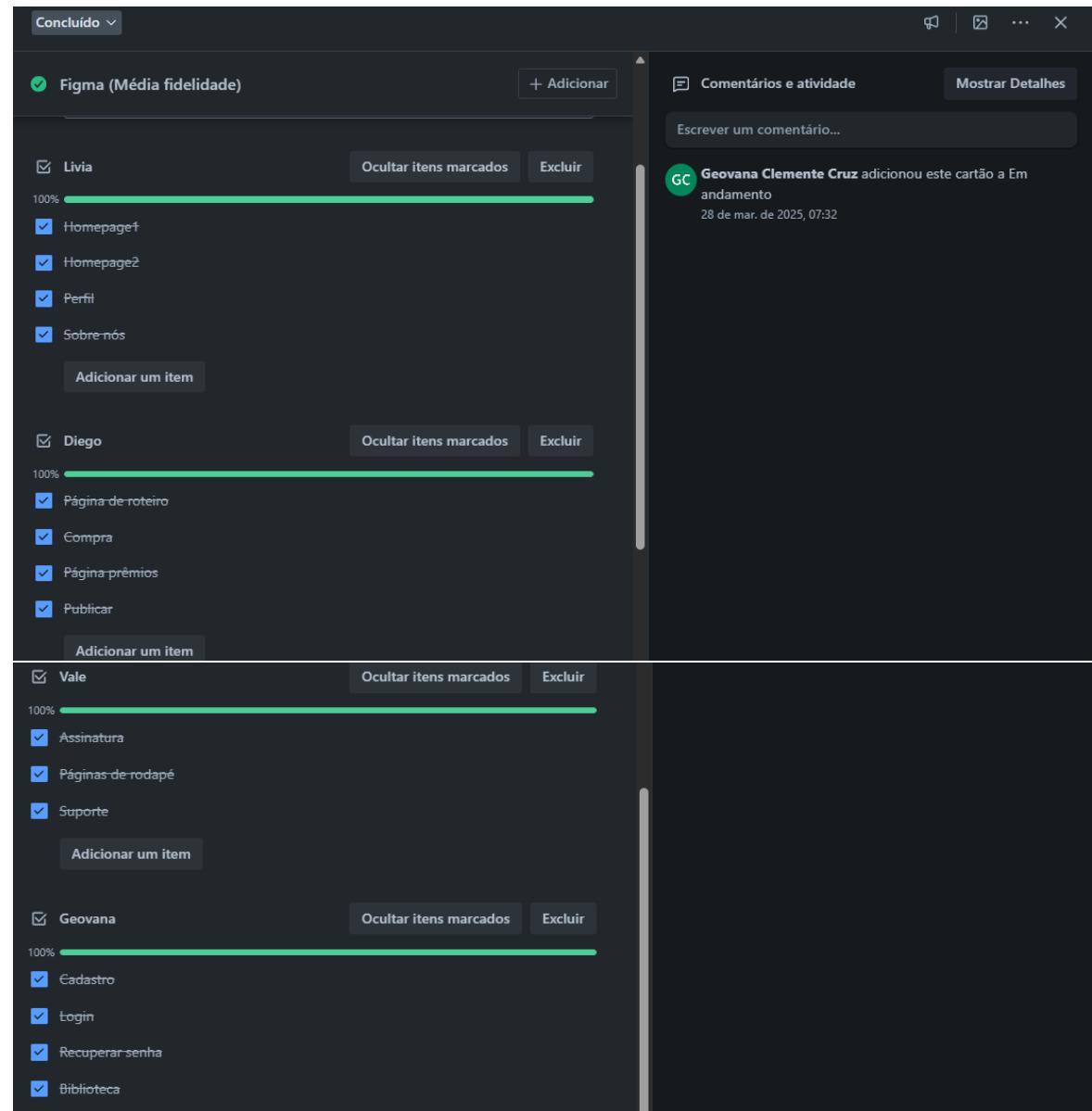
- Páginas de roteiro
- Páginas prêmios
- Compra
- Publicar

**Comentários e atividade** **Mostrar Detalhes**

**GC** Geovana Clemente Cruz adicionou este cartão a Em andamento  
7 de mar. de 2025, 14:36

**Escrever um comentário...**

Separados igualmente aos membros, cada um separou quatro páginas para fazer que iríamos fazer no figma de alta qualidade que íamos replicar no visual, o que mudamos no final pois separamos dois para o front e dois para back.



As mesmas páginas que fizemos na alta fidelidade, passamos para a média fidelidade, onde as pessoas já estavam familiarizadas com sua parte.

The screenshot shows a Trello board titled "Analise de risco e de custo". It contains two checklists:

- Analise de risco -Diego**: Status 100%, 1 item marked as completed.
- Analise de custo-Geovana**: Status 100%, 1 item marked as completed.

A comment from "Geovana Clemente Cruz" is visible on the right side of the board.

- Separamos entre dois integrantes que terminaram suas partes antes e estavam livres, onde concluímos a atividade de análises que estudamos algumas aulas anteriores da atividade.

## Dailys:

As dailys foram todas comentadas e adicionada ao nosso trello, com descrições sobre o que foi comentado na devida daily.

A daily é uma reunião rápida, feita diariamente, onde os membros da equipe compartilham o que fizeram no último dia, o que será produzido e entre outros.

The screenshot shows a Trello board titled "Dia 28/02/2025". It contains one checklist item:

- Dia 28/02/2025**: Status 100%, 1 item marked as completed.

A comment from "Diego Gomes Alves" is visible on the right side of the board.

1.

- 2.
- 
- 3.
- 
- 4.
- 
- 5.
-

- 6.
- 
- Dia 11/04/2025**
- + Adicionar Etiquetas Datas Checklist Membros
- Descrição
- Hj nós apresentamos o que já foi feito até agora na documentação(o diagrama, o levantamento de requisitos, o modelo conceitual e lógico, análise de riscos e de custo) e como está sendo a divisão de tarefas no trelio.
- Comentários e atividade
- Geovana Clemente Cruz adicionou este cartão a dailys  
11 de abr. de 2025, 15:28
- 7.
- 
- Dia 25/04/2025**
- + Adicionar Etiquetas Datas Checklist Membros
- Descrição
- Na daily de hoje vimos a estrutura do nosso banco de dados com professor , o que precisa modificar e toda a parte de backend.
- Comentários e atividade
- Geovana Clemente Cruz adicionou este cartão a dailys  
25 de abr. de 2025, 15:50
- 8.
- 
- Dia 09/05/2025**
- + Adicionar Etiquetas Datas Checklist Membros
- Descrição
- Na daily de hoje nós apresentamos como está o andamento de front-end e back-end para o professor e como está a divisão dessas tarefas e também relatamos que não há nenhum impedimento.
- Comentários e atividade
- Geovana Clemente Cruz adicionou este cartão a dailys  
9 de mai. de 2025, 15:19
- 9.
- 
- Dia 16/05/2025**
- + Adicionar Etiquetas Datas Checklist Membros
- Descrição
- Na daily de hoje nós conversamos sobre o avançamento do grupo em front e back e foram passadas algumas instruções para fazer no trelio.
- Comentários e atividade
- Geovana Clemente Cruz adicionou este cartão a dailys  
16 de mai. de 2025, 15:21
- 10.
- 
- Dia 23/05/2025**
- + Adicionar Etiquetas Datas Checklist Membros
- Descrição
- Na daily de hoje conversamos sobre o nosso banco de dados com o professor Celso e analisamos se precisava de algumas alterações.
- Comentários e atividade
- Geovana Clemente Cruz adicionou este cartão a dailys  
23 de mai. de 2025, 15:53

## 12 REPOSITÓRIO GIT/GITHUB

- Link do repositório GitHub
- Print da organização de pastas (padrão MVC)
- Explicação da estrutura

## 13 EXPLICAÇÃO DO CÓDIGO

Abaixo faremos a explicação do nosso código do back-end e front-end que justos trabalham para o desenvolvimento do nosso site:

### 13.1 Frontend

#### Cabeçalho e Rodapés:

- **Cabeçalho:**

```

<header>
    <div class="header-container">
        <div class="navbar">
            <div class="search-box">
                <input type="text" id="searchInput" placeholder="Pesquisar">
                <span onclick="clearSearch()">X</span>
            </div>

            <a href="#">Sobre Nós</a>
            <a href="#">Prêmios</a>
            <a href="#">Roteiro</a>
            <a href="#">Homepage</a>
            <a href="#">Biblioteca</a>
            <a href="#">Personagens</a>
            <a href="#">Publicar</a>

            <div class="user-profile-area">

            </div>

            <div class="profile-dropdown">
                <div class="profile-icon">
                    <span class="material-icons">person</span>
                    <!-- Badge de verificação na parte inferior direita do ícone -->
                    <div class="verified-badge">
                        <span class="material-icons">verified</span>
                    </div>
                </div>
                <div class="dropdown-content">
                    <a href="#" class="premium-option">
                        <span class="material-icons">verified</span>
                        Membro Premium
                    </a>
                </div>
            </div>
        </div>
    </div>

```

Esse código cria um header de site com uma barra de navegação contendo links para diferentes páginas, uma caixa de pesquisa com botão para limpar o texto, uma área de perfil de usuário com um ícone, que quando clicado irá para seu perfil caso tenha um, e um menu suspenso que mostra opções como "Membro Premium" caso tenha a assinatura.

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Header para Usuário Assinante</title>
7
8      <!-- Fonte Adamina -->
9      <link href="https://fonts.googleapis.com/css2?family=Adamina&display=swap" rel="stylesheet">
10     <!-- Fonte personalizada CreatoDisplay -->
11     <style>
12         @font-face {
13             font-family: 'CreatoDisplay';
14             src: url('/Whileplay/VIEWS/MEDIA/FONTE/creato_display/CreatoDisplay-ExtraBoldItalic.otf') format('opentype');
15             font-weight: normal;
16             font-style: normal;
17         }
18         body {
19             margin: 0;
20             font-family: 'CreatoDisplay', 'Adamina', serif;
21         }
22
23         header {
24             width: 100%;
25             margin: 0;
26             padding: 0;
27         }
28
29         .header-container {
30             background-image: url('/VIEWS/MEDIA/imagens/background.png');
31             background-size: cover;
32             background-position: center;
33             padding: 40px 0;
34             width: 100%;
35         }

```

Esse código define o início de nossos cabeçalhos de todas nossas páginas. Ele importa duas fontes (Adamina do Google Fonts e CreatoDisplay personalizada), remove margens do body, e aplica estilos ao header e ao container do cabeçalho, incluindo uma imagem de fundo. O objetivo é criar um visual personalizado e igual a todas as páginas.

- **Rodapé:**

```

<div class="rodape">
  <div class="producao">
    <h1>Produtores</h1>
    <p><br>Diego Alves Gomes
      <br>Geovana Clemente Cruz
      <br>Livia Mayumi Hayashida
      <br>Victor do Vale Souza
    </p>
  </div>
  <div class="suporte">
    <h1>Precisa de Suporte?</h1>
    <a href="https://www.google.com.br/">E-mail</a>
    <a href="suporte_com_login.html">Fale Conosco</a>
  </div>

  <div class="siga-nos">
    <h1>Siga-nos!</h1>
    <a href="">
      
    </a>
    <a href="">
      
    </a>
  </div>

  <p class="contato">
    Contato: aezi.whiteplay@gmail.com</p>
</div>

```

Este código define nosso rodapé (footer do site) a partir da div com classe rodapé. Assim temos três seções, `<div class="producao">`, composta por um `<h1>` que define o título e um `<p>` que apresenta os produtores juntamente com o `<br>` para as quebras de linhas, `<div class="suporte">`, com as partes de suporte para o usuário (`<a href="">...</a>`) links que simulam de envio de e-mail e página de suporte) e `<div class="siga-nos">` que também apresenta seu título `<h1>`, junto da implementação de ícones com links `<p class="contato">Contato: aezi.whiteplay@gmail.com</p>`.

## Páginas:

- Pagamento:



```

1 document.addEventListener("DOMContentLoaded", function() {
2     // Clique no botão de volta
3     const backButton = document.querySelector(".back-button");
4     backButton.addEventListener("click", function(e) {
5         e.preventDefault();
6         window.history.back(); // Volta para a página anterior no histórico do navegador
7     });
8
9     // Virar o cartão ao clicar
10    const card = document.querySelector(".card");
11    card.addEventListener("click", function() {
12        card.classList.toggle("flipped");
13    });
14
15    // Focar no CCV virá o cartão automaticamente
16    const ccvInput = document.getElementById("ccv");
17    ccvInput.addEventListener("focus", function() {
18        card.classList.add("flipped");
19    });
20
21    ccvInput.addEventListener("blur", function() {
22        card.classList.remove("flipped");
23    });
24
25    // Atualizar o número do cartão com formatação (espaço a cada 4 dígitos)
26    const numberInput = document.getElementById("number");
27    numberInput.addEventListener("input", function() {
28        let value = this.value.replace(/\D/g, "").substring(0, 16); // Remove não-dígitos e limita a 16 caracteres
29        let formattedValue = '';
30
31        // Adiciona espaço a cada 4 dígitos
32        for(let i = 0; i < value.length; i++) {
33            if(i > 0 & i % 4 === 0) {
34                formattedValue += ' ';
35            }
36            formattedValue += value[i];
37        }
38
39        this.value = formattedValue; // Atualiza o campo com a formatação
40
41        // Atualiza o visual do cartão
42        const cardNumber = document.querySelector(".card-number .mono");
43        cardNumber.textContent = formattedValue || "0000 0000 0000 0000";
44    });
45
46    // Atualizar o nome do titular
47    const holderInput = document.getElementById('holder');
48    holderInput.addEventListener("input", function() {
49        let value = this.value.replace(/\W+/g, " "); // Permite apenas letras e espaços
50        this.value = value.toUpperCase(); // Converte para maiúsculas
51
52        const cardHolder = document.querySelector(".card-holder .mono");
53        cardHolder.textContent = value || "Nome completo";
54    });
55
56    // Atualizar a data de expiração
57    const monthSelect = document.getElementById('month');
58    const yearSelect = document.getElementById('year');
59
60    function updateExpireDate() {
61        const month = monthSelect.value;
62        const year = yearSelect.value;
63        const cardDate = document.querySelector('.card-date .mono');
64        cardDate.textContent = `${month}/${year}`;
65    }
66
67    monthSelect.addEventListener('change', updateExpireDate);
68    yearSelect.addEventListener('change', updateExpireDate);
69
70    // Atualizar o CCV
71    ccvInput.addEventListener("input", function() {
72        let value = this.value.replace(/\D/g, "").substring(0, 3); // Apenas números, máximo 3
73        this.value = value;
74
75        const cardCode = document.querySelector(".card-code .mono");
76        cardCode.textContent = value || "123";
77    });
78
79    // Definir valores iniciais no cartão
80    updateExpireDate();
81 });

```

Esse código JavaScript adiciona interatividade ao formulário de cartão de crédito, permitindo que um botão com a classe `.back-button` volte para a página anterior do navegador, como também permite virar o cartão ao clicar nele (adicionando/removendo a classe `flipped`).

Formata o número do cartão enquanto o usuário digita e atualiza o visual do cartão na tela.

Atualiza o nome do titular no cartão, aceitando apenas letras e espaços, e converte para maiúsculas.

Atualiza a data de validade do cartão conforme o usuário seleciona mês e ano.

Atualiza o código CCV no cartão, aceitando apenas números e limitando a 3 dígitos.

Inicializa a data de validade do cartão ao carregar a página.

## Páginas roteiro e personagens:



```

1  $(".custom-carousel").owlCarousel({
2   autoWidth: true,
3   loop: true
4 });
5 $(document).ready(function () {
6   $(".custom-carousel .item").click(function () {
7     $(".custom-carousel .item").not($(this)).removeClass("active");
8     $(this).toggleClass("active");
9   });
10 });
11 $(".option").click(function () {
12   $(".option").removeClass("active");
13   $(this).addClass("active");
14 });
15 // Modal de ampliação dos cards de personagem
16 $(document).ready(function(){
17   $(".custom-carousel .item").on("click", function(){
18     var bg = $(this).css('background-image');
19     $('#modal-roteiro-img').css('background-image', bg);
20     var titulo = $(this).find('h3').text();
21     var texto = $(this).find('p').text();
22     $('#modal-roteiro-text').html('<strong>' + titulo + '</strong><br>' + texto);
23     // Resetar estrelas
24     $('.star-rating .star').each(function(){
25       $(this).attr('fill', 'none');
26     });
27     let rating = 0;
28     // Hover nas estrelas
29     $('.star-rating .star').off('mouseenter').on('mouseenter', function(){
30       var val = parseInt($(this).data('value'));
31       $('.star-rating .star').each(function(i){
32         $(this).attr('fill', i < val ? '#FFD700' : 'none');
33       });
34     });
35     // Sair do hover
36     $('.star-rating .star').off('mouseleave').on('mouseleave', function(){
37       $('.star-rating .star').each(function(i){
38         $(this).attr('fill', i < rating ? '#FFD700' : 'none');
39       });
40     });
41     // Clicar para avaliar
42     $('.star-rating .star').off('click').on('click', function(){
43       rating = parseInt($(this).data('value'));
44       $('.star-rating .star').each(function(i){
45         $(this).attr('fill', i < rating ? '#FFD700' : 'none');
46       });
47       // Aqui você pode adicionar lógica para salvar a avaliação
48     });
49     // Botão baixar: não faz nada
50     $('#baixar-roteiro-btn').off('click').on('click', function(){
51       // Nenhuma ação
52     });
53     // Botão comprar: redireciona para pagamento.html
54     $('#comprar-btn').off('click').on('click', function(){
55       window.location.href = '/VIEWS/pagamento/pagamento.html';
56     });
57     $('#modal-roteiro').css('display', 'flex');
58   });
59   $('#fechar-modal-roteiro').on('click', function(){
60     $('#modal-roteiro').hide();
61   });
62   // Fechar modal ao clicar fora do conteúdo
63   $('#modal-roteiro').on('click', function(e){
64     if(e.target === this) $(this).hide();
65   });
66 });

```

Esse java está presente em todas as páginas de roteiro e personagens, onde os cards apenas mudam de acordo com as páginas.

Ele: Inicializa o carrossel de personagens com Owl Carousel, permitindo rolagem infinita e largura automática.

Ao clicar em um item do carrossel, destaca o item clicado e remove o destaque dos outros.

Ao clicar em uma opção com a classe .option, marca ela como ativa.

Ao clicar em um item do carrossel, abre um modal mostrando a imagem, título e descrição do personagem.

No modal, permite avaliar com estrelas (hover e clique), mas não salva a avaliação.

O botão "Baixar Personagem" baixa o arquivo da pessoa.

O botão "Comprar" redireciona para a página de pagamento.

Permite fechar o modal clicando no botão "Fechar" ou fora do conteúdo do modal.



Esse trecho de código HTML adiciona mais um item ao carrossel de personagens, chamado "Última noiva", onde é igual a todos os outros cards, com uma imagem de fundo e uma breve descrição. Já mais embaixo no segundo código, exibe uma seção "Roteiristas Do Mês" com opções destacando roteiristas, mostrando nome, nota e imagem de fundo personalizada para cada um.

- 

```
// Função para limpar o campo de pesquisa
function clearSearch() {
    document.getElementById('searchInput').value = '';
}

// Controle da exibição dos campos de upload conforme a seleção
document.getElementById('personagem').addEventListener('change', function() {
    document.getElementById('upload-personagem').style.display = 'flex';
    document.getElementById('upload-roteiro').style.display = 'none';
});

document.getElementById('roteiro').addEventListener('change', function() {
    document.getElementById('upload-personagem').style.display = 'none';
    document.getElementById('upload-roteiro').style.display = 'flex';
});

// Mostrar nome do arquivo selecionado
document.getElementById('arquivo-personagem').addEventListener('change', function() {
    if (this.files.length > 0) {
        const fileName = this.files[0].name;
        this.nextElementSibling.textContent = fileName;
    }
});

document.getElementById('arquivo-roteiro').addEventListener('change', function() {
    if (this.files.length > 0) {
        const fileName = this.files[0].name;
        this.nextElementSibling.textContent = fileName;
    }
});

// Controle do checkbox de termos para habilitar/desabilitar o botão
document.getE (method) Document.getElementById(elementId: string): HTMLElement | null
    document.getElementById('submit-btn').disabled = !this.checked;
```

Já esse código funciona na página de publicar os arquivos, onde o código limpa o campo de pesquisa quando a função clearSearch() é chamada, alterna a exibição dos campos de upload: mostra o campo para upload de personagem (PNG) quando a opção "Personagem" é selecionada e mostra o campo para upload de roteiro (DOCX) quando a opção "Roteiro" é selecionada.

Habilita ou desabilita o botão de publicação conforme o usuário marca ou desmarca o checkbox de aceite dos termos.

## Login, esqueceu a senha e cadastro:

```

<body onclick= handleBackgroundClick(event)>
    <div class="login-container">
        <h1>Login</h1>
        <p>Bem vindo de volta criativo!</p>
        <form id="loginForm">
            <input type="email" placeholder="Email" required>
            <input type="password" placeholder="Password" required>
            <button type="submit">Logar</button>
        </form>
        <a href="recuperar_senha.html">Esqueceu sua senha?</a>
        <div class="footer"><a href="cadastro.html">Não tem login? Faça cadastro</a>
        </div>
    </div>

    <script>
        function handleBackgroundClick(event) {
            const container = document.querySelector('.login-container');
            if (!container.contains(event.target)) {
                const confirmRedirect = confirm("Deseja voltar para a homepage?");
                if (confirmRedirect) {
                    window.location.href = "homepage1.html";
                }
            }
        }
        // Redireciona para roteiros_com_login.html ao Logar
        document.addEventListener('DOMContentLoaded', function() {
            document.getElementById('loginForm').addEventListener('submit', function(e) {
                e.preventDefault();
                window.location.href = "/Whileplay/VIEWS/COM_logins/roteiros_com_login.html";
            });
        });
    </script>

```

- Já essa página de login é simples em relação a código sem ser css, onde esses simples códigos já trabalham em: exibir um formulário de login com campos para e-mail e senha, além de botões para logar, recuperar senha e cadastrar-se. Clicar fora da área do formulário, faz com que automaticamente é perguntado ao usuário se deseja voltar para a homepage; se sim, redireciona para "homepage1.html".

Ao enviar o formulário de login, impede o envio padrão e redireciona o usuário para a página "roteiros\_com\_login.html".

```

<body onclick="handleBackgroundClick(event)">
  <div class="login-container">
    <h1>Cadastro</h1>
    <p>Crie sua conta e comece agora!</p>
    <form id="cadastroForm">
      <input type="text" placeholder="Nome completo" required>
      <input type="email" placeholder="Email" required>
      <input type="password" placeholder="Senha" required>
      <button type="submit">Cadastrar</button>
    </form>
    <div class="footer">
      <a href="login.html">Já tem conta?</a>
    </div>
  </div>

<script>
  function handleBackgroundClick(event) {
    const container = document.querySelector('.login-container');
    if (!container.contains(event.target)) {
      const confirmRedirect = confirm("Deseja voltar para a homepage?");
      if (confirmRedirect) {
        window.location.href = "/homepage.html";
      }
    }
  }
  // Redireciona para Login.html ao cadastrar
  document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('cadastroForm').addEventListener('submit', function(e) {
      e.preventDefault();
      window.location.href = "login.html";
    });
  });
</script>

```

Já essa página de cadastro, segue a mesma linha de quantidade de código e o mesmo padrão visual, mudando apenas certas coisas como: Esse trecho de código exibe um formulário de cadastro com campos para nome, e-mail e senha. Quando o formulário é enviado, impede o envio padrão e redireciona para "login.html".

- ```

<body onclick="handleBackgroundClick(event)">
  <div class="recovery-container">
    <h1>Recuperar Senha</h1>
    <p>Preencha os dados abaixo para redefinir sua senha.</p>
    <form id="recoveryForm">
      <input type="email" placeholder="Email" required>
      <input type="password" id="senha1" placeholder="Nova senha" required>
      <input type="password" id="senha2" placeholder="Repetir nova senha" required>
      <button type="submit">Redefinir Senha</button>
    </form>
    <div class="footer">
      <a href="cadastro.html">Não tem conta? Cadastre-se</a>
      <a href="login.html">Lembrou sua conta? Faça seu login</a>
    </div>
  </div>

  <script>
    function handleBackgroundClick(event) {
      const container = document.querySelector('.recovery-container');
      if (!container.contains(event.target)) {
        window.history.back();
      }
    }
    // Redireciona para Login.html ao redefinir senha, só se as senhas forem iguais
    document.addEventListener('DOMContentLoaded', function() {
      document.getElementById('recoveryForm').addEventListener('submit', function(e) {
        e.preventDefault();
        const senha1 = document.getElementById('senha1').value;
        const senha2 = document.getElementById('senha2').value;
        if (senha1 !== senha2) {
          alert('As senhas digitadas não são iguais.');
          return;
        }
        // Aqui você pode adicionar a lógica para redefinir a senha, por exemplo, enviar para o backend
        // Após redefinir, redireciona para Login.html
        window.location.href = "login.html";
      });
    });
  </script>

```

Esse trecho de código exibe um formulário para redefinição de senha, onde o usuário informa o e-mail, digita a nova senha e repete a senha para confirmação. Quando o formulário é enviado, o código verifica se as duas senhas digitadas são iguais, se não forem, exibe um alerta. Se forem iguais, redefini a senha e redireciona o usuário para a página de login.

### • Homepage 1:

```

<section class="background">
  <div class="grid-container">
    <div class="container-titulos">
      
      <h1>Transforme suas Ideias em Arte</h1>
      <p>Conheça o aplicativo que te permite transformar suas ideias em roteiros e personagens. Com ferramentas</p>
    </div>

    <div class="button-container">
      <button class="login" onclick="window.location.href='login.html'">Login</button>
      <button class="cadastrar" onclick="window.location.href='cadastro.html'">Cadastrar</button>
    </div>
  </div>
</section>

```

Este código define a nossa primeira Homepage, que antecede as páginas de login e cadastro. Na parte inferior, a página apresenta dois botões interativos que direcionam o usuário para essas páginas. Já na parte superior, há um texto de apresentação com o título e uma breve descrição. Além disso, a página conta com uma imagem de fundo em tela cheia.

- **Homepage 2 (com login):**

```

1  </header>
2
3  <div class="grid-containner">
4      <div class="grid-item grid-item-1">
5          
6          <h1>Transforma suas Ideias em Filmes</h1>
7          <h2>O Site que fará sua Vida a seus Personagens e Heróis</h2>
8      </div>
9
10     <div class="grid-item grid-item-2">
11         <div class="container-roteiros">
12             <div class="contaniner-texto">
13                 <h3>O dia D</h3>
14                 <p>O Dia D é um filme de 2004 que conta a preparação para a invasão da Normandia, que ocorreu em 6 de Junho de 1944. O filme mostra os desafios enfrentados pelo Comandante Supremo Aliado na Europa, Dwight D. Eisenhower. </p>
15             </div>
16             <div class="contaniner-texto">
17                 <h3>O rato sabido</h3>
18                 <p>O rato sabido - Ratatouille é uma animação da Disney sobre um ratinho que sonha em ser chef de cozinha. </p>
19             </div>
20             <div class="contaniner-texto">
21                 <h3>Amizade entre amigos</h3>
22                 <p>Amizade entre amigos: Em 1989, algumas semanas antes da queda do muro de Berlim, três jovens italianos decidem visitar o leste Europeu à procura de aventura e partem em uma jornada que mudará suas vidas para sempre. </p>
23             </div>
24         </div>
25     </div>
26     <div class="rodape">
27         <div class="producao">
28             <ul>
29                 <li>Produzido por: Alcides Gomes</li>
30                 <li>Giovana Clemente Cruz</li>
31                 <li>Lívia Mayumi Hayashida</li>
32                 <li>Victor do Vale Souza</li>
33             </ul>
34         </div>
35         <div class="apoio">
36             <ul>
37                 <li>E-mail: https://www.google.com.br/</li>
38                 <li>Email</li>
39             </ul>
40         </div>
41         <div class="social">
42             <ul>
43                 <li>Contato: aez.whiteplay@gmail.com</li>
44             </ul>
45         </div>
46         <div class="social">
47             <ul>
48                 <li>Instagram</li>
49             </ul>
50         </div>
51         <div class="contato">
52             Contato: aez.whiteplay@gmail.com</div>
53         </div>
54     </div>

```

Este código define a segunda homepage, exibida após o login. No topo da página, há uma barra de navegação (navbar) localizada na tag <header> para o menu, que contém o menu principal e o ícone de perfil (<div class="profile-dropdown">). Logo abaixo temos três containers que trazem uma breve descrição de alguns dos roteiros do site.

- **Sobre nós com login:**

```

1 <div class="logo">
2   
3 </div>
4
5 <div class="container">
6   <h1>O que é o AEZ?</h1>
7   <p>A AEZ é uma plataforma voltada para roteiristas e criadores de personagens que buscam dar visibilidade ao seu trabalho. Nossa missão é conectar criadores com produtores, investidores e público-alvo. A AEZ oferece ferramentas para gerenciar projetos, encontrar parceiros e divulgar suas histórias de forma eficiente e impactante. Junte-se a nós e faça parte dessa nova era do entretenimento!
8
9   <h2>Equipe de desenvolvimento do projeto:</h2>
10
11  <div class="equipe">
12    <div class="circulo1"><img src="" alt="Dev 1"></div>
13    <div class="circulo2"><img src="" alt="Dev 2"></div>
14    <div class="circulo3"><img src="" alt="Dev 3"></div>
15    <div class="circulo4"><img src="" alt="Dev 4"></div>
16  </div>
17
18  <p>Contato: aez.whileplay@gmail.com</p>
19 </div>
20
21 <div class="rodape">
22   <div class="producao">
23     <h2>Produtores</h2>
24     <p><br>Diego Alves Gomes
25       <br>Geovana Clemente Cruz
26       <br>Livia Mayumi Hayashida
27       <br>Victor do Vale Souza
28     </p>
29   </div>
30   <div class="suporte">
31     <h2>Precisa de Suporte?</h2>
32     <a href="https://www.google.com.br/">E-mail</a>
33     <a href="suporte_com_login.html">Fale Conosco</a>
34   </div>
35
36   <div class="sigaa-nos">
37     <h2>Siga-nos!</h2>
38     <a href="">
39       
40     </a>
41     <a href="">
42       
43     </a>
44   </div>
45
46   <p class="contato">
47     Contato: aez.whileplay@gmail.com</p>
48 </div>
49 </body>

```

Este código define a página “Sobre Nós” com login, que utiliza o mesmo menu de navegação da segunda homepage (versão com login). Abaixo da barra de navegação, há um container contendo uma breve explicação sobre o projeto AEZ (<h1>O que é o AEZ?</h1>). Em seguida, a seção <div class="equipe"> exibe fotos da equipe de desenvolvimento. Ao final da página, há um parágrafo com as informações de contato da equipe: <p>Contato: aez.whileplay@gmail.com</p>.

- **Suporte com login:**

```

1 <div class="logo">
2   
3 </div>
4
5 <div class="container">
6   <p>Aqui estamos aqui para ajudá-lo!<br/>
7   Nossa equipe é altamente dedicada a clientes, parceiros e colaboradores, garantindo uma experiência segura, prática e personalizada em cada etapa do seu roteiro.<br/>
8   Se você tiver dúvidas, precisa de orientação sobre nossos pacotes ou deseja relatar algum problema, nossa equipe de atendimento está pronta para oferecer a assistência necessária – sempre com foco em resolver rapidamente e garantir sua satisfação.<br/>
9   Conte com a gente para tornar sua jornada ainda mais tranquila!</p>
10 </div>
11 <p>E-mail: aez.whiteplay@gmail.com</p>
12 <p>Atendimentos: das 8h às 20h</p>
13 </div>
14
15 <div class="rodape">
16   <div class="social">
17     <a href="https://www.facebook.com/aezwhiteplay/"><img alt="Icone do Facebook" style="width: 20px; height: 20px;"/></a>
18     <a href="https://www.instagram.com/aezwhiteplay/"><img alt="Icone do Instagram" style="width: 20px; height: 20px;"/></a>
19     <a href="https://www.youtube.com/channel/UCtDgkqyfJLjXWVQHmOOGwA" style="margin-left: 10px;"><img alt="Icone do YouTube" style="width: 20px; height: 20px;"/></a>
20     <a href="https://www.tiktok.com/@aezwhiteplay" style="margin-left: 10px;"><img alt="Icone do TikTok" style="width: 20px; height: 20px;"/></a>
21   </div>
22   <div class="suporte">
23     <h3>Precisa de Suporte?</h3>
24     <a href="https://www.google.com.br/">E-mail</a>
25     <a href="mailto:suporte_com_login.html">Fale Conosco</a>
26   </div>
27
28 <div class="siganos">
29   <h3>Siga-nos:</h3>
30   <img alt="Icone do Facebook" style="width: 30px; height: 30px; margin-right: 10px;"/>
31   <img alt="Icone do Instagram" style="width: 30px; height: 30px; margin-right: 10px;"/>
32   <img alt="Icone do YouTube" style="width: 30px; height: 30px; margin-right: 10px;"/>
33   <img alt="Icone do TikTok" style="width: 30px; height: 30px;"/>
34 </div>
35 </div>
36 <p class="contato">
37   Contato: aez.whiteplay@gmail.com</p>
38 </div>

```

Este código define a página de “Suporte” com login. Ela mantém a mesma barra de navegação das demais páginas e, abaixo, apresenta um container com informações de contato para dúvidas, além dos horários de atendimento da equipe.

- **Perfil com login:**

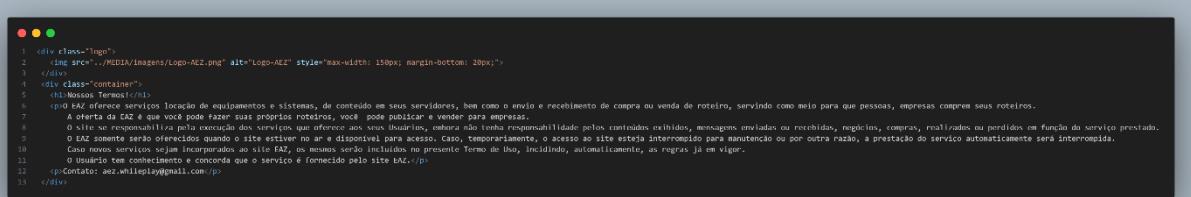
```

1 <div class="logo">
2   
3 </div>
4
5 <div class="container">
6   <div class="fotoperfil">
7     
8   </div>
9
10 <div class="form-section">
11   <div class="coluna-esquerda">
12     <div class="campo">
13       <label for="nomecompleto">Nome Completo:</label>
14       <input type="text" id="nomecompleto" name="nomecompleto">
15     </div>
16
17     <div class="campo">
18       <label for="username">Username:</label>
19       <input type="text" id="username" name="username">
20     </div>
21
22     <div class="campo">
23       <label for="biografia">Biografia:</label>
24       <textarea id="biografia" name="biografia" rows="4"></textarea>
25     </div>
26   </div>
27
28   <div class="coluna-direita">
29     <div class="campo">
30       <label for="email">E-mail:</label>
31       <input type="email" id="email" name="email">
32     </div>
33
34     <div class="campo">
35       <label for="senha">Senha:</label>
36       <input type="password" id="senha" name="senha">
37     </div>
38
39     <button class="salvar">Salvar dados</button>
40   </div>
41 </div>
42 </div>
43 </div>

```

Este código define a página de perfil do usuário com login que, assim como as demais páginas mantém a mesma barra de navegação no topo. Em seguida, um container que exibe a foto de perfil e uma `<div class="form-section">`, que contém os campos de entrada de informações (inputs) em que o usuário pode atualizar suas informações. Por fim, um botão, `<button class="salvar">Salvar dados</button>`, que registra estas informações.

- **Termo:**



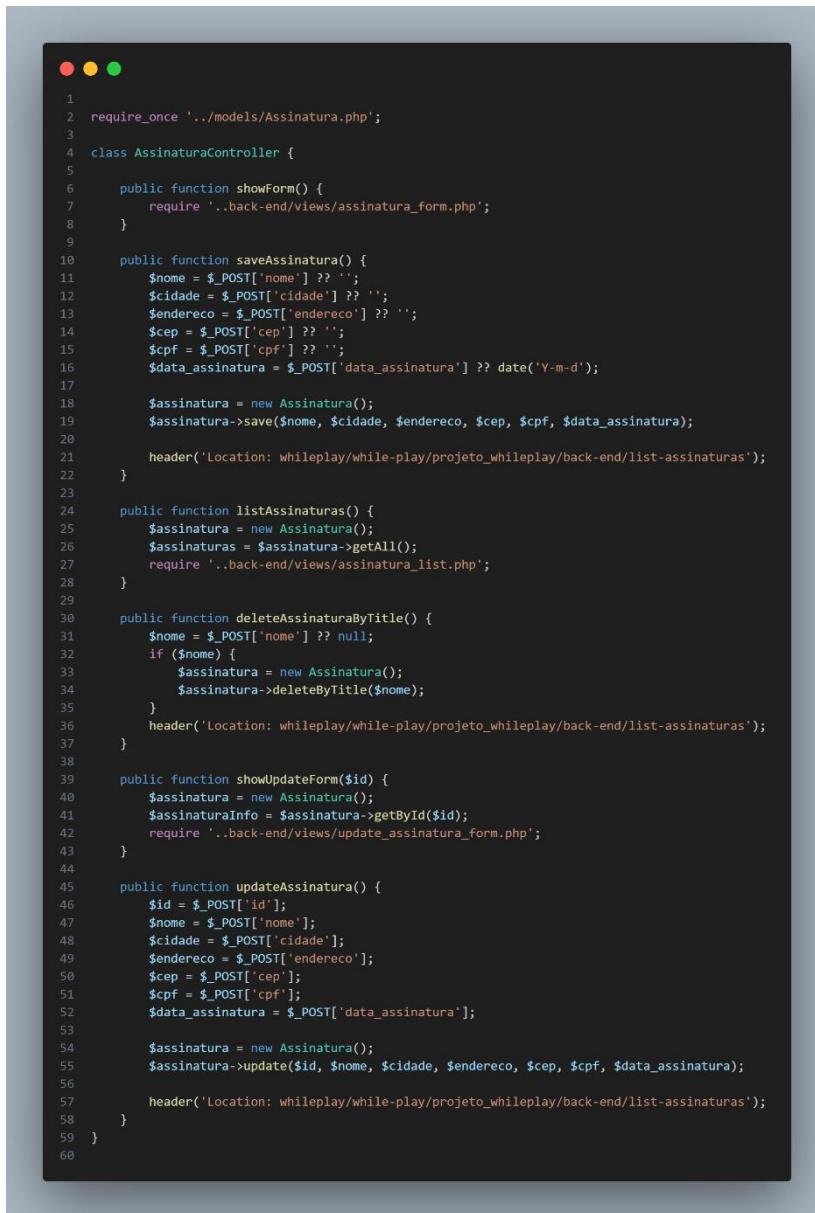
```

1 <div class="Logo">
2   
3 </div>
4 <div class="container">
5   <h1>TÉRMINOS DE USO</h1>
6   <p>A FAZ é uma plataforma destinada a fornecer locação de equipamentos e sistemas, de conteúdo em seus usuários, bem como o envio e recebimento de compra ou venda de roteiro, servindo como meio para que pessoas, empresas comprem seus roteiros.
7   <p>A oferta da FAZ é que você pode fazer suas próprias rotas, você pode publicar e vender para empresas.
8   <p>O site se responsabiliza pela execução dos serviços que oferece aos seus usuários, embora não tenha responsabilidade pelos conteúdos exibidos, mensagens enviadas ou recebidas, negociações, compras, realizados ou perdidos em função do serviço prestado.
9   <p>Os serviços serão oferecidos quando o site estiver no ar e disponível para acesso. Caso, temporariamente, o acesso ao site esteja interrompido para manutenção ou por outra razão, a prestação do serviço automaticamente será interrompida.
10  <p>Este documento tem caráter legal e concorda que o serviço é fornecido pelo site FAZ. /p>
11  <p>Contato: adz.whiteplay@gmail.com.br</p>
12 </div>

```

Este código define a página de termos de uso, que mantém a mesma barra de navegação no topo presente em quase todas as páginas. Em seguida, a estrutura apresenta um container contendo as informações sobre os termos de uso do site, incluindo orientações sobre direitos autorais e o e-mail de contato da equipe.

## 13.2 Backend



```

1 require_once '../models/Assinatura.php';
2
3 class AssinaturaController {
4
5   public function showForm() {
6     require '..back-end/views/assinatura_form.php';
7   }
8
9   public function saveAssinatura() {
10    $nome = $_POST['nome'] ?? '';
11    $cidade = $_POST['cidade'] ?? '';
12    $endereco = $_POST['endereco'] ?? '';
13    $cep = $_POST['cep'] ?? '';
14    $cpf = $_POST['cpf'] ?? '';
15    $data_assinatura = $_POST['data_assinatura'] ?? date('Y-m-d');
16
17    $assinatura = new Assinatura();
18    $assinatura->save($nome, $cidade, $endereco, $cep, $cpf, $data_assinatura);
19
20    header('Location: whileplay/while-play/projeto_whileplay/back-end/list-assinaturas');
21  }
22
23
24   public function listAssinaturas() {
25    $assinatura = new Assinatura();
26    $assinaturas = $assinatura->getAll();
27    require '..back-end/views/assinatura_list.php';
28  }
29
30   public function deleteAssinaturaByTitle() {
31    $nome = $_POST['nome'] ?? null;
32    if ($nome) {
33      $assinatura = new Assinatura();
34      $assinatura->deleteByTitle($nome);
35    }
36    header('Location: whileplay/while-play/projeto_whileplay/back-end/list-assinaturas');
37  }
38
39   public function showUpdateForm($id) {
40    $assinatura = new Assinatura();
41    $assinaturaInfo = $assinatura->getById($id);
42    require '..back-end/views/update_assinatura_form.php';
43  }
44
45   public function updateAssinatura() {
46    $id = $_POST['id'];
47    $nome = $_POST['nome'];
48    $cidade = $_POST['cidade'];
49    $endereco = $_POST['endereco'];
50    $cep = $_POST['cep'];
51    $cpf = $_POST['cpf'];
52    $data_assinatura = $_POST['data_assinatura'];
53
54    $assinatura = new Assinatura();
55    $assinatura->update($id, $nome, $cidade, $endereco, $cep, $cpf, $data_assinatura);
56
57    header('Location: whileplay/while-play/projeto_whileplay/back-end/list-assinaturas');
58  }
59 }

```

**Assinatura – Controller:** Este código define um AssinaturaController que gerencia operações de assinaturas. Ele inclui o modelo Assinatura.php para interagir com o

banco de dados. As funções do controlador permitem: exibir formulários para criação e edição (showForm, showUpdateForm), salvar novas assinaturas (saveAssinatura), listar todas as assinaturas (listAssinaturas), excluir assinaturas por nome (deleteAssinaturaByTitle) e atualizar assinaturas existentes (updateAssinatura). Cada função processa dados de formulários (via \$\_POST) e, após a operação, redireciona o usuário para a lista de assinaturas.



```

1 <?php
2
3 require_once '../models/Pagamento.php';
4
5 class PagamentoController {
6
7     public function showForm() {
8         require '..back-end/views/pagamento_form.php';
9     }
10
11    public function savePagamento() {
12        $nome_do_cartao = $_POST['nome_do_cartao'] ?? '';
13        $numero_do_cartao = $_POST['numero_do_cartao'] ?? '';
14        $data_de_vencimento = $_POST['data_de_vencimento'] ?? '';
15        $codigo = $_POST['codigo'] ?? '';
16
17        $pagamento = new Pagamento();
18        $pagamento->save($nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo);
19
20        header('Location: whileplay/while-play/projeto_whileplay/back-end/list-pagamentos');
21        exit;
22    }
23
24    public function listPagamentos() {
25        $pagamento = new Pagamento();
26        $pagamentos = $pagamento->getAll();
27        require '..back-end/views/pagamento_list.php';
28    }
29
30    public function deletePagamentoById() {
31        $id_pagamento = $_POST['id_pagamento'] ?? null;
32
33        if ($id_pagamento) {
34            $pagamento = new Pagamento();
35            $pagamento->deleteById($id_pagamento);
36        }
37
38        header('Location: whileplay/while-play/projeto_whileplay/back-end/list-pagamentos');
39        exit;
40    }
41
42    public function showUpdateForm($id) {
43        $pagamento = new Pagamento();
44        $pagamentoInfo = $pagamento->getById($id);
45        require '..back-end/views/update_pagamento_form.php';
46    }
47
48    public function updatePagamento() {
49        $id_pagamento = $_POST['id_pagamento'] ?? null;
50        $nome_do_cartao = $_POST['nome_do_cartao'] ?? '';
51        $numero_do_cartao = $_POST['numero_do_cartao'] ?? '';
52        $data_de_vencimento = $_POST['data_de_vencimento'] ?? '';
53        $codigo = $_POST['codigo'] ?? '';
54
55        if ($id_pagamento) {
56            $pagamento = new Pagamento();
57            $pagamento->update($id_pagamento, $nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo);
58        }
59
60        header('Location: whileplay/while-play/projeto_whileplay/back-end/list-pagamentos');
61        exit;
62    }
63 }

```

**Pagamento – Controller:** Este código define um PagamentoController que gerencia operações de pagamento. Ele inclui o modelo Pagamento.php para interagir com o banco de dados. As funções do controlador permitem: exibir formulários para criação (showForm), salvar novos pagamentos (savePagamento), listar todos os pagamentos (listPagamentos), excluir pagamentos por ID (deletePagamentoById), exibir formulários para edição (showUpdateForm) e atualizar pagamentos existentes

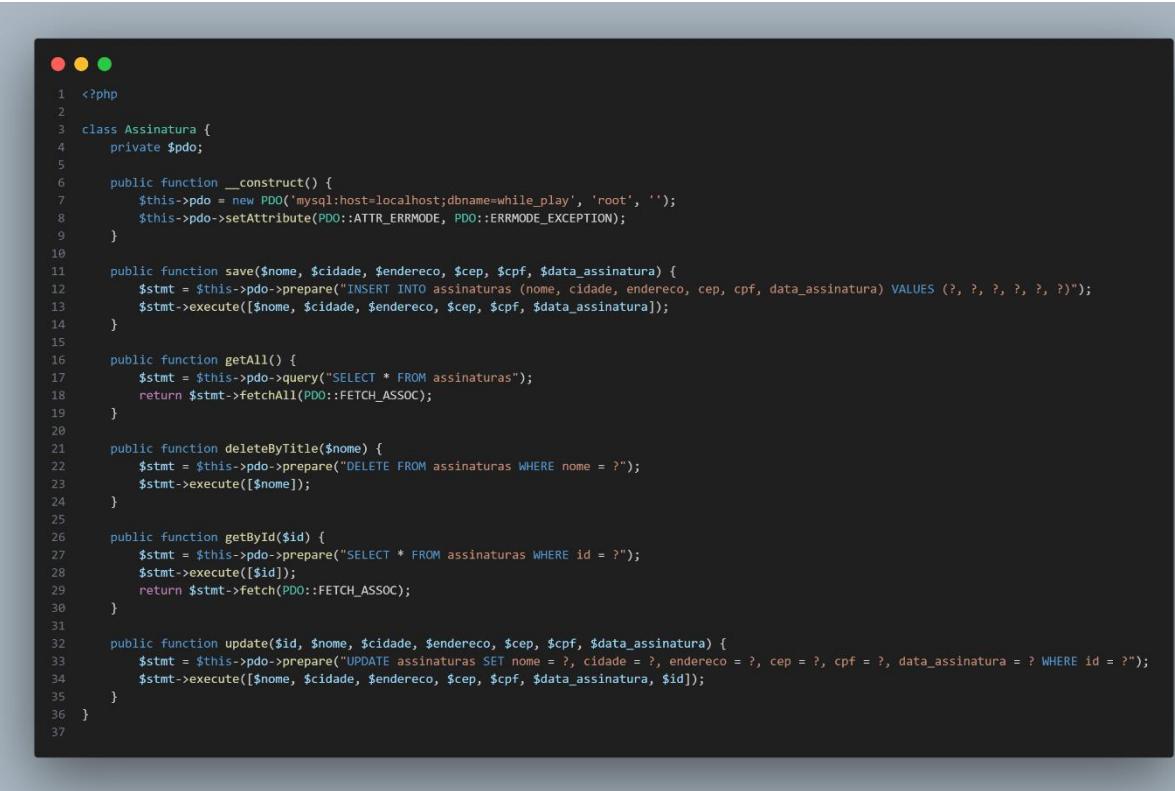
(updatePagamento). Cada função processa dados de formulários (via `$_POST`) e, após a operação, redireciona o usuário para a lista de pagamentos.



```

1 <?php
2
3 require_once '../models/Roteiro.php';
4
5 class RoteiroController {
6
7     public function showForm() {
8         require '../views/routes/form.php';
9     }
10
11     public function saveRoteiro() {
12         $titulo = $_POST['titulo'];
13         $categoria = $_POST['categoria'];
14         $visualizacoes = $_POST['visualizacoes'];
15         $assinatura_id = $_POST['assinatura_id'];
16
17         // Uclico da imagem
18         $caminho_imagem = '';
19         if (isset($_FILES['imagem'])) {
20             $error = $_FILES['imagem']['error'];
21             if ($error == UPLOAD_ERR_OK) {
22                 $tmp_name = $_FILES['imagem']['tmp_name'];
23                 $fileinfo = getimagesize($tmp_name);
24                 $ext = pathinfo($tmp_name, PATHINFO_EXTENSION);
25                 $filename = uniqid('img_'.rand()).'.'.$ext;
26                 $newfilename = $caminho_imagem.'img_'.$filename;
27                 $desctruth = uploaddir.$newfilename;
28
29                 if (move_uploaded_file($tmp_name, $desctruth)) {
30                     // caminho relativo para salvar no banco, adequado conforme estrutura do projeto
31                     $caminho_imagem = 'img/'.$filename;
32                 }
33             }
34         }
35
36         $roteiro = new Roteiro();
37         $roteiro->save($titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id);
38
39         header('location: whiteplay/whiteplay/projeto_whiteplay/back_end/list_roteiros');
40     }
41 }
42
43 public function listRoteiros() {
44     $roteiro = new Roteiro();
45     $roteiros = $roteiro->getAll();
46     require '../back_end/views/routes_list.php';
47 }
48
49 public function deleteRoteiroById($id) {
50     if ($id) {
51         $roteiro = new Roteiro();
52         $roteiro->deleteById($id);
53     }
54
55     header('location: whiteplay/whiteplay/projeto_whiteplay/back_end/list_roteiros');
56 }
57
58
59 public function showUpdateForm($id) {
60     $roteiro = new Roteiro();
61     $roteiroInfo = $roteiro->getById($id);
62     require '../views/routes_update_form.php';
63 }
64
65 public function updateRoteiro() {
66     $id = $_POST['id'];
67     $titulo = $_POST['titulo'];
68     $categoria = $_POST['categoria'];
69     $visualizacoes = $_POST['visualizacoes'];
70     $assinatura_id = $_POST['assinatura_id'];
71
72     $roteiro = new Roteiro();
73     $roteiroInfo = $roteiro->getById($id);
74
75     $caminho_imagem = $roteiroInfo['caminho_imagem']; // manter imagem antiga por padrão
76
77     // Se roteiro tem imagem, fazer upload e substituir caminho
78     if (isset($_FILES['imagem'])) {
79         $error = $_FILES['imagem']['error'];
80         if ($error == UPLOAD_ERR_OK) {
81             $tmp_name = $_FILES['imagem']['tmp_name'];
82             $fileinfo = getimagesize($tmp_name);
83             $ext = pathinfo($tmp_name, PATHINFO_EXTENSION);
84             $filename = uniqid('img_'.rand()).'.'.$ext;
85             $desctruth = uploaddir.$filename;
86
87             if (move_uploaded_file($tmp_name, $desctruth)) {
88                 $caminho_imagem = 'img/'.$filename;
89             }
90         }
91     }
92
93     $roteiro->update($id, $titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id);
94
95     header('location: whiteplay/whiteplay/projeto_whiteplay/back_end/list_roteiros');
96 }
97
98
99 }
```

**Roteiro – Controller:** Este código define um RoteiroController que gerencia operações de roteiros turísticos. Ele inclui o modelo Roteiro.php para interagir com o banco de dados. As funções do controlador permitem: exibir formulários para criação e edição (showForm, showUpdateForm), salvar novos roteiros (saveRoteiro) incluindo o upload de imagens, listar todos os roteiros (listRoteiros), excluir roteiros por ID (deleteRoteiroById) e atualizar roteiros existentes (updateRoteiro) também gerenciando a substituição de imagens. Cada função processa dados de formulários (via `$_POST` e `$_FILES` para uploads de arquivos) e, após a operação, redireciona o usuário para a lista de roteiros.



```

1 <?php
2
3 class Assinatura {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($nome, $cidade, $endereco, $cep, $cpf, $data_assinatura) {
12        $stmt = $this->pdo->prepare("INSERT INTO assinaturas (nome, cidade, endereco, cep, cpf, data_assinatura) VALUES (?, ?, ?, ?, ?, ?)");
13        $stmt->execute([$nome, $cidade, $endereco, $cep, $cpf, $data_assinatura]);
14    }
15
16    public function getAll() {
17        $stmt = $this->pdo->query("SELECT * FROM assinaturas");
18        return $stmt->fetchAll(PDO::FETCH_ASSOC);
19    }
20
21    public function deleteByTitle($nome) {
22        $stmt = $this->pdo->prepare("DELETE FROM assinaturas WHERE nome = ?");
23        $stmt->execute([$nome]);
24    }
25
26    public function getById($id) {
27        $stmt = $this->pdo->prepare("SELECT * FROM assinaturas WHERE id = ?");
28        $stmt->execute([$id]);
29        return $stmt->fetch(PDO::FETCH_ASSOC);
30    }
31
32    public function update($id, $nome, $cidade, $endereco, $cep, $cpf, $data_assinatura) {
33        $stmt = $this->pdo->prepare("UPDATE assinaturas SET nome = ?, cidade = ?, endereco = ?, cep = ?, cpf = ?, data_assinatura = ? WHERE id = ?");
34        $stmt->execute([$nome, $cidade, $endereco, $cep, $cpf, $data_assinatura, $id]);
35    }
36 }
37

```

**Assinatura – Model:** Este código define a classe Assinatura, que atua como um modelo para gerenciar a interação com o banco de dados para a tabela assinaturas. Através do PDO (PHP Data Objects), ele estabelece uma conexão com o banco de dados no construtor. As funções da classe permitem: inserir novas assinaturas (save) com base nos dados fornecidos, buscar todas as assinaturas existentes (getAll), excluir uma assinatura específica pelo nome (deleteByTitle), obter os detalhes de uma assinatura pelo seu ID (getById) e atualizar informações de uma assinatura existente (update) com base no seu ID e novos dados. Cada função executa operações SQL preparadas para garantir segurança e eficiência na comunicação com o banco de dados.

```

1 <?php
2
3 class Pagamento {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    // Salvar um novo pagamento (sem passar id, pois é auto_increment)
12    public function save($nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo) {
13        $stmt = $this->pdo->prepare("INSERT INTO pagamento (nome_do_cartao, numero_do_cartao, data_de_vencimento, codigo) VALUES (?, ?, ?, ?)");
14        $stmt->execute([$nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo]);
15    }
16
17    // Buscar todos os pagamentos
18    public function getAll() {
19        $stmt = $this->pdo->query("SELECT * FROM pagamento");
20        return $stmt->fetchAll(PDO::FETCH_ASSOC);
21    }
22
23    // Deletar pelo id_pagamento
24    public function deleteById($id_pagamento) {
25        $stmt = $this->pdo->prepare("DELETE FROM pagamento WHERE id_pagamento = ?");
26        $stmt->execute([$id_pagamento]);
27    }
28
29    // Buscar um pagamento pelo id_pagamento
30    public function getById($id_pagamento) {
31        $stmt = $this->pdo->prepare("SELECT * FROM pagamento WHERE id_pagamento = ?");
32        $stmt->execute([$id_pagamento]);
33        return $stmt->fetch(PDO::FETCH_ASSOC);
34    }
35
36    // Atualizar um pagamento pelo id pagamento
37    public function update($id_pagamento, $nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo) {
38        $stmt = $this->pdo->prepare("UPDATE pagamento SET nome_do_cartao = ?, numero_do_cartao = ?, data_de_vencimento = ?, codigo = ? WHERE id_pagamento = ?");
39        $stmt->execute([$nome_do_cartao, $numero_do_cartao, $data_de_vencimento, $codigo, $id_pagamento]);
40    }
41 }

```

**Pagamento – Model:** Este código define a classe Pagamento, que atua como um modelo para gerenciar a interação com o banco de dados para a tabela pagamento. Através do PDO (PHP Data Objects), ele estabelece uma conexão com o banco de dados no construtor. As funções da classe permitem: inserir novos pagamentos (save) com base nos dados fornecidos, buscar todos os pagamentos existentes (getAll), excluir um pagamento específico pelo ID (deleteById), obter os detalhes de um pagamento pelo seu ID (getById) e atualizar informações de um pagamento existente (update) com base no seu ID e novos dados. Cada função executa operações SQL preparadas para garantir segurança e eficiência na comunicação com o banco de dados.



```

1 <?php
2
3 class Roteiro {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id) {
12        $stmt = $this->pdo->prepare(
13            "INSERT INTO roteiros (titulo, categoria, caminho_imagem, visualizacoes, assinatura_id)
14            VALUES (?, ?, ?, ?, ?)
15        ");
16        $stmt->execute([$titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id]);
17    }
18
19    public function getAll() {
20        $stmt = $this->pdo->query("SELECT * FROM roteiros");
21        return $stmt->fetchAll(PDO::FETCH_ASSOC);
22    }
23
24    public function getById($id) {
25        $stmt = $this->pdo->prepare("SELECT * FROM roteiros WHERE id = ?");
26        $stmt->execute([$id]);
27        return $stmt->fetch(PDO::FETCH_ASSOC);
28    }
29
30    public function update($id, $titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id) {
31        $stmt = $this->pdo->prepare(
32            "UPDATE roteiros
33            SET titulo = ?, categoria = ?, caminho_imagem = ?, visualizacoes = ?, assinatura_id = ?
34            WHERE id = ?
35        ");
36        $stmt->execute([$titulo, $categoria, $caminho_imagem, $visualizacoes, $assinatura_id, $id]);
37    }
38
39    public function deleteById($id) {
40        $stmt = $this->pdo->prepare("DELETE FROM roteiros WHERE id = ?");
41        $stmt->execute([$id]);
42    }
43 }
44
45
46
47
48
49
50
51
52

```

**Roteiro – Model:** Este código define a classe Roteiro (modelo). Através do PDO (PHP Data Objects), ele estabelece uma conexão com o banco de dados no construtor. As funções da classe permitem: inserir novos roteiros (save) com base nos dados fornecidos (título, categoria, caminho da imagem, visualizações e ID da assinatura), buscar todos os roteiros existentes (getAll), obter os detalhes de um roteiro pelo seu ID (getById), atualizar informações de um roteiro existente (update) com base no seu ID e novos dados, e excluir um roteiro específico pelo ID (deleteById). Cada função executa operações SQL preparadas para garantir segurança e eficiência na comunicação com o banco de dados.

```
1 <?php
2
3 class Database {
4     private $host = 'localhost';
5     private $db_name = 'while_play';
6     private $username = 'root';
7     private $password = '';
8     public $conn;
9
10    public function getConnection() {
11        $this->conn = null;
12
13        try {
14            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this->username, $this->password);
15            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
16        } catch (PDOException $exception) {
17            echo "Connection error: " . $exception->getMessage();
18        }
19
20        return $this->conn;
21    }
22 }
23
```

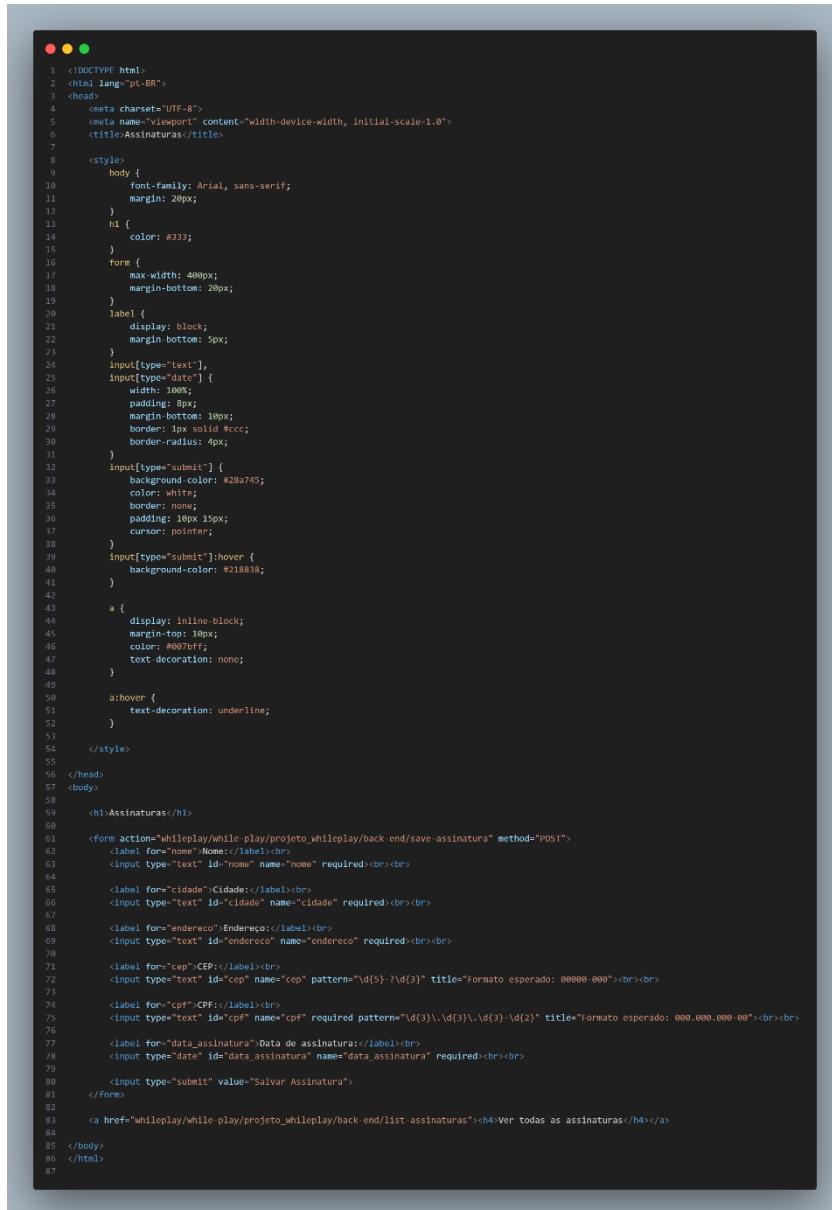
**Database – Config:** Este código define a classe Database, que é responsável por estabelecer e gerenciar a conexão com um banco de dados MySQL. A classe armazena as credenciais de conexão (servidor localhost, nome do banco while\_play, usuário root e senha vazia) como propriedades privadas. A função pública getConnection() tenta criar uma conexão PDO (PHP Data Objects) usando essas credenciais. Em caso de sucesso, ela configura o PDO para lançar exceções em caso de erros (PDO::ERRMODE\_EXCEPTION), o que facilita a depuração. Se ocorrer um erro durante a conexão, ele captura a exceção e exibe uma mensagem de erro. Finalmente, a função retorna o objeto de conexão, que pode ser usado por outras partes da aplicação para interagir com o banco de dados.

```

1 <?php
2
3 // Ativar exibição de erros para depuração
4 ini_set('display_errors', 1);
5 ini_set('display_startup_errors', 1);
6 error_reporting(E_ALL);
7
8 // Corrigir controllers
9 require_once './controllers/assassinatoController.php';
10 require_once './controllers/roteiroController.php';
11 require_once './controllers/agenteController.php';
12 require_once './controllers/publicController.php';
13 require_once './controllers/publicoController.php';
14 require_once './controllers/publicoController.php';
15 require_once './controllers/suporteController.php';
16
17 // Capturar URL da requisição
18 $request = $_SERVER['REQUEST_URI'];
19
20
21 switch ($request) {
22     case 'whiteplay/white-play/projetos:whiteplay/back-end/public/assassinato':
23         $controller = new AssassinatoController();
24         $controller->showForm();
25         break;
26
27     case 'whiteplay/white-play/projetos:whiteplay/back-end/save-assassinato':
28         $controller = new AssassinatoController();
29         $controller->createAssassinato();
30         break;
31
32     case 'whiteplay/white-play/projetos:whiteplay/back-end/list-assassinatos':
33         $controller = new AssassinatoController();
34         $controller->listAssassinatos();
35         break;
36
37     case 'whiteplay/white-play/projetos:whiteplay/back-end/delete-assassinato':
38         $controller = new AssassinatoController();
39         $controller->deleteAssassinatoById();
40         break;
41
42     case 'whiteplay/white-play/projetos:whiteplay/back-end/update-assassinato':
43         $controller = new AssassinatoController();
44         $controller->updateAssassinato();
45         break;
46
47     // Pagamento
48     case 'whiteplay/white-play/projetos:whiteplay/back-end/public/pagamento':
49         $controller = new PagamentoController();
50         $controller->showForm();
51         break;
52
53     case 'whiteplay/white-play/projetos:whiteplay/back-end/save-pagamento':
54         $controller = new PagamentoController();
55         $controller->create();
56         break;
57
58     case 'whiteplay/white-play/projetos:whiteplay/back-end/list-pagamentos':
59         $controller = new PagamentoController();
60         $controller->listPagamentos();
61         break;
62
63     case 'whiteplay/white-play/projetos:whiteplay/back-end/delete-pagamento':
64         $controller = new PagamentoController();
65         $controller->deletePagamentoById();
66         break;
67
68     case 'whiteplay/white-play/projetos:whiteplay/back-end/update-pagamento':
69         $controller = new PagamentoController();
70         $id = $matches[1];
71         $controller = new PagamentoController();
72         $controller->showUpdateForm($id);
73         break;
74
75     case 'whiteplay/white-play/projetos:whiteplay/back-end/update-pagamentos':
76         $controller = new PagamentoController();
77         $controller->updatePagamento();
78         break;
79
80     // Rotas
81     case 'whiteplay/white-play/projetos:whiteplay/back-end/public/roteiro':
82         $controller = new RoteiroController();
83         $controller->showForm();
84         break;
85
86     case 'whiteplay/white-play/projetos:whiteplay/back-end/delete-roteiro':
87         $controller = new RoteiroController();
88         $controller->deleteRoteiroById($id ?? null);
89         break;
90
91     case 'whiteplay/white-play/projetos:whiteplay/back-end/update-roteiro':
92         $id = $matches[1];
93         $controller = new RoteiroController();
94         $controller->showUpdateForm($id);
95         break;
96
97     case 'whiteplay/white-play/projetos:whiteplay/back-end/update-roteiros':
98         $controller = new RoteiroController();
99         $controller->updateRoteiros();
100        break;
101
102     case '/whiteplay/public/perfil':
103         $controller = new PerfilController();
104         $controller->showForm();
105         break;
106
107     case '/white_play/savePerfil':
108         $controller = new PerfilController();
109         $controller->create();
110         break;
111
112     case '/white_play/list-assassinato':
113         $controller = new PerfilController();
114         $controller->listPerfil();
115         break;
116
117     case '/white_play/delete-perfil':
118         require_once './controllers/PerfilController.php';
119         $controller = new PerfilController();
120         $controller->deletePerfilById();
121         break;
122
123     case '/white_play-update-perfil':
124         require_once './controllers/PerfilController.php';
125         $id = $matches[1];
126         $controller = new PerfilController();
127         $controller->showUpdateForm($id);
128         break;
129
130     case '/white_play-public/public':
131         $controller = new publicController();
132         $controller->showForm();
133         break;
134
135     case '/white_play/save-public':
136         $controller = new publicController();
137         $controller->create();
138         break;
139
140     case '/white_play/list-public':
141         $controller = new publicController();
142         $controller->listPublic();
143         break;
144
145     case '/white_play-delete-public':
146         $controller = new publicController();
147         $controller->deletePublicById();
148         break;
149
150     case '/whiteplay_update-public':
151         $id = $matches[1];
152         $controller = new publicController();
153         $controller->showUpdateForm($id);
154         break;
155
156     case '/whiteplay_update-public':
157         $controller = new publicController();
158         $controller->updatePublic();
159         break;
160
161     // Personagens
162     case '/whiteplay/public/personagens':
163         $controller = new PersonagensController();
164         $controller->showForm();
165         break;
166
167     case '/whiteplay-personagens':
168         $controller = new PersonagensController();
169         $controller->show();
170         break;
171
172     case '/whiteplay-list-personagens':
173         $controller = new PersonagensController();
174         $controller->listPersonagens();
175         break;
176
177     case '/whiteplay-delete-personagens':
178         $controller = new PersonagensController();
179         $controller->deletePersonagens();
180         break;
181
182     case '/whiteplay_update-personagens':
183         $controller = new PersonagensController();
184         $controller->showUpdateForm($id ?? null);
185         break;
186
187     case '/whiteplay_update-personagens':
188         $id = $matches[1];
189         $controller = new PersonagensController();
190         $controller->showUpdateForm($id);
191         break;
192
193     case '/whiteplay_update-personagens':
194         $controller = new PersonagensController();
195         $controller->updatePersonagens();
196         break;
197
198     // Suporte
199     case '/whiteplay/public/suporte':
200         $controller = new SuporteController();
201         $controller->showForm();
202         break;
203
204     case '/whiteplay-save-suporte':
205         $controller = new SuporteController();
206         $controller->create();
207         break;
208
209     case '/whiteplay-list-suporte':
210         $controller = new SuporteController();
211         $controller->listSuporte();
212         break;
213
214 }

```

**Index – Public:** Este código define o arquivo index.php, localizado em public. Ele utiliza a variável `$_GET['url']` para determinar qual controlador e método devem ser executados com base na URL acessada. Através de uma estrutura switch, ele compara a URL com rotas predefinidas e instancia o controlador correspondente (AssinaturaController, PagamentoController ou RoteiroController), chamando o método adequado para cada ação (como listar, salvar ou excluir registros). Se a URL não corresponder a nenhuma rota, é exibida uma mensagem de "Página não encontrada". Esse roteador é parte essencial da arquitetura MVC, conectando as requisições HTTP aos controladores, que interagem com os modelos (responsáveis pelo banco de dados) e as visões (responsáveis pela interface do usuário).



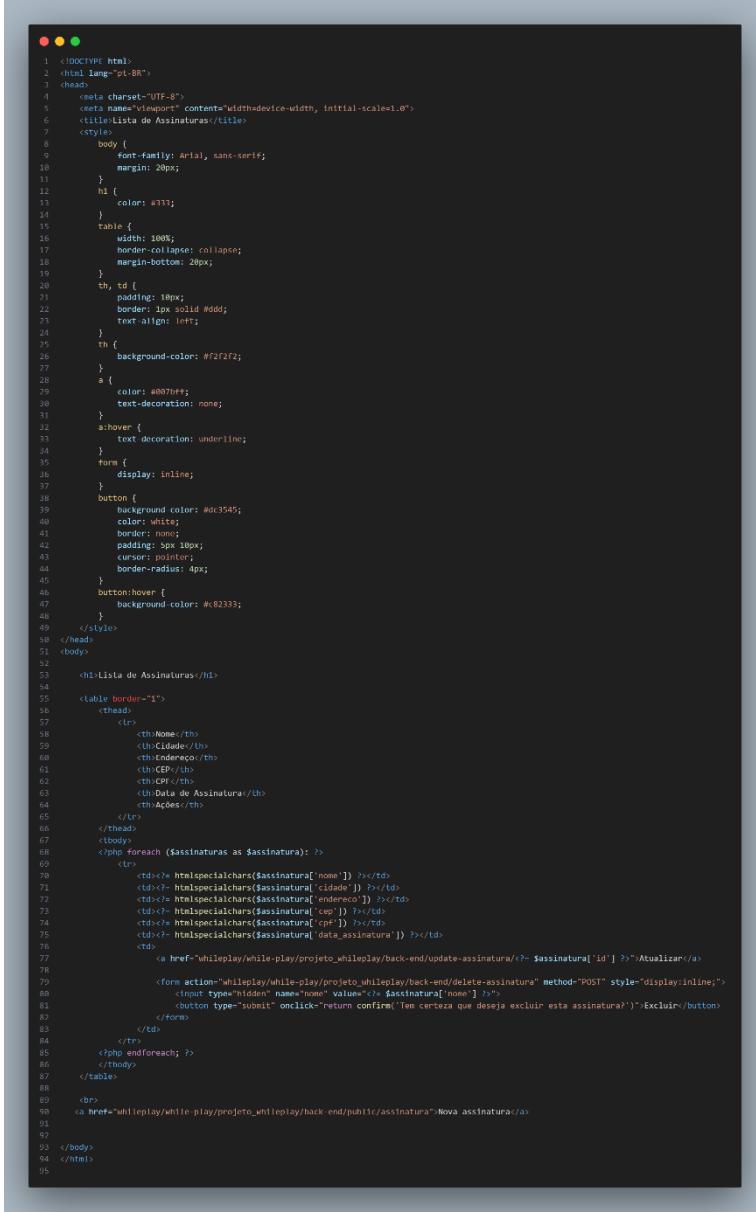
```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Assinaturas</title>
7
8   <style>
9     body {
10       font-family: Arial, sans-serif;
11       margin: 20px;
12     }
13     h1 {
14       color: #333;
15     }
16     form {
17       max-width: 400px;
18       margin-bottom: 20px;
19     }
20     label {
21       display: block;
22       margin-bottom: 5px;
23     }
24     input[type="text"],
25     input[type="date"] {
26       width: 100%;
27       padding: 8px;
28       margin-bottom: 10px;
29       border: 1px solid #ccc;
30       border-radius: 4px;
31     }
32     input[type="submit"] {
33       background-color: #28a745;
34       color: white;
35       border: none;
36       padding: 10px 15px;
37       cursor: pointer;
38     }
39     input[type="submit"]:hover {
40       background-color: #218838;
41     }
42     a {
43       display: inline-block;
44       margin-top: 10px;
45       color: #00bbff;
46       text-decoration: none;
47     }
48     a:hover {
49       text-decoration: underline;
50     }
51   </style>
52 </head>
53 <body>
54   <h1>Assinaturas</h1>
55   <form action="whileplay/whileplay/projeto/whileplay/back_end/save-assinatura" method="POST">
56     <label for="nome">Nome:</label><br>
57     <input type="text" id="nome" name="nome" required><br><br>
58     <label for="cidade">Cidade:</label><br>
59     <input type="text" id="cidade" name="cidade" required><br><br>
60     <label for="endereco">Endereço:</label><br>
61     <input type="text" id="endereco" name="endereco" required><br><br>
62     <label for="cep">CEP:</label><br>
63     <input type="text" id="cep" name="cep" pattern="\d{5}-\d{3}" title="Formato esperado: 00000-000"><br><br>
64     <label for="cpf">CPF:</label><br>
65     <input type="text" id="cpf" name="cpf" required pattern="\d{3}.\d{3}.\d{3}.\d{2}" title="Formato esperado: 000.000.000-00"><br><br>
66     <label for="data_assinatura">Data de assinatura:</label><br>
67     <input type="date" id="data_assinatura" name="data_assinatura" required><br><br>
68     <input type="submit" value="Salvar Assinatura">
69   </form>
70   <a href="whileplay/whileplay/projeto/whileplay/back_end/list_assinaturas"><h4>Ver todas as assinaturas</h4></a>
71
72 </body>
73 </html>

```

**Assinatura Form – Views:** O código é usado para coletar dados de "assinatura" como nome, cidade, endereço, CEP, CPF e data. Todos os campos são obrigatórios e

possuem validação básica. Ao ser enviado, o formulário envia os dados via POST para uma rota do backend (/save-assinatura), que provavelmente é processada por um controlador PHP. Há também um link para visualizar todas as assinaturas cadastradas. O código representa a parte visual (frontend) da aplicação, enquanto o processamento ocorre no backend.

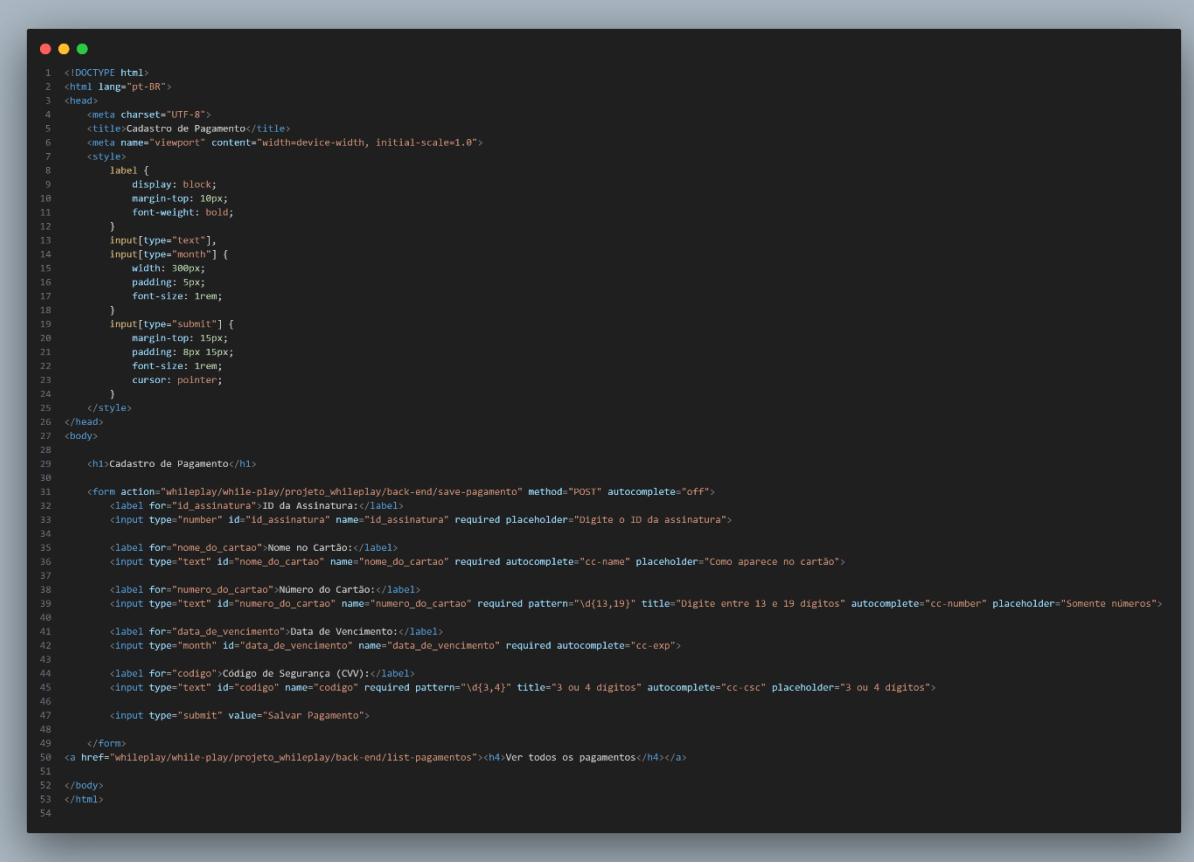


```

1 <!DOCTYPE Html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Lista de Assinaturas</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10      margin: 20px;
11    }
12    h1 {
13      color: #333;
14    }
15    table {
16      width: 100%;
17      border-collapse: collapse;
18      margin-bottom: 20px;
19    }
20    th, td {
21      padding: 10px;
22      border: 1px solid #ddd;
23      text-align: left;
24    }
25    th {
26      background-color: #f2f2f2;
27    }
28    a {
29      color: #007bff;
30      text-decoration: none;
31    }
32    a:hover {
33      text-decoration: underline;
34    }
35    form {
36      display: inline;
37    }
38    button {
39      background-color: #dc3545;
40      color: white;
41      border: none;
42      padding: 5px 10px;
43      cursor: pointer;
44      border-radius: 4px;
45    }
46    button:hover {
47      background-color: #823333;
48    }
49  </style>
50 </head>
51 <body>
52   <h1>Lista de Assinaturas</h1>
53   <table border="1">
54     <thead>
55       <tr>
56         <th>Nome</th>
57         <th>Cidade</th>
58         <th>Endereço</th>
59         <th>CEP</th>
60         <th>CPF</th>
61         <th>Data de Assinatura</th>
62         <th>Ações</th>
63       </tr>
64     </thead>
65     <tbody>
66       <?php foreach ($assinaturas as $assinatura): ?>
67       <tr>
68         <td><?php echo htmlspecialchars($assinatura['nome']) ?></td>
69         <td><?php echo htmlspecialchars($assinatura['cidade']) ?></td>
70         <td><?php echo htmlspecialchars($assinatura['endereco']) ?></td>
71         <td><?php echo htmlspecialchars($assinatura['cep']) ?></td>
72         <td><?php echo htmlspecialchars($assinatura['cpf']) ?></td>
73         <td><?php echo htmlspecialchars($assinatura['data_assinatura']) ?></td>
74         <td>
75           <a href="whileplay/whiteplay/projeto_whileplay/back-end/update-assinatura/?=<?php echo $assinatura['id'] ?>">Atualizar</a>
76           <br>
77           <form action="whileplay/whiteplay/projeto_whileplay/back-end/delete-assinatura" method="POST" style="display:inline;">
78             <input type="hidden" name="name" value=<?php echo $assinatura['name'] ?>>
79             <button type="submit" onclick="return confirm('Tem certeza que deseja excluir esta assinatura?')>Excluir</button>
80           </form>
81         </td>
82       </tr>
83     <?php endforeach; ?>
84   </tbody>
85 </table>
86   <br>
87   <a href="whileplay/whiteplay/projeto_whileplay/back-end/public/assinatura">Nova assinatura</a>
88 </body>
89 </html>
90
91
92
93
94
95

```

**Assinatura List – Views:** O código é usado para listar as assinaturas já cadastradas em formato de tabela. Cada linha da tabela exibe dados como nome, cidade, endereço, CEP, CPF e data de assinatura. Também há duas ações disponíveis para cada item: um link para atualizar a assinatura e um botão para excluí-la, com confirmação.

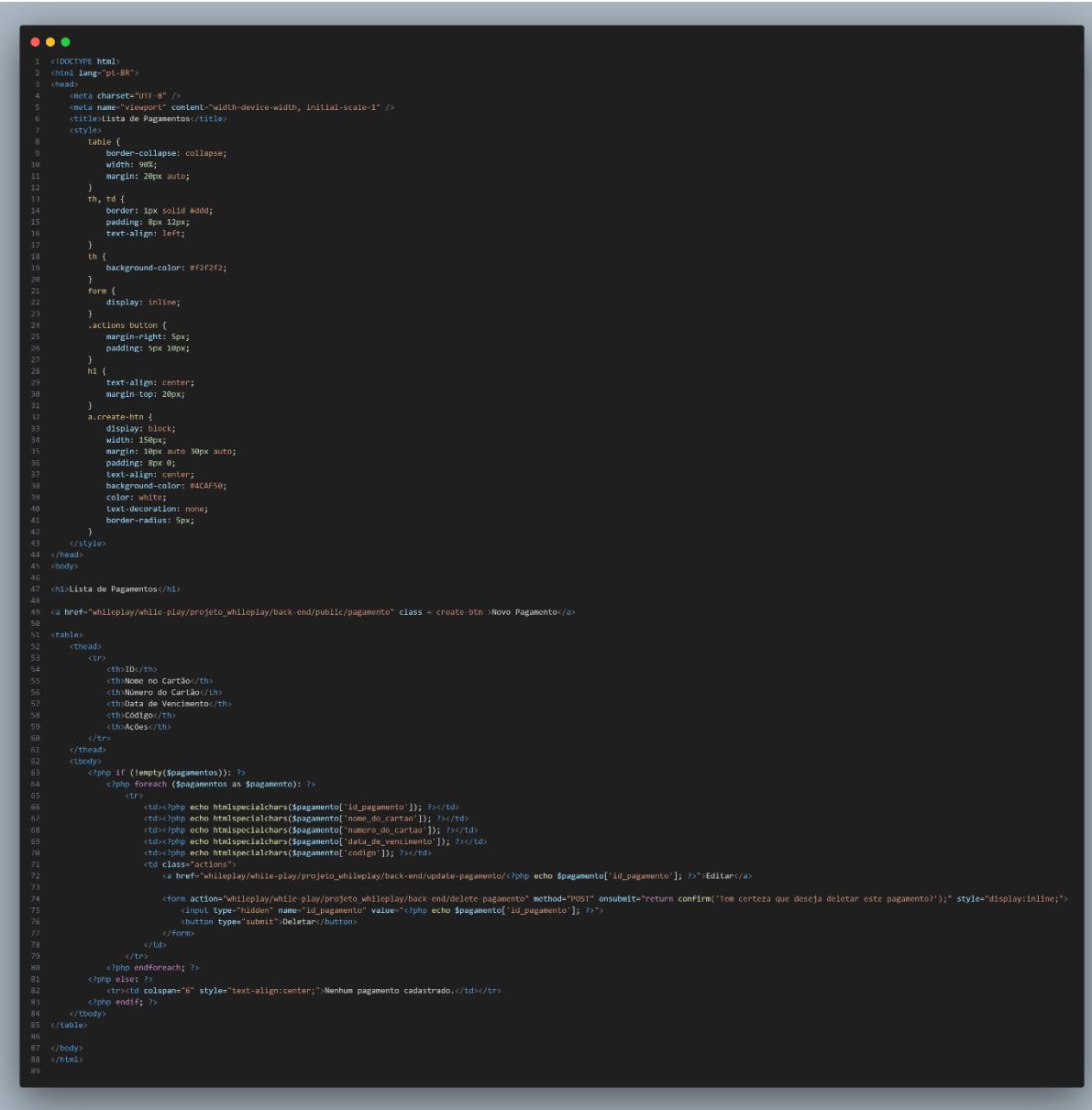


```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Cadastro de Pagamento</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <style>
8     label {
9       display: block;
10      margin-top: 10px;
11      font-weight: bold;
12    }
13    input[type="text"],
14    input[type="month"] {
15      width: 300px;
16      padding: 5px;
17      font-size: 1rem;
18    }
19    input[type="submit"] {
20      margin-top: 15px;
21      padding: 8px 15px;
22      font-size: 1rem;
23      cursor: pointer;
24    }
25  </style>
26 </head>
27 <body>
28   <h1>Cadastro de Pagamento</h1>
29   <form action="whileplay/while-play/projeto_whileplay/back-end/save-pagamento" method="POST" autocomplete="off">
30     <label for="id_assinatura">ID da Assinatura:</label>
31     <input type="number" id="id_assinatura" name="id_assinatura" required placeholder="Digite o ID da assinatura">
32
33     <label for="nome_do_cartao">Nome no Cartão:</label>
34     <input type="text" id="nome_do_cartao" name="nome_do_cartao" required autocomplete="cc-name" placeholder="Como aparece no cartão">
35
36     <label for="numero_do_cartao">Número do Cartão:</label>
37     <input type="text" id="numero_do_cartao" name="numero_do_cartao" required pattern="\d{13,19}" title="Digite entre 13 e 19 dígitos" autocomplete="cc-number" placeholder="Somente números">
38
39     <label for="data_de_vencimento">Data de Vencimento:</label>
40     <input type="month" id="data_de_vencimento" name="data_de_vencimento" required autocomplete="cc-exp">
41
42     <label for="codigo">Código de Segurança (CVV):</label>
43     <input type="text" id="codigo" name="codigo" required pattern="\d{3,4}" title="3 ou 4 dígitos" autocomplete="cc-csc" placeholder="3 ou 4 dígitos">
44
45     <input type="submit" value="Salvar Pagamento">
46   </form>
47   <a href="whileplay/while-play/projeto_whileplay/back-end/list-pagamentos"><h4>Ver todos os pagamentos</h4></a>
48
49 </body>
50 </html>
51
52
53
54

```

**Pagamento Form – Views:** O código é usado para coletar dados de pagamento como ID da assinatura, nome no cartão, número do cartão, data de vencimento e código de segurança. Todos os campos são obrigatórios e possuem validação básica por meio de atributos HTML. Ao ser enviado, o formulário envia os dados via POST para uma rota do backend (/save-pagamento).



```

1 <!DOCTYPE html>
2 
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1" />
6   <title>Lista de Pagamentos</title>
7   <style>
8     table {
9       border-collapse: collapse;
10      width: 90%;
11      margin: 20px auto;
12    }
13    th, td {
14      border: 1px solid #000;
15      padding: 8px 10px;
16      text-align: left;
17    }
18    th {
19      background-color: #f2f2f2;
20    }
21    form {
22      display: inline;
23    }
24    .actions button {
25      margin-right: 5px;
26      padding: 5px 10px;
27    }
28    h1 {
29      text-align: center;
30      margin-top: 20px;
31    }
32    a.create-btn {
33      display: inline-block;
34      width: 100px;
35      margin-left: 10px auto 30px auto;
36      padding: 8px 0;
37      text-align: center;
38      background-color: #dcaf50;
39      color: white;
40      text-decoration: none;
41      border-radius: 5px;
42    }
43  </style>
44 </head>
45 <body>
46
47 <h1>Lista de Pagamentos</h1>
48 <a href="whileplay/while-play/projeto_whileplay/back-end/public/pagamento" class="create-btn">Novo Pagamento</a>
49
50 <table>
51   <thead>
52     <tr>
53       <th>ID</th>
54       <th>Nome no Cartão</th>
55       <th>Número do Cartão</th>
56       <th>Data de Vencimento</th>
57       <th>Código</th>
58       <th>Ações</th>
59     </tr>
60   </thead>
61   <tbody>
62     <php if (!empty($pagamentos)); >>
63     <php foreach ($pagamentos as $pagamento); >>
64       <tr>
65         <td><php echo htmlspecialchars($pagamento['id_pagamento']); ></td>
66         <td><php echo htmlspecialchars($pagamento['nome_do_cartao']); ></td>
67         <td><php echo htmlspecialchars($pagamento['numero_do_cartao']); ></td>
68         <td><php echo htmlspecialchars($pagamento['data_de_vencimento']); ></td>
69         <td><php echo htmlspecialchars($pagamento['codigo']); ></td>
70         <td class="actions">
71           <a href="whileplay/while-play/projeto_whileplay/back-end/update-pagamento/<php echo $pagamento['id_pagamento']; ?>">Editar</a>
72
73           <form action="whileplay/while-play/projeto_whileplay/back-end/delete_pagamento" method="POST" onsubmit="return confirm('Tem certeza que deseja deletar este pagamento?');" style="display:inline;">
74             <input type="hidden" name="id_pagamento" value="<php echo $pagamento['id_pagamento']; ?>">
75             <button type="submit">Delete</button>
76           </form>
77         </td>
78       </tr>
79     <php endforeach; >
80   </tbody>
81   <tr><td colspan="6" style="text-align:center;">Nenhum pagamento cadastrado.</td></tr>
82   <php endif; >
83 </table>
84 </body>
85 </html>
86
87
88
89

```

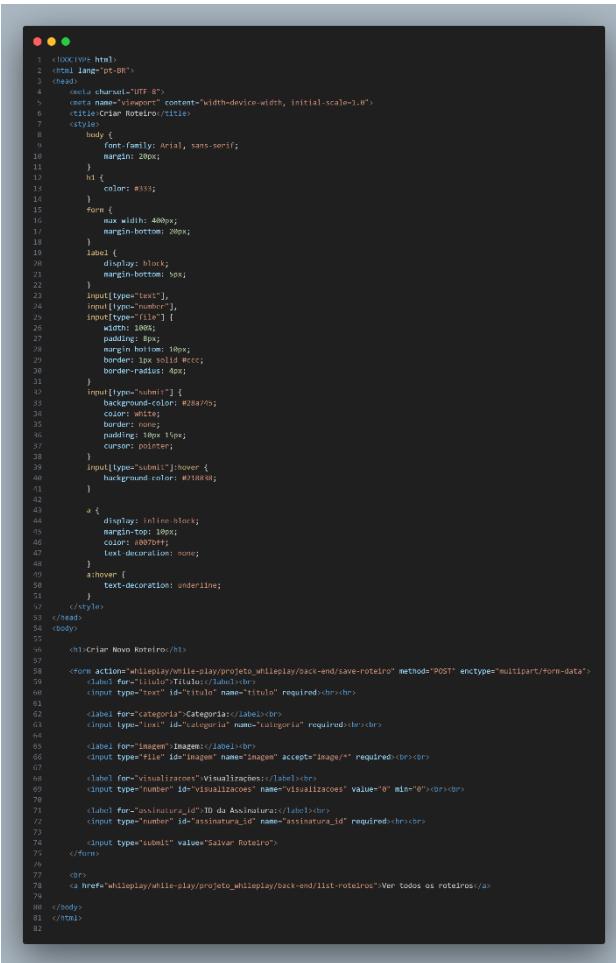
**Pagamento List– Views:** O código apresenta um formulário para atualizar um pagamento existente. Ele exibe os dados atuais do pagamento (como nome e número do cartão, data de vencimento e código de segurança) já preenchidos nos campos, permitindo que o usuário os edite. Há um campo oculto que contém o ID do pagamento para que o sistema saiba qual registro deve ser atualizado.

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Atualizar Roteiro</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8   <style>
9     body {
10       font-family: Arial, sans-serif;
11       margin: 20px;
12     }
13     h1 {
14       color: #333;
15     }
16     form {
17       max-width: 400px;
18       margin-bottom: 20px;
19     }
20     label {
21       display: block;
22       margin-bottom: 5px;
23     }
24     input[type="text"],
25     input[type="number"],
26     input[type="file"] {
27       width: 100px;
28       padding: 8px;
29       margin-bottom: 10px;
30       border: 1px solid #ccc;
31       border-radius: 4px;
32     }
33     input[type="submit"] {
34       background-color: #28a745;
35       color: white;
36       border: none;
37       padding: 10px 15px;
38       cursor: pointer;
39     }
40     input[type="submit"]:hover {
41       background-color: #218838;
42     }
43     a {
44       display: inline-block;
45       margin-top: 10px;
46       color: #007bff;
47       text-decoration: none;
48     }
49     a:hover {
50       text-decoration: underline;
51     }
52   </style>
53 </head>
54 <body>
55
56   <h1>Atualizar Roteiro:</h1>
57
58   <form action="whileplay/while play/projeto_whileplay/hack_end/update_rroteiro" method="POST" enctype="multipart/form-data">
59     <input type="hidden" name="id" value="php echo htmlspecialchars($roteiroInfo['id']); ?&gt;"&gt;
60
61     &lt;label for="titulo"&gt;Título:&lt;/label&gt;&lt;br&gt;
62     &lt;input type="text" id="titulo" name="titulo" value="<?php echo htmlspecialchars($roteiroInfo['titulo']); ?&gt;" required&gt;&lt;br&gt;&lt;br&gt;
63
64     &lt;label for="categoria"&gt;Categoria:&lt;/label&gt;&lt;br&gt;
65     &lt;input type="text" id="categoria" name="categoria" value="<?php echo htmlspecialchars($roteiroInfo['categoria']); ?&gt;" required&gt;&lt;br&gt;&lt;br&gt;
66
67     &lt;label for="imagem_atual"&gt;Imagem atual:&lt;/label&gt;&lt;br&gt;
68     &lt;img src="/Sistema CRUD-Projeto/&lt;?php echo htmlspecialchars($roteiroInfo['caminho_imagem']); ?&gt;" alt="Imagem do roteiro" style="max-width:200px; max-height:150px;"&gt;&lt;br&gt;&lt;br&gt;
69
70     &lt;label for="imagem"&gt;Trocar imagem (opcional):&lt;/label&gt;&lt;br&gt;
71     &lt;input type="file" id="imagem" name="imagem" accept="image/*"&gt;&lt;br&gt;&lt;br&gt;
72
73     &lt;label for="visualizacoes"&gt;Visualizações:&lt;/label&gt;&lt;br&gt;
74     &lt;input type="number" id="visualizacoes" name="visualizacoes" value="<?php echo htmlspecialchars($roteiroInfo['visualizacoes']); ?&gt;" min="0"&gt;&lt;br&gt;&lt;br&gt;
75
76     &lt;label for="assinatura_id"&gt;ID da assinatura:&lt;/label&gt;&lt;br&gt;
77     &lt;input type="number" id="assinatura_id" name="assinatura_id" value="<?php echo htmlspecialchars($roteiroInfo['assinatura_id']); ?&gt;" required&gt;&lt;br&gt;&lt;br&gt;
78
79     &lt;input type="submit" value="Atualizar Roteiro"&gt;
80   &lt;/form&gt;
81
82   &lt;br&gt;
83   &lt;a href="whileplay/while play/projeto_whileplay/back_end/list_rroteiros"&gt;Voltar para a lista de roteiros&lt;/a&gt;
84
85 &lt;/body&gt;
86 &lt;/html&gt;
87
</pre

```

**Update Roteiro – Views:** Ele exibe um formulário já preenchido com os dados do roteiro selecionado (título, categoria, imagem, visualizações e ID da assinatura), permitindo que o usuário edite as informações e envie alterações via POST para o backend. Também é possível trocar a imagem do roteiro.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="utf-8">
5   <meta content="width=device-width, initial-scale=1.0" name="viewport">
6   <title>Criar Roteiro</title>
7 </head>
8 <body>
9   <h1>Criar Novo Roteiro</h1>
10  <form action="whileplay/while-play/projeto_whileplay/back-end/save-roteiro" method="POST" enctype="multipart/form-data">
11    <input type="text" id="titulo" name="titulo" required><br>
12    <label for="categoria">Categoria:</label><br>
13    <input type="text" id="categoria" name="categoria" required><br>
14    <label for="image">Imagem:</label><br>
15    <input type="file" id="image" name="image" accept="image/*" required><br>
16    <label for="visualizacoes">Visualizações:</label><br>
17    <input type="number" id="visualizacoes" name="visualizacoes" value="0" min="0"><br>
18    <label for="assinatura_id">ID da Assinatura:</label><br>
19    <input type="number" id="assinatura_id" name="assinatura_id" required><br>
20    <input type="submit" value="Salvar Roteiro">
21  </form>
22  <br>
23  <a href="whileplay/while-play/projeto_whileplay/back-end/list-roteiros">Ver todos os roteiros</a>
24
25 </body>
26 </html>

```

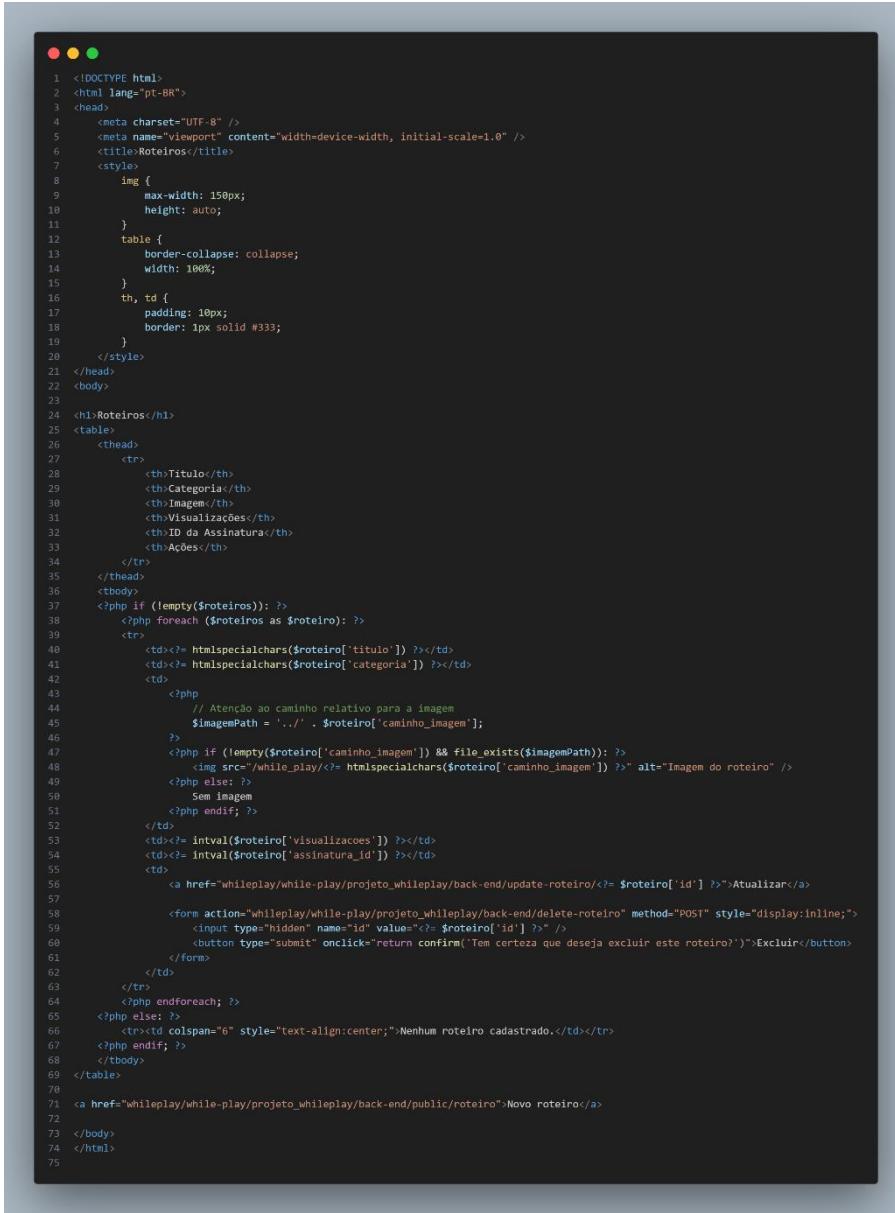
**Update Assinatura – Views:** O código é usado para coletar dados do roteiro como título, categoria, imagem, visualizações e id de assinatura. Todos os campos são obrigatórios e ao ser enviado, o formulário envia os dados via POST para uma rota.

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <title>Atualizar Pagamento</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1" />
7   <style>
8     form {
9       max-width: 400px;
10      margin: 30px auto;
11      padding: 20px;
12      border: 1px solid #ddd;
13      border-radius: 8px;
14      font-family: Arial, sans-serif;
15    }
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21    input[type="text"],
22    input[type="date"],
23    input[type="number"] {
24      width: 100%;
25      padding: 8px;
26      margin-top: 5px;
27      box-sizing: border-box;
28    }
29    input[type="submit"] {
30      margin-top: 20px;
31      background-color: #4CAF50;
32      color: white;
33      border: none;
34      padding: 12px;
35      cursor: pointer;
36      border-radius: 5px;
37      width: 100%;
38      font-size: 16px;
39    }
40    @media {
41      display: block;
42      margin: 15px auto;
43      text-align: center;
44      color: #555;
45      text-decoration: none;
46    }
47  </style>
48 </head>
49 <body>
50
51 <h2 style="text-align:center;">Atualizar Pagamento</h2>
52
53 <form action="whileplay/while-play/projeto_whileplay/back-end/update-pagamento" method="POST">
54
55   <label for="id_pagamento">ID do Pagamento:</label>
56   <input type="hidden" name="id_pagamento" value="php echo htmlspecialchars($pagamentoInfo['id_pagamento']); ?">
57
58   <label for="nome_do_cartao">Nome no Cartão:</label>
59   <input type="text" id="nome_do_cartao" name="nome_do_cartao" value="php echo htmlspecialchars($pagamentoInfo['nome_do_cartao']); ?" required>
60
61   <label for="numero_do_cartao">Número do Cartão:</label>
62   <input type="text" id="numero_do_cartao" name="numero_do_cartao" value="php echo htmlspecialchars($pagamentoInfo['numero_do_cartao']); ?" required pattern="\d{13,19}" title="Insira um número válido do cartão">
63
64   <label for="data_de_vencimento">Data de Vencimento:</label>
65   <input type="month" id="data_de_vencimento" name="data_de_vencimento" value="php echo htmlspecialchars($pagamentoInfo['data_de_vencimento']); ?" required>
66
67   <label for="codigo">Código:</label>
68   <input type="text" id="codigo" name="codigo" value="php echo htmlspecialchars($pagamentoInfo['codigo']); ?" required pattern="\d{3,4}" title="Código de 3 ou 4 dígitos">
69
70   <input type="submit" value="Atualizar Pagamento">
71 </form>
72
73 <a href="whileplay/while-play/projeto_whileplay/back-end/list-pagamentos">Voltar para a lista de pagamentos</a>
74
75 </body>
76 </html>
77

```

**Update Pagamento – Views:** O código é usado para coletar dados do roteiro como título, categoria, imagem, visualizações e id de assinatura. Todos os campos são obrigatórios e ao ser enviado, o formulário envia os dados via POST para uma rota.



```

1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Roteiros</title>
7      <style>
8          img {
9              max-width: 150px;
10             height: auto;
11         }
12         table {
13             border-collapse: collapse;
14             width: 100%;
15         }
16         th, td {
17             padding: 10px;
18             border: 1px solid #333;
19         }
20     </style>
21 </head>
22 <body>
23
24 <h1>Roteiros</h1>
25 <table>
26     <thead>
27         <tr>
28             <th>Título</th>
29             <th>Categoria</th>
30             <th>Imagem</th>
31             <th>Visualizações</th>
32             <th>ID da Assinatura</th>
33             <th>Ações</th>
34         </tr>
35     </thead>
36     <tbody>
37     <?php if (!empty($roteiros)): ?>
38     <?php foreach ($roteiros as $roteiro): ?>
39         <tr>
40             <td><?= htmlspecialchars($roteiro['titulo']) ?></td>
41             <td><?= htmlspecialchars($roteiro['categoria']) ?></td>
42             <td>
43                 <?php
44                     // Atenção ao caminho relativo para a imagem
45                     $imagePath = '../' . $roteiro['caminho_imagem'];
46                 ?>
47                 <?php if (!empty($roteiro['caminho_imagem']) && file_exists($imagePath)): ?>
48                     
49                 <?php else: ?>
50                     Sem imagem
51                 <?php endif; ?>
52             </td>
53             <td><?= intval($roteiro['visualizacoes']) ?></td>
54             <td><?= intval($roteiro['assinatura_id']) ?></td>
55             <td>
56                 <a href="whileplay/while-play/projeto_whileplay/back-end/update-roteiro/<?= $roteiro['id'] ?>">Atualizar</a>
57
58                 <form action="whileplay/while-play/projeto_whileplay/back-end/delete-roteiro" method="POST" style="display:inline;">
59                     <input type="hidden" name="id" value=<?= $roteiro['id'] ?> />
60                     <button type="submit" onclick="return confirm('Tem certeza que deseja excluir este roteiro?')>Excluir</button>
61                 </form>
62             </td>
63         </tr>
64     <?php endforeach; ?>
65     <?php else: ?>
66         <tr><td colspan="6" style="text-align:center;">Nenhum roteiro cadastrado.</td></tr>
67     <?php endif; ?>
68     </tbody>
69 </table>
70
71 <a href="whileplay/while-play/projeto_whileplay/back-end/public/roteiro">Novo roteiro</a>
72
73 </body>
74 </html>

```

**Roteiro List – Views:** O principal objetivo desse código é percorrer a lista \$roteiros que vem do backend e criar uma linha na tabela para cada item, preenchendo os dados. Há também botões para o usuário atualizar ou excluir cada roteiro, sempre pedindo confirmação antes de excluir para evitar erros.

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Atualizar Assinatura</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8   <style>
9     body {
10       font-family: Arial, sans-serif;
11       margin: 20px;
12     }
13     h1 {
14       color: #333;
15     }
16     form {
17       max-width: 400px;
18       margin-bottom: 20px;
19     }
20     label {
21       display: block;
22       margin-bottom: 5px;
23     }
24     input[type="text"],
25     input[type="date"] {
26       width: 100%;
27       padding: 8px;
28       margin-bottom: 10px;
29       border: 1px solid #ccc;
30       border-radius: 4px;
31     }
32     input[type="submit"] {
33       background-color: #28a745;
34       color: white;
35       border: none;
36       padding: 10px 15px;
37       cursor: pointer;
38     }
39     input[type="submit"]:hover {
40       background-color: #21a883;
41     }
42     a {
43       display: inline-block;
44       margin-top: 10px;
45       color: #007bff;
46       text-decoration: none;
47     }
48     a:hover {
49       text-decoration: underline;
50     }
51   </style>
52 </head>
53 <body>
54
55 <h1>Atualizar Assinatura</h1>
56
57 <form action="whileplay/while-play/projeto_whileplay/back-end/update-assinatura" method="POST">
58   <input type="hidden" name="id" value=<?= htmlspecialchars($assinaturaInfo['id']) ?>>
59
60   <label for="nome">Nome:</label>
61   <input type="text" id="nome" name="nome" value=<?= htmlspecialchars($assinaturaInfo['nome']) ?>> required<br><br>
62
63   <label for="cidade">Cidade:</label>
64   <input type="text" id="cidade" name="cidade" value=<?= htmlspecialchars($assinaturaInfo['cidade']) ?>> required<br><br>
65
66   <label for="endereco">Endereço:</label>
67   <input type="text" id="endereco" name="endereco" value=<?= htmlspecialchars($assinaturaInfo['endereco']) ?>> required<br><br>
68
69   <label for="cep">CEP:</label>
70   <input type="text" id="cep" name="cep" value=<?= htmlspecialchars($assinaturaInfo['cep']) ?>> pattern="^d{5}-?d{3}" title="Formato: 00000-000"<br><br>
71
72   <label for="cpf">CPF:</label>
73   <input type="text" id="cpf" name="cpf" value=<?= htmlspecialchars($assinaturaInfo['cpf']) ?>> required pattern="^d{3}\.\d{3}.\d{3}\.\d{2}" title="Formato: 000.000.000-00"<br><br>
74
75   <label for="data_assinatura">Data de Assinatura:</label>
76   <input type="date" id="data_assinatura" name="data_assinatura" value=<?= htmlspecialchars($assinaturaInfo['data_assinatura']) ?>> required<br><br>
77
78   <input type="submit" value="Atualizar">
79 </form>
80
81 <a href="whileplay/while-play/projeto_whileplay/back-end/list-assinaturas">Voltar para a lista</a>
82
83 </body>
84 </html>
85

```

**Roteiro Update – Views:** exibe um formulário para editar uma assinatura existente no sistema. Ele pré-preenche os campos com os dados atuais da assinatura (como nome, cidade, CPF, etc.), usando a variável \$assinaturaInfo, e envia as alterações via POST para o backend (/update-assinatura) para atualizar as informações. Também inclui um campo oculto com o ID da assinatura e um link para voltar à lista completa. A página funciona como o frontend da edição, enquanto o backend cuida da atualização real.

```

1 <?php
2
3 class Perfil {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($nome_completo, $username, $email, $senha, $biografia, $foto_url, $data_criacao) {
12        $stmt = $this->pdo->prepare(
13            "INSERT INTO perfil (nome_completo, username, email, senha, biografia, foto_url, data_criacao)
14            VALUES (?, ?, ?, ?, ?, ?, ?)
15        ");
16        $stmt->execute([$nome_completo, $username, $email, $senha, $biografia, $foto_url, $data_criacao]);
17    }
18
19    public function getAll() {
20        $stmt = $this->pdo->query("SELECT * FROM perfil");
21        return $stmt->fetchAll(PDO::FETCH_ASSOC);
22    }
23
24    public function getById($id) {
25        $stmt = $this->pdo->prepare("SELECT * FROM perfil WHERE id = ?");
26        $stmt->execute([$id]);
27        return $stmt->fetch(PDO::FETCH_ASSOC);
28    }
29
30    public function update($id, $nome_completo, $username, $email, $senha, $biografia, $foto_url, $data_criacao) {
31        $stmt = $this->pdo->prepare(
32            "UPDATE perfil
33            SET nome_completo = ?, username = ?, email = ?, senha = ?, biografia, foto_url = ?, data_criacao = ?
34            WHERE id = ?
35        ");
36        $stmt->execute([$nome_completo, $username, $email, $senha, $biografia, $foto_url, $data_criacao, $id]);
37    }
38
39    public function deleteById($id) {
40        $stmt = $this->pdo->prepare("DELETE FROM perfil WHERE id = ?");
41        $stmt->execute([$id]);
42    }
43 }

```

**Models – Perfil:** A classe Perfil em PHP realiza operações básicas de banco de dados (CRUD) usando PDO com segurança contra SQL Injection. Ela permite criar, buscar, atualizar e deletar perfis de usuário. Apesar de ser funcional e bem estruturada, o código pode ser melhorado com criptografia de senhas (password\_hash), tratamento de exceções (try/catch) e validação dos dados antes do uso. É um bom exemplo inicial de manipulação segura de dados com PDO em PHP.



```

1 <?php
2
3 class Personagem {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($id_sobre, $mais_bem_avaliados, $lançados_recentemente, $caminho_imagem) {
12        $stmt = $this->pdo->prepare(
13            "INSERT INTO personagens (id_sobre, mais_bem_avaliados$, lançados_recentemente, caminho_imagem)
14            VALUES (?, ?, ?, ?)
15        ");
16        $stmt->execute([$id_sobre, $mais_bem_avaliados, $lançados_recentemente, $caminho_imagem]);
17    }
18
19    public function getAll() {
20        $stmt = $this->pdo->query("SELECT * FROM personagens");
21        return $stmt->fetchAll(PDO::FETCH_ASSOC);
22    }
23
24    public function getById($id) {
25        $stmt = $this->pdo->prepare("SELECT * FROM personagens WHERE id = ?");
26        $stmt->execute([$id]);
27        return $stmt->fetch(PDO::FETCH_ASSOC);
28    }
29
30    public function update( $id_sobre, $mais_bem_avaliados, $lançados_recentemente, $caminho_imagem) {
31        $stmt = $this->pdo->prepare(
32            "UPDATE personagens
33            SET id_sobre = ?, mais_bem_avaliados = ?, lançados_recentemente = ?, caminho_imagem = ?
34            WHERE id = ?
35        ");
36        $stmt->execute([$id_sobre, $mais_bem_avaliados, $lançados_recentemente, $caminho_imagem, $id]);
37    }
38
39    public function deleteById($id) {
40        $stmt = $this->pdo->prepare("DELETE FROM personagens WHERE id = ?");
41        $stmt->execute([$id]);
42    }
43 }

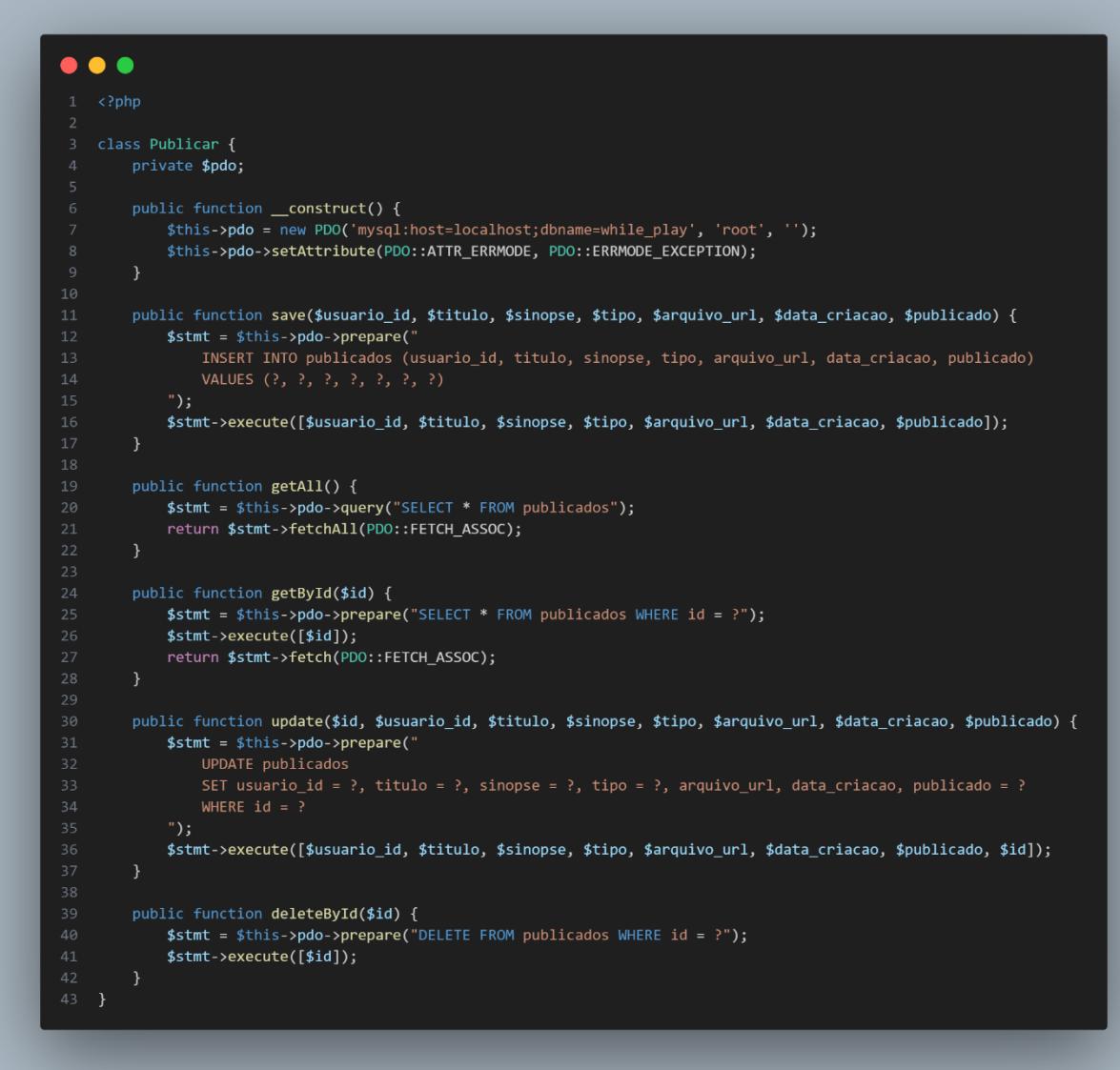
```

**Models – Personagens:** O código apresentado define a classe Personagem em PHP, responsável por executar operações CRUD (Create, Read, Update, Delete) em uma tabela chamada personagens, utilizando a extensão PDO para interação com o banco de dados MySQL. A conexão é estabelecida no construtor da classe, com o tratamento de erros configurado para lançar exceções, o que facilita a identificação de falhas.

Os métodos implementados permitem inserir novos registros (save), buscar todos os personagens (getAll), buscar por ID (getById), atualizar um registro (update) e excluir um personagem pelo ID (deleteById). O uso de **prepared statements** com execute() garante maior segurança contra ataques de SQL Injection.

No entanto, há um pequeno erro de digitação na linha de inserção do método save: a variável \$mais\_bem\_avaliados\$ contém um símbolo \$ a mais, o que geraria um erro

de sintaxe. Corrigindo isso, a classe apresenta uma estrutura funcional, organizada e segura para operações básicas com banco de dados.



```

1 <?php
2
3 class Publicar {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($usuario_id, $titulo, $sinopse, $tipo, $arquivo_url, $data_criacao, $publicado) {
12        $stmt = $this->pdo->prepare(
13            "INSERT INTO publicados (usuario_id, titulo, sinopse, tipo, arquivo_url, data_criacao, publicado)
14            VALUES (?, ?, ?, ?, ?, ?, ?)"
15        );
16        $stmt->execute([$usuario_id, $titulo, $sinopse, $tipo, $arquivo_url, $data_criacao, $publicado]);
17    }
18
19    public function getAll() {
20        $stmt = $this->pdo->query("SELECT * FROM publicados");
21        return $stmt->fetchAll(PDO::FETCH_ASSOC);
22    }
23
24    public function getById($id) {
25        $stmt = $this->pdo->prepare("SELECT * FROM publicados WHERE id = ?");
26        $stmt->execute([$id]);
27        return $stmt->fetch(PDO::FETCH_ASSOC);
28    }
29
30    public function update($id, $usuario_id, $titulo, $sinopse, $tipo, $arquivo_url, $data_criacao, $publicado) {
31        $stmt = $this->pdo->prepare(
32            "UPDATE publicados
33            SET usuario_id = ?, titulo = ?, sinopse = ?, tipo = ?, arquivo_url, data_criacao, publicado = ?
34            WHERE id = ?"
35        );
36        $stmt->execute([$usuario_id, $titulo, $sinopse, $tipo, $arquivo_url, $data_criacao, $publicado, $id]);
37    }
38
39    public function deleteById($id) {
40        $stmt = $this->pdo->prepare("DELETE FROM publicados WHERE id = ?");
41        $stmt->execute([$id]);
42    }
43 }

```

**Models – Publicar:** O código define a classe publicar, que gerencia publicações em um sistema PHP conectado a um banco de dados MySQL usando PDO. A classe implementa as operações CRUD (Create, Read, Update, Delete) na tabela publicados. A conexão com o banco é estabelecida no construtor, com o tratamento de erros configurado para lançar exceções, o que é importante para identificar problemas rapidamente.

O método `save()` insere uma nova publicação com dados como `usuario_id`, `título`, `sinopse`, `tipo`, `arquivo_url`, `data_criacao` e `publicado`. O uso de **prepared statements** com `execute()` garante segurança contra SQL Injection. Os métodos `getAll()` e `getById()` permitem buscar todos os registros ou um específico pelo ID. Já o método `update()` atualiza os dados de uma publicação existente, e `deleteById()` realiza a exclusão com base no ID.

A estrutura é organizada e funcional, seguindo boas práticas básicas. Como sugestão de melhoria, seria interessante adicionar **validação dos dados, tratamento de exceções com try/catch**, e eventualmente separar a lógica de conexão em uma camada própria para facilitar testes e manutenção.



```

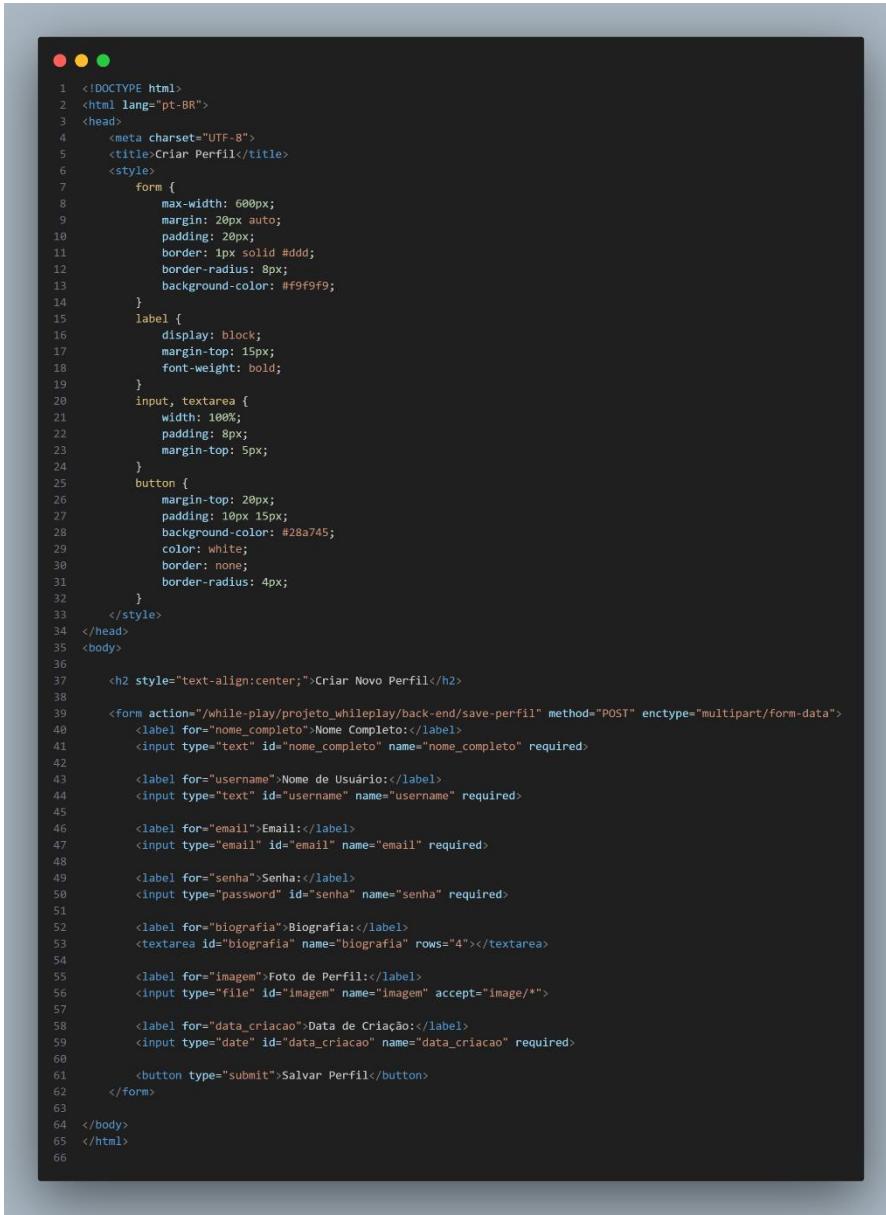
1 <?php
2
3 class Suporte {
4     private $pdo;
5
6     public function __construct() {
7         $this->pdo = new PDO('mysql:host=localhost;dbname=while_play', 'root', '');
8         $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
9     }
10
11    public function save($usuario_id, $mensagem, $data_envio) {
12        $stmt = $this->pdo->prepare("INSERT INTO suportes (usuario_id, mensagem, data_envio) VALUES (?, ?, ?)");
13        $stmt->execute([$usuario_id, $mensagem, $data_envio]);
14    }
15
16    public function getAll() {
17        $stmt = $this->pdo->query("SELECT * FROM suportes");
18        return $stmt->fetchAll(PDO::FETCH_ASSOC);
19    }
20
21    public function deleteByTitle($usuario_id) {
22        $stmt = $this->pdo->prepare("DELETE FROM suportes WHERE usuario_id = ?");
23        $stmt->execute([$usuario_id]);
24    }
25
26    public function getById($id) {
27        $stmt = $this->pdo->prepare("SELECT * FROM suportes WHERE id = ?");
28        $stmt->execute([$id]);
29        return $stmt->fetch(PDO::FETCH_ASSOC);
30    }
31
32    public function update($id, $usuario_id, $mensagem, $data_envio) {
33        $stmt = $this->pdo->prepare("UPDATE suportes SET usuario_id = ?, mensagem = ?, data_envio = ? WHERE id = ?");
34        $stmt->execute([$usuario_id, $mensagem, $data_envio, $id]);
35    }
36 }
37

```

**Models – Suporte:** A classe Suporte em PHP é responsável por gerenciar registros de mensagens de suporte no banco de dados `while_play`, utilizando PDO para conexão e execução de comandos SQL. Ela implementa as funções básicas de um CRUD, permitindo salvar, buscar, atualizar e excluir mensagens de suporte associadas a usuários.

O método `save()` insere uma nova mensagem na tabela `suportes`, recebendo o `usuario_id`, a `mensagem` e a `data_envio`. O método `getAll()` retorna todas as mensagens, enquanto `getById()` busca uma mensagem específica pelo `id`. O método `update()` atualiza os dados de uma mensagem existente, e `deleteByTitle()` (nome confuso, pois na verdade deleta por `usuario_id`) remove todas as mensagens associadas a um usuário.

O código é bem estruturado e usa **prepared statements**, o que garante proteção contra SQL Injection. No entanto, seria interessante melhorar o nome do método `deleteByTitle()` para algo como `deleteById()` para refletir com clareza sua funcionalidade. Além disso, incluir **tratamento de exceções (try/catch)** e **validação de dados** tornaria o código mais robusto.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Criar Perfil</title>
6   <style>
7     form {
8       max-width: 600px;
9       margin: 20px auto;
10      padding: 20px;
11      border: 1px solid #ddd;
12      border-radius: 8px;
13      background-color: #f9f9f9;
14    }
15    label {
16      display: block;
17      margin-top: 15px;
18      font-weight: bold;
19    }
20    input, textarea {
21      width: 100%;
22      padding: 8px;
23      margin-top: 5px;
24    }
25    button {
26      margin-top: 20px;
27      padding: 10px 15px;
28      background-color: #28a745;
29      color: white;
30      border: none;
31      border-radius: 4px;
32    }
33  </style>
34 </head>
35 <body>
36
37  <h2 style="text-align:center;">Criar Novo Perfil</h2>
38
39  <form action="/while-play/projeto_whileplay/back-end/save-perfil" method="POST" enctype="multipart/form-data">
40    <label for="nome_completo">Nome Completo:</label>
41    <input type="text" id="nome_completo" name="nome_completo" required>
42
43    <label for="username">Nome de Usuário:</label>
44    <input type="text" id="username" name="username" required>
45
46    <label for="email">Email:</label>
47    <input type="email" id="email" name="email" required>
48
49    <label for="senha">Senha:</label>
50    <input type="password" id="senha" name="senha" required>
51
52    <label for="biografia">Biografia:</label>
53    <textarea id="biografia" name="biografia" rows="4"></textarea>
54
55    <label for="imagem">Foto de Perfil:</label>
56    <input type="file" id="imagem" name="imagem" accept="image/*">
57
58    <label for="data_criacao">Data de Criação:</label>
59    <input type="date" id="data_criacao" name="data_criacao" required>
60
61    <button type="submit">Salvar Perfil</button>
62  </form>
63
64 </body>
65 </html>
66

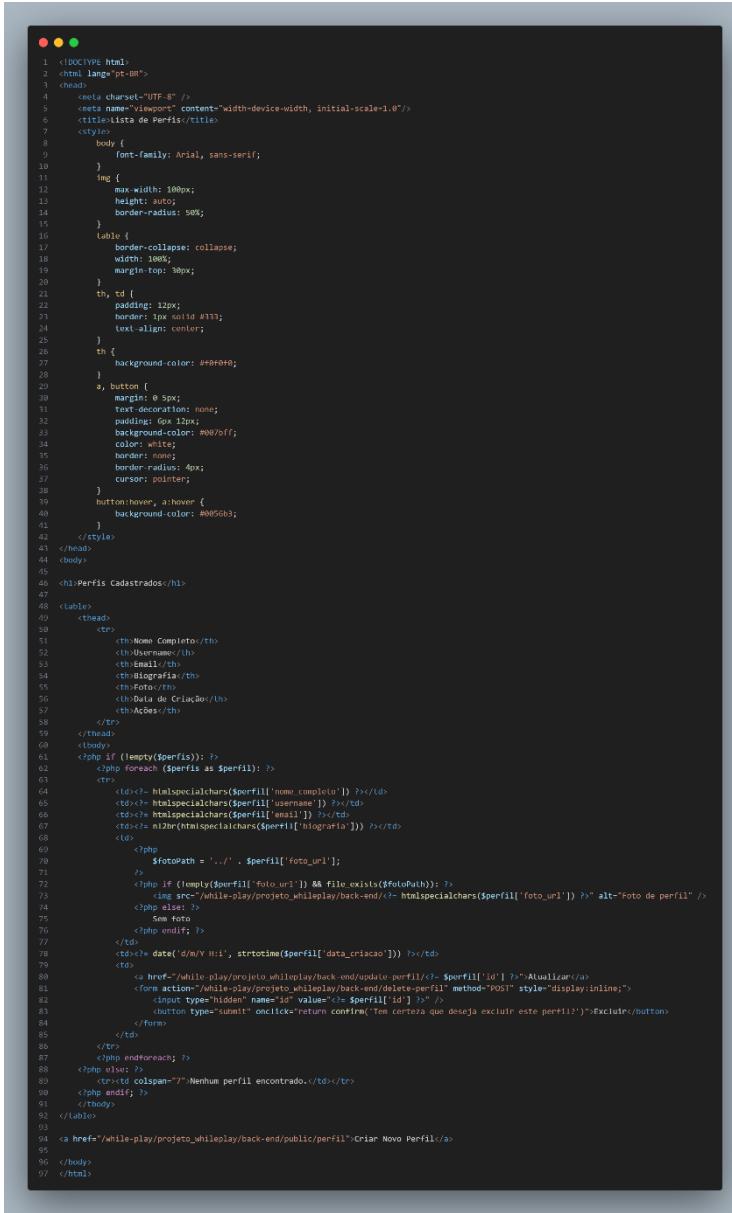
```

**Perfil – Form:** Este código HTML representa um formulário para criação de perfis de usuário. Ele coleta dados como nome completo, nome de usuário, email, senha, biografia, foto de perfil e data de criação. O formulário utiliza o método POST e o tipo multipart/form-data para permitir o envio de arquivos (imagem de perfil).

Os campos possuem validação básica via HTML com o atributo `required`, garantindo que o usuário preencha todos os dados obrigatórios. O estilo CSS interno proporciona

um layout limpo e centralizado, com foco em usabilidade e visual agradável. O botão "Salvar Perfil" finaliza o envio das informações para o back-end, no arquivo `save-perfil.php`.

Esse formulário é ideal para sistemas de cadastro de usuários e pode ser facilmente integrado com scripts PHP para persistência em banco de dados.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <title>Lista de Perfis</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10    }
11    img {
12      max-width: 100px;
13      height: auto;
14      border-radius: 50%;
15    }
16    table {
17      border-collapse: collapse;
18      width: 100%;
19      margin-top: 30px;
20    }
21    th, td {
22      padding: 10px;
23      border: 1px solid #333;
24      text-align: center;
25    }
26    th {
27      background-color: #e6f2ff;
28    }
29    a, button {
30      margin: 5px;
31      text-decoration: none;
32      padding: 5px 15px;
33      background-color: #007bff;
34      color: white;
35      border: none;
36      border-radius: 4px;
37      cursor: pointer;
38    }
39    button:hover, a:hover {
40      background-color: #006933;
41    }
42  </style>
43 </head>
44 <body>
45   <h1>Perfis Cadastrados:</h1>
46   <table>
47     <thead>
48       <tr>
49         <th>Nome Completo</th>
50         <th>Username</th>
51         <th>Email</th>
52         <th>Foto</th>
53         <th>Data de Criação</th>
54         <th>Ações</th>
55       </tr>
56     </thead>
57     <tbody>
58       <tr>
59         <td></td>
60         <td></td>
61         <td></td>
62         <td></td>
63         <td></td>
64         <td></td>
65       </tr>
66     </tbody>
67   </table>
68   <?php if (!empty($perfis)): ?>
69   <?php foreach ($perfis as $perfil): ?>
70   <tr>
71     <td><?php echo htmlspecialchars($perfil['nome_completo']) ?></td>
72     <td><?php echo htmlspecialchars($perfil['username']) ?></td>
73     <td><?php echo htmlspecialchars($perfil['email']) ?></td>
74     <td><?php echo htmlspecialchars($perfil['foto_url']) ?></td>
75     <td><?php echo date('d/m/Y H:i', strtotime($perfil['data_criacao'])) ?></td>
76     <td>
77       <a href="/while-play/projeto/whileplay/back-end/update-perfil/<- $perfil['id'] ?>">Atualizar</a>
78       <form action="/while-play/projeto/whileplay/back-end/delete-perfil" method="POST" style="display:inline;">
79         <input type="Hidden" name="id" value="<?php echo $perfil['id'] ?>" />
80         <button type="Submit" onclick="return confirm('Tem certeza que deseja excluir este perfil?')">>Excluir</button>
81       </form>
82     </td>
83   </tr>
84   <?php endforeach; ?>
85   <tr><td colspan="7">Nenhum perfil encontrado.</td></tr>
86   <?php endif; ?>
87   </tbody>
88   </table>
89   <a href="/while-play/projeto/whileplay/back-end/public/perfil">Criar Novo Perfil</a>
90 </body>
91 </html>
92 </!doctype>
93 <a href="#">while-play/projeto_whileplay/back-end/public/perfil</a>
94 </body>
95 </html>
96 </html>
97 </html>

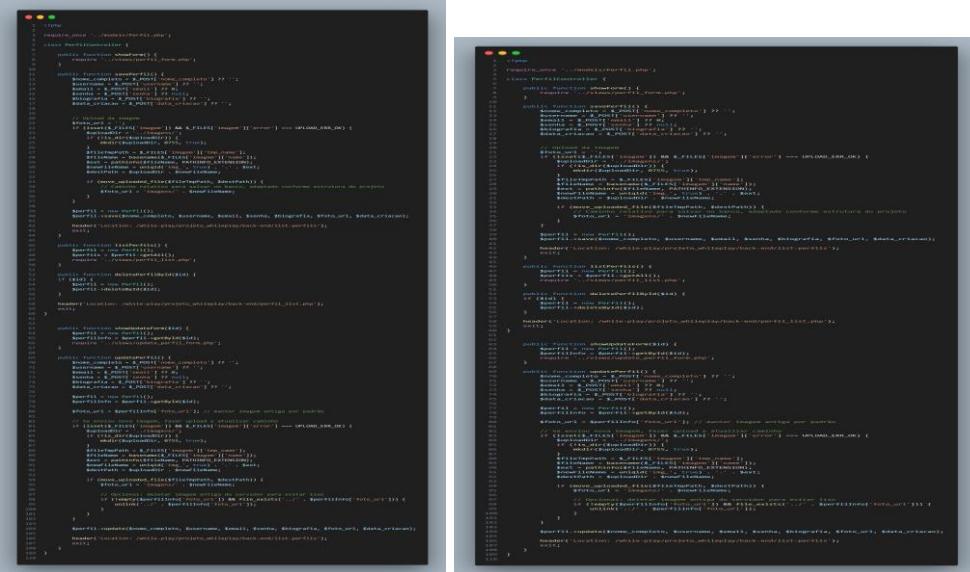
```

**Perfil – List:** Este código HTML/PHP exibe uma **tabela de listagem de perfis cadastrados**. O layout é responsivo e utiliza uma estilização simples para melhorar a legibilidade da tabela e os botões de ação.

Cada linha da tabela representa um perfil de usuário, exibindo: nome completo, nome de usuário, email, biografia, foto de perfil (se disponível), data de criação e opções de ações (“Atualizar” e “Excluir”).

O script usa PHP para iterar sobre um array \$perfis, protegendo os dados com htmlspecialchars() e nl2br() para evitar XSS e preservar formatações de texto. Também verifica se a imagem existe fisicamente antes de exibi-la. A exclusão é feita via formulário POST, com botão de confirmação JavaScript. Ao fim da página, há um link que redireciona para o formulário de criação de novos perfis.

Esse código é ideal para o front-end de uma área administrativa simples que gerencia cadastros de usuários.



```

class PerfilController extends Controller
{
    public function index()
    {
        $perfis = $this->Perfil->listar();
        return view('perfil.index', [
            'perfis' => $perfis
        ]);
    }

    public function novo()
    {
        $dados = [
            'nome_completo' => null,
            'username' => null,
            'email' => null,
            'biografia' => null,
            'data_criacao' => null,
            'foto_perfil' => null
        ];
        return view('perfil.create', [
            'dados' => $dados
        ]);
    }

    public function store()
    {
        $dados = [
            'nome_completo' => $this->request('nome_completo'),
            'username' => $this->request('username'),
            'email' => $this->request('email'),
            'biografia' => $this->request('biografia'),
            'data_criacao' => date('Y-m-d H:i:s'),
            'foto_perfil' => $this->request('foto_perfil')
        ];
        $this->Perfil->criar($dados);
        return redirect('/perfil');
    }

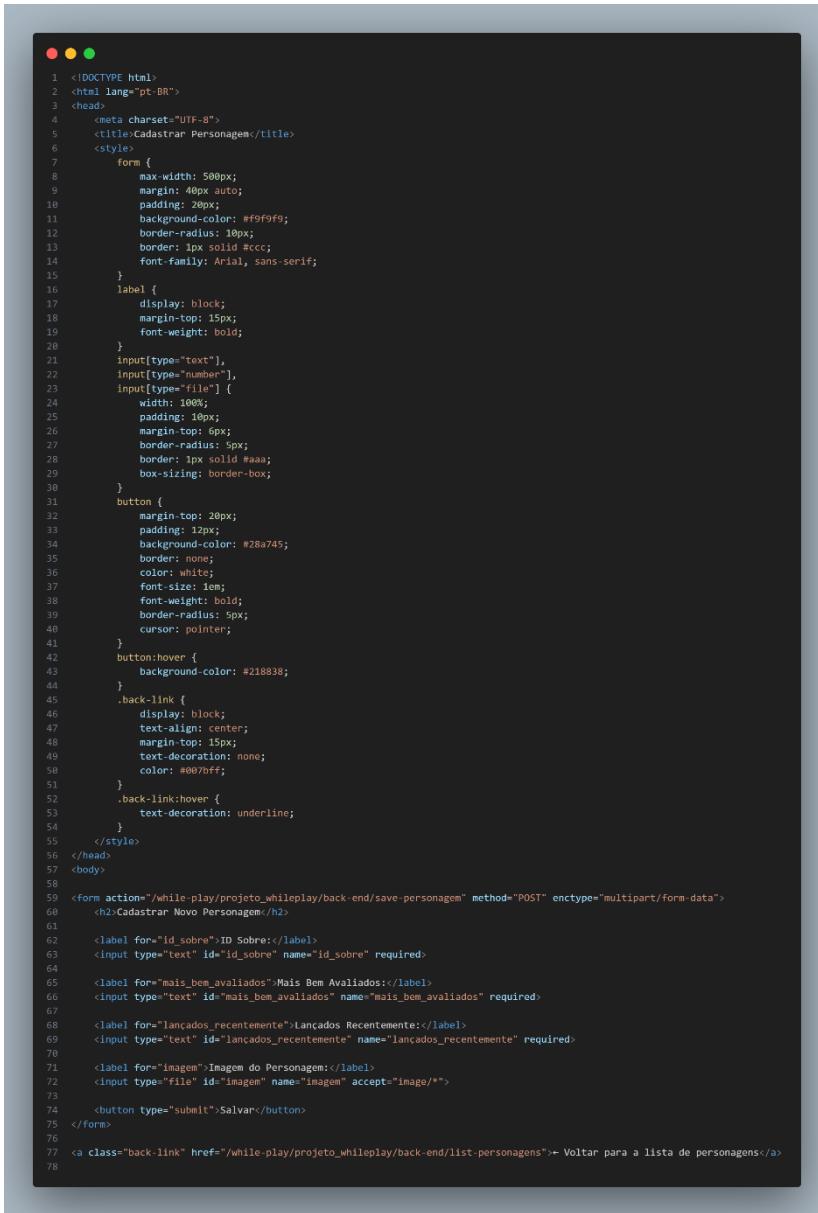
    public function editar($id)
    {
        $perfil = $this->Perfil->find($id);
        if ($perfil) {
            return view('perfil.edit', [
                'perfil' => $perfil
            ]);
        } else {
            return redirect('/perfil');
        }
    }

    public function update($id)
    {
        $dados = [
            'nome_completo' => $this->request('nome_completo'),
            'username' => $this->request('username'),
            'email' => $this->request('email'),
            'biografia' => $this->request('biografia'),
            'data_criacao' => $this->request('data_criacao'),
            'foto_perfil' => $this->request('foto_perfil')
        ];
        $perfil = $this->Perfil->find($id);
        if ($perfil) {
            $perfil->atualizar($dados);
            return redirect('/perfil');
        } else {
            return redirect('/perfil');
        }
    }

    public function destroy($id)
    {
        $perfil = $this->Perfil->find($id);
        if ($perfil) {
            $foto_perfil = $perfil->foto_perfil;
            if (file_exists($foto_perfil)) {
                unlink($foto_perfil);
            }
            $perfil->deletar();
            return redirect('/perfil');
        } else {
            return redirect('/perfil');
        }
    }
}

```

**Perfil – Controller:** Esse código PHP representa um controlador chamado PerfilController, responsável por gerenciar perfis de usuários no sistema. Ele permite criar, listar, atualizar e excluir perfis, além de realizar o upload de imagens de perfil. Na criação e edição, os dados são recebidos via formulário (POST), incluindo informações como nome, email, senha, biografia e foto. O código trata o envio de imagens, gerando nomes únicos para evitar conflitos e armazenando os arquivos em uma pasta específica no servidor. Na atualização, se uma nova imagem for enviada, ele substitui a antiga e remove o arquivo anterior do servidor para evitar arquivos desnecessários. Além disso, após cada ação (criar, editar ou excluir), o sistema redireciona o usuário para a página de listagem de perfis, mantendo o fluxo organizado e funcional.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Cadastrar Personagem</title>
6   <style>
7     form {
8       max-width: 500px;
9       margin: 40px auto;
10      padding: 20px;
11      background-color: #f9f9f9;
12      border-radius: 10px;
13      border: 1px solid #ccc;
14      font-family: Arial, sans-serif;
15    }
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21    input[type="text"],
22    input[type="number"],
23    input[type="file"] {
24      width: 100%;
25      padding: 10px;
26      margin-top: 6px;
27      border-radius: 5px;
28      border: 1px solid #aaa;
29      box-sizing: border-box;
30    }
31    button {
32      margin-top: 20px;
33      padding: 12px;
34      background-color: #28a745;
35      border: none;
36      color: white;
37      font-size: 1em;
38      font-weight: bold;
39      border-radius: 5px;
40      cursor: pointer;
41    }
42    button:hover {
43      background-color: #218838;
44    }
45    .back-link {
46      display: block;
47      text-align: center;
48      margin-top: 15px;
49      text-decoration: none;
50      color: #007bff;
51    }
52    .back-link:hover {
53      text-decoration: underline;
54    }
55  </style>
56 </head>
57 <body>
58 <form action="/while-play/projeto_whileplay/back-end/save-personagem" method="POST" enctype="multipart/form-data">
59   <h2>Cadastrar Novo Personagem</h2>
60
61   <label for="id_sobre">ID Sobre:</label>
62   <input type="text" id="id_sobre" name="id_sobre" required>
63
64   <label for="mais_bem_avaliados">Mais Bem Avaliados:</label>
65   <input type="text" id="mais_bem_avaliados" name="mais_bem_avaliados" required>
66
67   <label for="lançados_recentemente">Lançados Recentemente:</label>
68   <input type="text" id="lançados_recentemente" name="lançados_recentemente" required>
69
70   <label for="imagem">Imagem do Personagem:</label>
71   <input type="file" id="imagem" name="imagem" accept="image/*">
72
73   <button type="submit">Salvar</button>
74 </form>
75
76 <a class="back-link" href="/while-play/projeto_whileplay/back-end/list-personagens">+ Voltar para a lista de personagens</a>
77

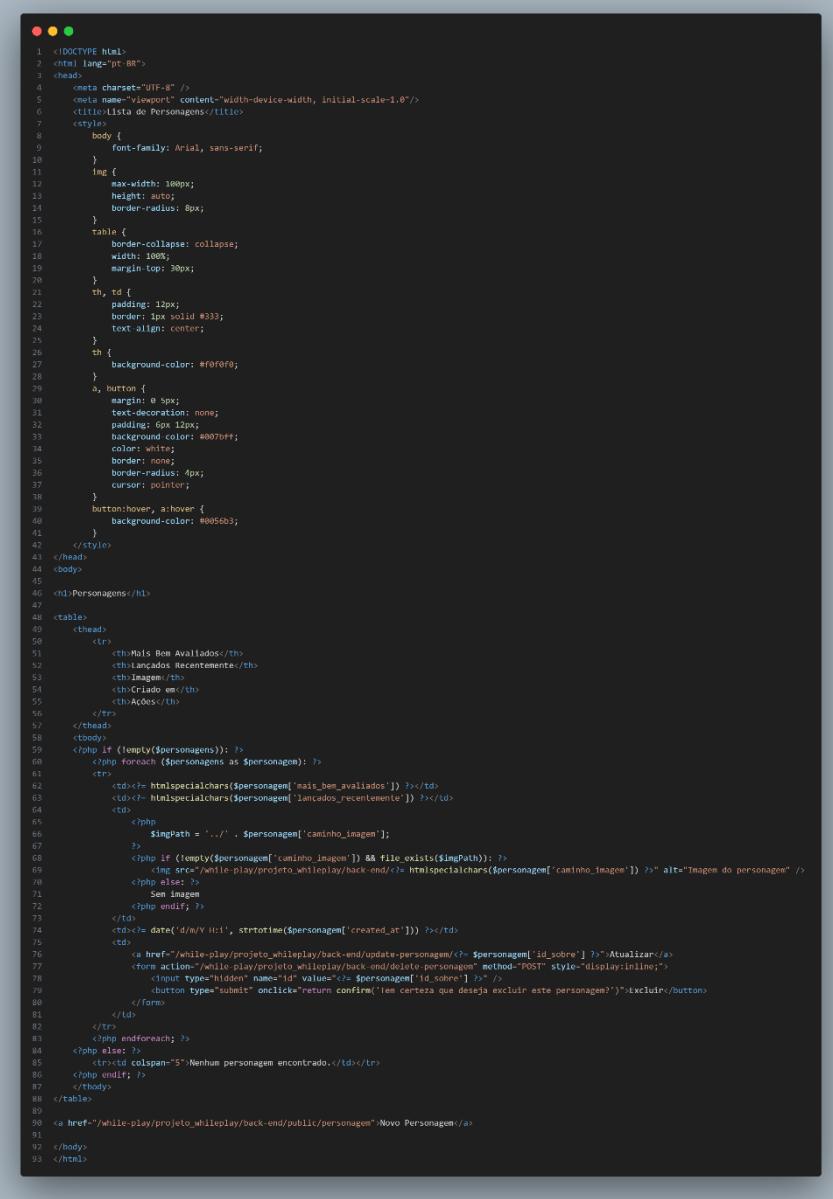
```

**Personagens – form:** Esse código é uma página HTML com um formulário para cadastro de personagens. Ele permite que o usuário insira informações como ID Sobre, quantidade de "Mais Bem Avaliados", "Lançados Recentemente" e também envie uma imagem do personagem. O formulário envia os dados para um script PHP (`save-personagem`) utilizando o método POST e o tipo multipart/form-data, que é necessário para upload de arquivos.

O código também contém um bloco de CSS embutido que estiliza o formulário, deixando-o centralizado na página, com fundo claro, bordas arredondadas, espaçamento interno e campos com largura total. O botão de envio possui uma cor verde, que muda para um tom mais escuro ao passar o mouse. Há ainda um link na

parte inferior que permite ao usuário retornar para a listagem de personagens cadastrados.

Esse formulário é essencial para alimentar o banco de dados com novos personagens, junto com suas imagens e informações, de forma organizada e visualmente agradável.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <title>Lista de Personagens</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10    }
11    img {
12      max-width: 100px;
13      height: auto;
14      border-radius: 8px;
15    }
16    table {
17      border-collapse: collapse;
18      width: 100%;
19      margin-top: 30px;
20    }
21    th, td {
22      padding: 12px;
23      border: 1px solid #333;
24      text-align: center;
25    }
26    th {
27      background-color: #f0f0f0;
28    }
29    a, button {
30      margin: 0 8px;
31      text-decoration: none;
32      padding: 6px 12px;
33      background-color: #007bff;
34      color: white;
35      border: none;
36      border-radius: 4px;
37      cursor: pointer;
38    }
39    button:hover, a:hover {
40      background-color: #0056b3;
41    }
42   </style>
43 </head>
44 <body>
45
46 <h1>Personagens</h1>
47
48 <table>
49   <thead>
50     <tr>
51       <th>Mais Bem Avaliados</th>
52       <th>Último Lançamento</th>
53       <th>Imagem</th>
54       <th>Criado em</th>
55       <th>Ações</th>
56     </tr>
57   </thead>
58   <tbody>
59   <?php if (!empty($personagens)): ?>
60     <?php foreach ($personagens as $personagem): ?>
61       <tr>
62         <td><?php echo htmlspecialchars($personagem['mais_bem_avaliados']); ?></td>
63         <td><?php echo htmlspecialchars($personagem['lançados_recentemente']); ?></td>
64         <td>
65           <?php
66             $imgPath = './' . $personagem['caminho_imagem'];
67           ><?php if (!empty($personagem['caminho_imagem']) && file_exists($imgPath)): ?>
68             
69           <?php else: ?>
70             Sem imagem
71           <?php endif; ?>
72           <td><?php echo date('d/m/Y H:i', strtotime($personagem['created_at'])); ?></td>
73           <td>
74             <a href="/while-play/projeto.whiteplay/back-end/update-personagem/?=<?php echo $personagem['id_sobre']; ?>">Atualizar</a>
75             <a href="/while-play/projeto.whiteplay/back-end/delete-personagem" method="POST" style="display:inline;">
76               <input type="hidden" name="id" value=<?php echo $personagem['id_sobre']; ?>>
77               <button type="submit" onclick="return confirm('Tem certeza que deseja excluir este personagem?')>Excluir</button>
78             </a>
79           </td>
80         </tr>
81       <?php endforeach; ?>
82     <?php else: ?>
83       <tr><td colspan="5">Nenhum personagem encontrado.</td></tr>
84     <?php endif; ?>
85   </tbody>
86 </table>
87
88 <a href="/while-play/projeto.whiteplay/back-end/public/personagem">Novo Personagem</a>
89
90 </body>
91 </html>

```

**Personagens – list:** Esse código é uma página HTML com PHP que exibe uma tabela de listagem dos personagens cadastrados no sistema. A interface é estilizada com CSS para deixar a tabela organizada, com células bem definidas, bordas, espaçamentos e uma aparência limpa e agradável. As imagens dos personagens são exibidas com bordas arredondadas, e caso não haja imagem, aparece o texto “Sem imagem”.

Cada linha da tabela mostra informações como "Mais Bem Avaliados", "Lançados Recentemente", uma imagem (se houver), a data de criação do cadastro e ações. As ações permitem editar (botão "Atualizar") ou excluir o personagem (botão "Excluir", que pede uma confirmação antes de remover).

O código verifica se há personagens cadastrados e, caso não existam, exibe uma mensagem informando que nenhum personagem foi encontrado. No final da página, há um link que leva o usuário para a página de cadastro de um novo personagem. Assim, essa página funciona como um painel de gerenciamento dos personagens no sistema.

```

#!/usr/bin/env php
require_once __DIR__ . '/model/personagem.php';
class PersonagemController {
    public function index() {
        $view = View::create('personagens_form.php');
    }
}

public function inserirPersonagem() {
    if (!$_FILES['image']['error']) {
        $extensao = $_FILES['image']['type'];
        $tamanho = $_FILES['image']['size'];
        $tempo = $_FILES['image']['tmp_name'];
        $novoNome = md5(uniqid(rand(), true)) . '.' . $extensao;
        $diretorio = 'uploads/' . $novoNome;
        move_uploaded_file($tempo, $diretorio);
        $urlImagem = 'uploads/' . $novoNome;
    } else {
        $urlImagem = null;
    }

    $personagem = new Personagem();
    $personagem->setNome($_POST['nome']);
    $personagem->setDescricao($_POST['descricao']);
    $personagem->setGenero($_POST['genero']);
    $personagem->setAnoLancamento($_POST['ano_lancamento']);
    $personagem->setNotaMedia($_POST['nota_media']);
    $personagem->setNotaCritica($_POST['nota_critica']);
    $personagem->setNotaPublico($_POST['nota_publico']);
    $personagem->setNotaRevisor($_POST['nota_revisor']);
    $personagem->setNotaEditorial($_POST['nota_editorial']);

    $personagem->setImagem($urlImagem); // setar imagem antes de salvar
    $personagem->salvar(); // salvar personagem
}

public function alterarPersonagem($id) {
    $personagem = Personagem::getById($id);
    $view = View::create('personagens_form.php');
}

public function excluirPersonagem($id) {
    $personagem = Personagem::getById($id);
    $personagem->deletar();
}

public function listarPersonagens() {
    $personagens = Personagem::listarBasicos();
    $view = View::create('listar_personagens.php');
    $view->set('personagens', $personagens);
    $view->set('url_imagem', 'uploads/');
}

header('location: /public/painel/administrador/listar_personagens');
}

class PersonagemController {
    public function index() {
        $view = View::create('personagens_form.php');
    }
}

public function inserirPersonagem() {
    if (!$_FILES['image']['error']) {
        $extensao = $_FILES['image']['type'];
        $tamanho = $_FILES['image']['size'];
        $tempo = $_FILES['image']['tmp_name'];
        $novoNome = md5(uniqid(rand(), true)) . '.' . $extensao;
        $diretorio = 'uploads/' . $novoNome;
        move_uploaded_file($tempo, $diretorio);
        $urlImagem = 'uploads/' . $novoNome;
    } else {
        $urlImagem = null;
    }

    $personagem = new Personagem();
    $personagem->setNome($_POST['nome']);
    $personagem->setDescricao($_POST['descricao']);
    $personagem->setGenero($_POST['genero']);
    $personagem->setAnoLancamento($_POST['ano_lancamento']);
    $personagem->setNotaMedia($_POST['nota_media']);
    $personagem->setNotaCritica($_POST['nota_critica']);
    $personagem->setNotaPublico($_POST['nota_publico']);
    $personagem->setNotaRevisor($_POST['nota_revisor']);
    $personagem->setNotaEditorial($_POST['nota_editorial']);

    $personagem->setImagem($urlImagem); // setar imagem antes de salvar
    $personagem->salvar(); // salvar personagem
}

public function alterarPersonagem($id) {
    $personagem = Personagem::getById($id);
    $view = View::create('personagens_form.php');
}

public function excluirPersonagem($id) {
    $personagem = Personagem::getById($id);
    $personagem->deletar();
}

public function listarPersonagens() {
    $personagens = Personagem::listarBasicos();
    $view = View::create('listar_personagens.php');
    $view->set('personagens', $personagens);
}

header('location: /public/painel/administrador/listar_personagens');
}

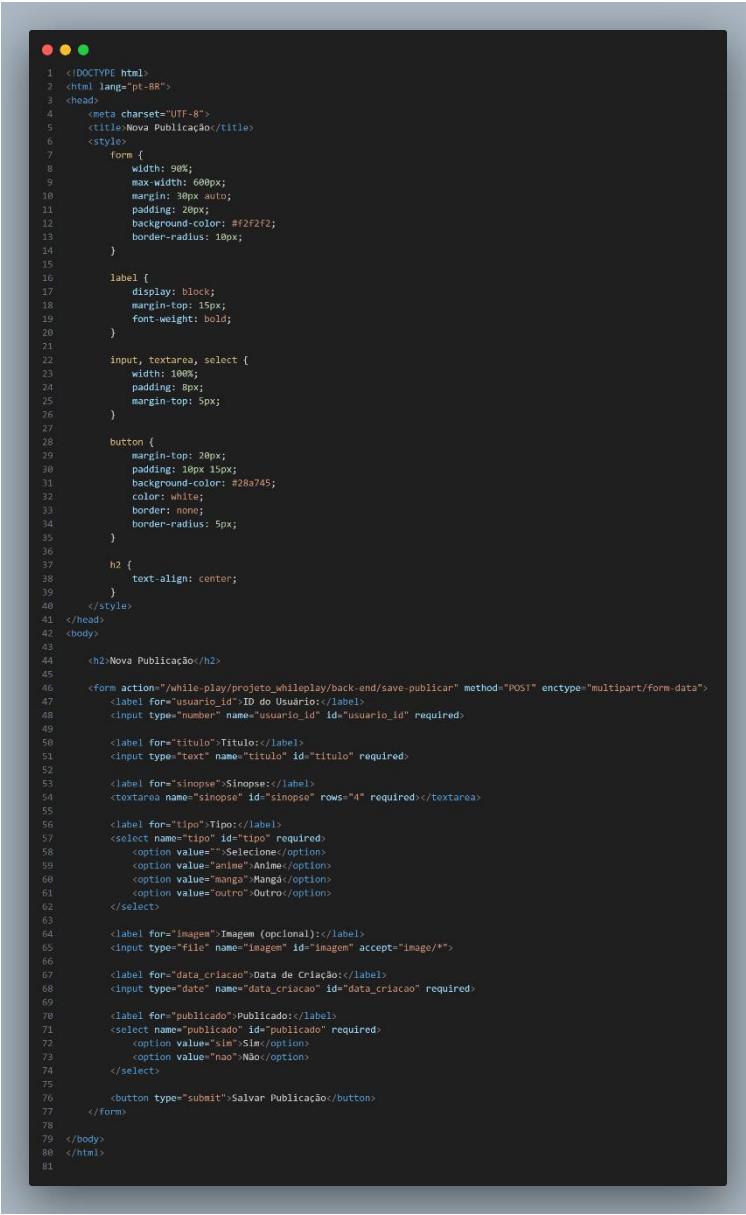
```

**Personagens – Controller:** Este código é um controlador PHP chamado PersonagemController responsável pelas operações de cadastro, listagem, atualização e exclusão de personagens. Ele gerencia também o upload e armazenamento das imagens dos personagens.

O método `showForm()` carrega o formulário de cadastro, enquanto `savePersonagem()` recebe os dados enviados pelo formulário, faz o upload da imagem (se houver), gera um nome único para evitar conflitos e salva as informações no banco de dados.

O método `listPersonagens()` busca todos os personagens e exibe na lista. Já o `deletePersonagemById($id)` remove um personagem específico pelo ID. O `showUpdateForm($id)` carrega os dados de um personagem existente para edição. Por fim, `updatePersonagem()` permite alterar os dados e, se o usuário enviar uma nova imagem, ela substitui a anterior — incluindo a opção de excluir a imagem antiga do servidor, evitando arquivos desnecessários.

Esse controlador garante o funcionamento completo das operações CRUD (Criar, Ler, Atualizar e Deletar) no sistema de gerenciamento de personagens.



```

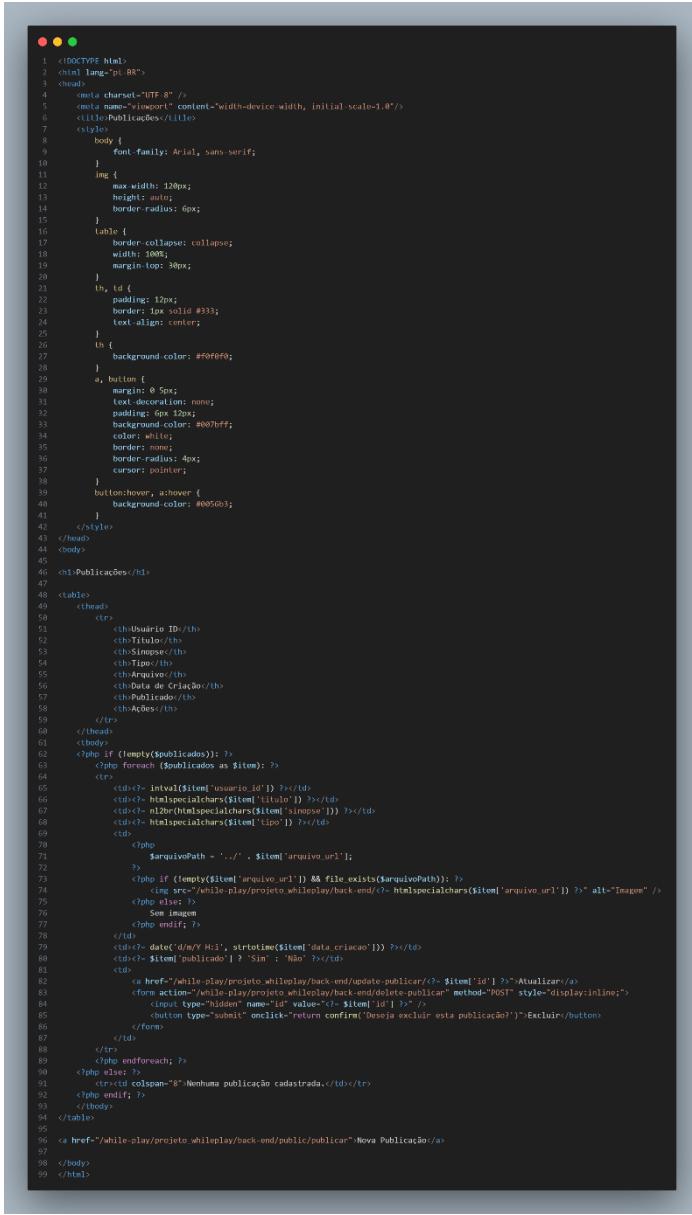
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <title>Nova Publicação</title>
6   <style>
7     form {
8       width: 90%;
9       max-width: 600px;
10      margin: 30px auto;
11      padding: 20px;
12      background-color: #f2f2f2;
13      border-radius: 10px;
14    }
15
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21
22    input, textarea, select {
23      width: 100%;
24      padding: 8px;
25      margin-top: 5px;
26    }
27
28    button {
29      margin-top: 20px;
30      padding: 10px 15px;
31      background-color: #28a745;
32      color: white;
33      border: none;
34      border-radius: 5px;
35    }
36
37    h2 {
38      text-align: center;
39    }
40  </style>
41 </head>
42 <body>
43
44  <h2>Nova Publicação</h2>
45
46  <form action="/while-play/projeto.whiteplay/back-end/save-publicar" method="POST" enctype="multipart/form-data">
47    <label for="usuario_id">ID do Usuário:</label>
48    <input type="number" name="usuario_id" id="usuario_id" required>
49
50    <label for="titulo">Título:</label>
51    <input type="text" name="titulo" id="titulo" required>
52
53    <label for="sinopse">Sinopse:</label>
54    <textarea name="sinopse" id="sinopse" rows="4" required></textarea>
55
56    <label for="tipo">Tipo:</label>
57    <select name="tipo" id="tipo" required>
58      <option value="">Selecione</option>
59      <option value="anime">Anime</option>
60      <option value="manga">Mangá</option>
61      <option value="outro">Outro</option>
62    </select>
63
64    <label for="imagem">Imagem (opcional):</label>
65    <input type="file" name="imagem" id="imagem" accept="image/*">
66
67    <label for="data_criacao">Data de Criação:</label>
68    <input type="date" name="data_criacao" id="data_criacao" required>
69
70    <label for="publicado">Publicado:</label>
71    <select name="publicado" id="publicado" required>
72      <option value="sim">Sim</option>
73      <option value="nao">Não</option>
74    </select>
75
76    <button type="submit">Salvar Publicação</button>
77  </form>
78
79 </body>
80 </html>
81

```

**Publicar – form:** Este código HTML representa um formulário para criar uma nova publicação. Ele coleta informações como ID do usuário, título, sinopse, tipo de publicação (anime, mangá ou outro), data de criação, status de publicado e permite também o upload de uma imagem opcional.

O formulário está estilizado com CSS interno, garantindo uma aparência mais agradável. Ele possui campos bem organizados e alinhados, com espaçamentos, preenchimentos e um botão de envio estilizado em verde, que melhora a experiência do usuário.

Ao ser preenchido e enviado, os dados são enviados via método POST para o script PHP localizado no caminho back-end/save-publicar. O `enctype="multipart/form-data"` permite que arquivos (como imagens) sejam enviados junto com os outros dados.



```

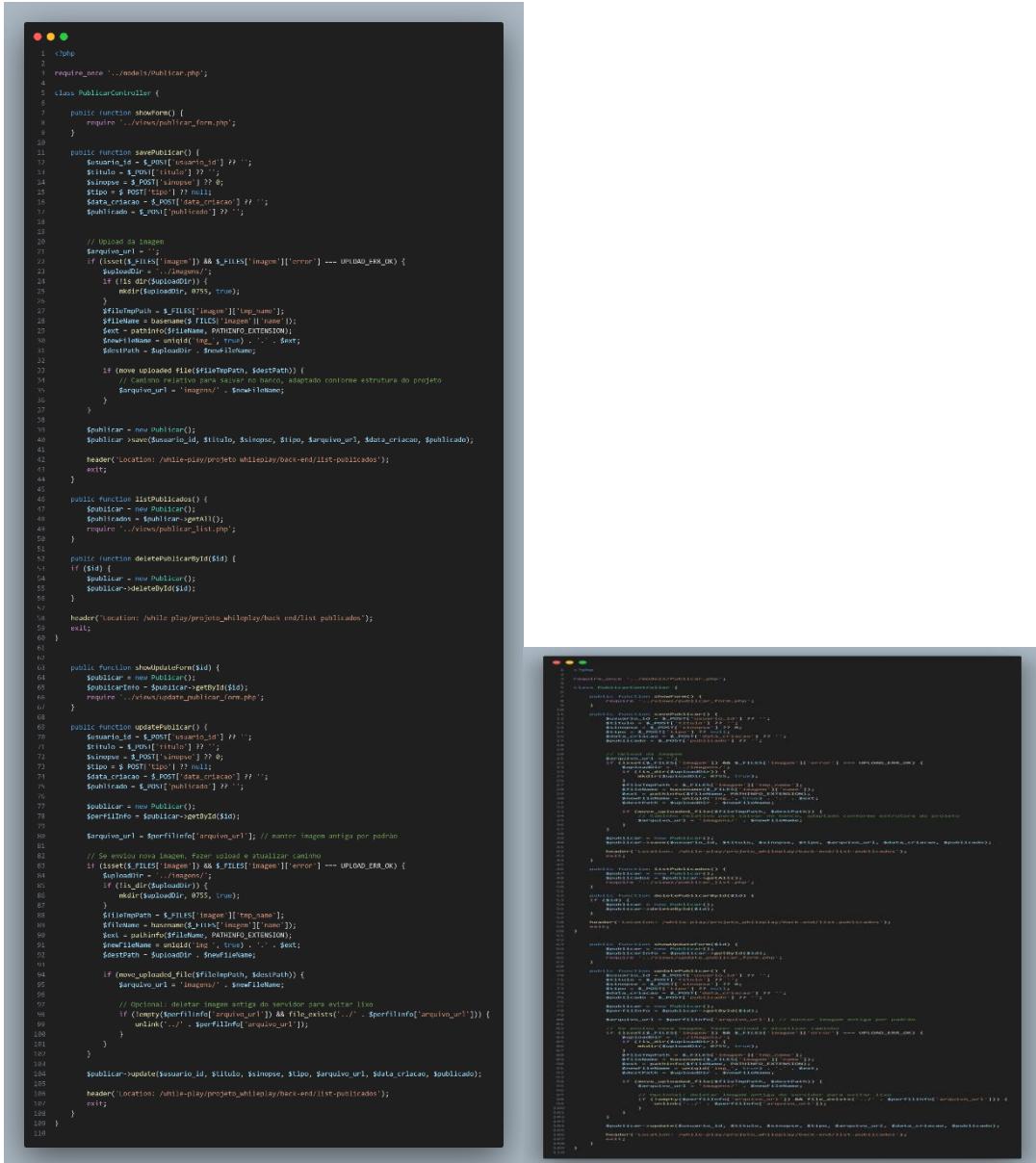
1 <!DOCTYPE html>
2 <html lang="pt_BR">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Publicações</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10    }
11    img {
12      max-width: 120px;
13      height: auto;
14      border-radius: 6px;
15    }
16    table {
17      border-collapse: collapse;
18      width: 100%;
19      margin-top: 30px;
20    }
21    th, td {
22      padding: 12px;
23      border: 1px solid #333;
24      text-align: center;
25    }
26    th {
27      background-color: #f0f0f0;
28    }
29    a {
30      margin: 0 5px;
31      text-decoration: none;
32      padding: 6px 12px;
33      background-color: #007bff;
34      color: white;
35      border: 1px solid #0056b3;
36      border-radius: 4px;
37      cursor: pointer;
38    }
39    a:hover, a:active {
40      background-color: #0056b3;
41    }
42  </style>
43 </head>
44 <body>
45 
46 <h1>Publicações</h1>
47 
48 <table>
49   <thead>
50     <tr>
51       <th>Usuário ID:</th>
52       <th>Título:</th>
53       <th>Sinopse:</th>
54       <th>Tipo:</th>
55       <th>Data de Criação:</th>
56       <th>Publicado:</th>
57       <th>Ações:</th>
58     </tr>
59   </thead>
60   <tbody>
61     <php if (!empty($publicados)): >
62       <php foreach ($publicados as $item): >
63         <tr>
64           <td><?php echo $item['usuario_id']; ></td>
65           <td><?php echo htmlspecialchars($item['titulo']); ></td>
66           <td><?php echo nl2br(htmlspecialchars($item['sinopse'])); ></td>
67           <td><?php echo htmlspecialchars($item['tipo']); ></td>
68           <td>
69             <?php
70               $arquivoPath = '../../../../../' . $item['arquivo_url'];
71             ></td>
72             <?php if (!empty($item['arquivo_url']) && file_exists($arquivoPath)): >
73               
74             <?php else: >
75               Sem imagem
76             </?php endif; >
77           </td>
78           <td>
79             <?php echo date('d/m/Y H:i', strtotime($item['data_criacao'])); ></td>
80             <?php if ($item['publicado'] == 1) ?> Sim : Não </td>
81           <td>
82             <a href="while-play/projeto/whileplay/back-end/update-publicar/<?php echo $item['id']; ?>">Atualizar</a>
83             <form action="while-play/projeto/whileplay/back-end/delete-publicar" method="POST" style="display:inline;">
84               <input type="hidden" name="id" value=<?php echo $item['id']; ?> />
85               <button type="submit" onclick="return confirm('Deseja excluir esta publicação?')>Excluir</button>
86             </form>
87           </td>
88         </tr>
89       <php endforeach; >
90     <php else: >
91       <tr><td colspan="8">Nenhuma publicação cadastrada.</td></tr>
92     <php endif; >
93   </tbody>
94 </table>
95 
96 <a href="while-play/projeto/whileplay/back-end/public/publicar">Nova Publicação</a>
97 
98 </body>
99 </html>

```

**Publicar – list:** Esse código HTML exibe uma **lista de publicações** em formato de tabela. Ele mostra informações como ID do usuário, título, sinopse, tipo, imagem (arquivo), data de criação, status de publicado e opções de ações (atualizar ou excluir).

O layout da tabela é estilizado com CSS interno, deixando o visual mais limpo, com cantos arredondados nas imagens e botões azuis para as ações. Se não houver imagem, aparece a mensagem "Sem imagem".

Os dados são gerados dinamicamente em PHP, percorrendo o array `$publicados`. Existem também links para atualizar ou excluir uma publicação, além de um botão no final da página que leva para o formulário de criação de uma nova publicação.



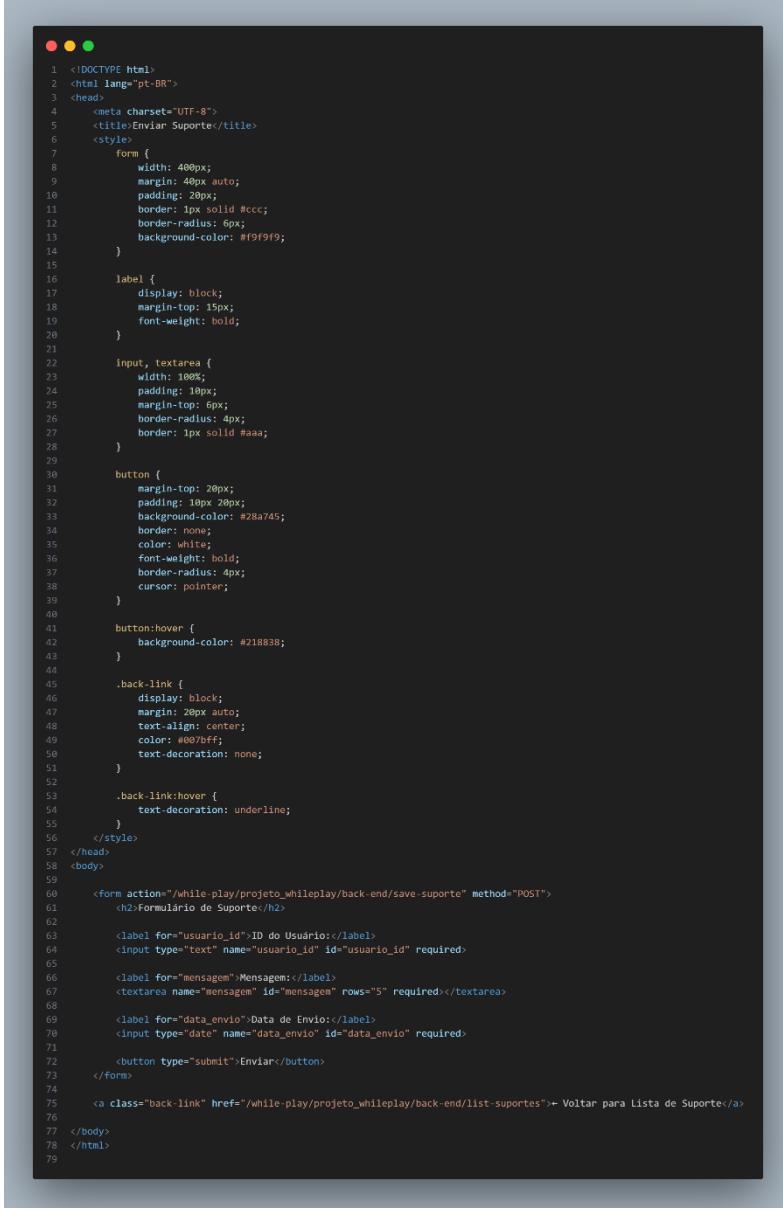
```

1 <?php
2
3 require_once '../models/Publicar.php';
4
5 class PublicarController {
6
7     public function showForm() {
8         require '../views/publicar_form.php';
9     }
10
11    public function savePublicar() {
12        $this = &$_POST;
13        $titulo = $this['titulo'] ?? '';
14        $sinopse = $this['sinopse'] ?? '';
15        $stipo = $this['tipo'] ?? null;
16        $data_criacao = $this['data_criacao'] ?? '';
17        $publicado = $this['publicado'] ?? '';
18
19        // Usando as imagens
20        $arquivo_url = '';
21        if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
22            $uploadDir = __DIR__ . '/uploads/';
23            $fileName = basename($_FILES['image']['name']);
24            $ext = pathinfo($fileName, PATHINFO_EXTENSION);
25            $newFileName = uniqid('img_') . '.' . $ext;
26            $destPath = $uploadDir . $newFileName;
27
28            if (move_uploaded_file($fileTmpPath, $destPath)) {
29                // Caminho relativo para salvar no banco, adaptado conforme estrutura do projeto
30                $arquivo_url = 'images/' . $newFileName;
31            }
32        }
33
34        $publicar = new Publicar();
35        $publicar->save($usuario_id, $titulo, $sinopse, $stipo, $arquivo_url, $data_criacao, $publicado);
36
37        header('Location: /while-play/projeto_whileplay/back-end/list-publicados');
38        exit();
39    }
40
41    public function listPublicados() {
42        $publicar = new Publicar();
43        $publicar->list();
44        require '../views/publicar_list.php';
45    }
46
47    public function deletePublicarById($id) {
48        if ($id) {
49            $publicar = new Publicar();
50            $publicar->deleteById($id);
51        }
52
53        header('Location: /while-play/projeto_whileplay/back-end/list-publicados');
54        exit();
55    }
56
57    public function showUpdateForm($id) {
58        $publicar = new Publicar();
59        $publicarInfo = $publicar->getById($id);
60        require '../views/publicar_form.php';
61    }
62
63    public function updatePublicar() {
64        $usuario_id = $_POST['usuario_id'] ?? '';
65        $titulo = $_POST['titulo'] ?? '';
66        $sinopse = $_POST['sinopse'] ?? '';
67        $stipo = $_POST['tipo'] ?? null;
68        $data_criacao = $_POST['data_criacao'] ?? '';
69        $publicado = $_POST['publicado'] ?? '';
70
71        $publicar = new Publicar();
72        $perfilInfo = $publicar->getById($id);
73
74        $arquivo_url = $perfilInfo['arquivo_url']; // Manter imagem antiga por padrão
75
76        // Se envio nova imagem, fazer upload e atualizar caminho
77        if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
78            $uploadDir = __DIR__ . '/uploads/';
79            $fileName = basename($_FILES['image']['name']);
80            $ext = pathinfo($fileName, PATHINFO_EXTENSION);
81            $newFileName = uniqid('img_') . '.' . $ext;
82            $destPath = $uploadDir . $newFileName;
83
84            if (move_uploaded_file($fileTmpPath, $destPath)) {
85                $arquivo_url = 'images/' . $newFileName;
86            }
87
88            $stiloPath = $perfilInfo['image'][0]['tmp_name'];
89            $fileName = basename($stiloPath['image'][0]['name']);
90            $ext = pathinfo($fileName, PATHINFO_EXTENSION);
91            $newFileName = uniqid('img_') . '.' . $ext;
92            $destPath = $uploadDir . $newFileName;
93
94            if (move_uploaded_file($stiloPath, $destPath)) {
95                $arquivo_url = 'images/' . $newFileName;
96
97                // Optional: deletar imagem antiga do servidor para evitar lixo
98                if (!empty($perfilInfo['arquivo_url'])) && file_exists('../' . $perfilInfo['arquivo_url'])) {
99                    unlink('../' . $perfilInfo['arquivo_url']);
100                }
101            }
102        }
103
104        $publicar->update($usuario_id, $titulo, $sinopse, $stipo, $arquivo_url, $data_criacao, $publicado);
105
106        header('Location: /while-play/projeto_whileplay/back-end/list-publicados');
107        exit();
108    }
109
110 }

```

**Publicar – Controller:** O código representa um **controller PHP** chamado **PublicarController**, que é responsável por gerenciar as ações de um sistema de publicações. Ele contém um método para **exibir o formulário de cadastro** (**showForm**) e outro para **salvar uma nova publicação** (**savePublicar**), que também realiza o upload da imagem, gera um nome único para o arquivo e cria a pasta de imagens caso ela não exista. Além disso, possui um método para **listar todas as publicações** (**listPublicados**) e outro para **excluir uma publicação** com base no seu ID (**deletePublicarById**). Há também um método para **exibir o formulário de edição** (**showUpdateForm**) e um método para **atualizar uma publicação** existente (**updatePublicar**), que permite alterar os dados e, caso uma nova imagem seja enviada, substitui a antiga e remove a anterior do servidor para evitar arquivos

desnecessários. O código também faz redirecionamentos após cada operação para a página de listagem das publicações.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Enviar Suporte</title>
6   <style>
7     form {
8       width: 400px;
9       margin: 40px auto;
10      padding: 20px;
11      border: 1px solid #ccc;
12      border-radius: 6px;
13      background-color: #f9f9f9;
14    }
15
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21
22    input, textarea {
23      width: 100%;
24      padding: 10px;
25      margin-top: 6px;
26      border-radius: 4px;
27      border: 1px solid #aaa;
28    }
29
30    button {
31      margin-top: 20px;
32      padding: 10px 20px;
33      background-color: #28a745;
34      border: none;
35      color: white;
36      font-weight: bold;
37      border-radius: 4px;
38      cursor: pointer;
39    }
40
41    button:hover {
42      background-color: #218838;
43    }
44
45    .back-link {
46      display: block;
47      margin: 20px auto;
48      text-align: center;
49      color: #007bff;
50      text-decoration: none;
51    }
52
53    .back-link:hover {
54      text-decoration: underline;
55    }
56  </style>
57 </head>
58 <body>
59
60   <form action="/while-play/projeto_whileplay/back-end/save-suporte" method="POST">
61     <h2>Formulário de Suporte</h2>
62
63     <label for="usuario_id">ID do Usuário:</label>
64     <input type="text" name="usuario_id" id="usuario_id" required>
65
66     <label for="mensagem">Mensagem:</label>
67     <textarea name="mensagem" id="mensagem" rows="5" required></textarea>
68
69     <label for="data_envio">Data de Envio:</label>
70     <input type="date" name="data_envio" id="data_envio" required>
71
72     <button type="submit">Enviar</button>
73   </form>
74
75   <a class="back-link" href="/while-play/projeto_whileplay/back-end/list-suportes">← Voltar para Lista de Suporte</a>
76
77 </body>
78 </html>
79

```

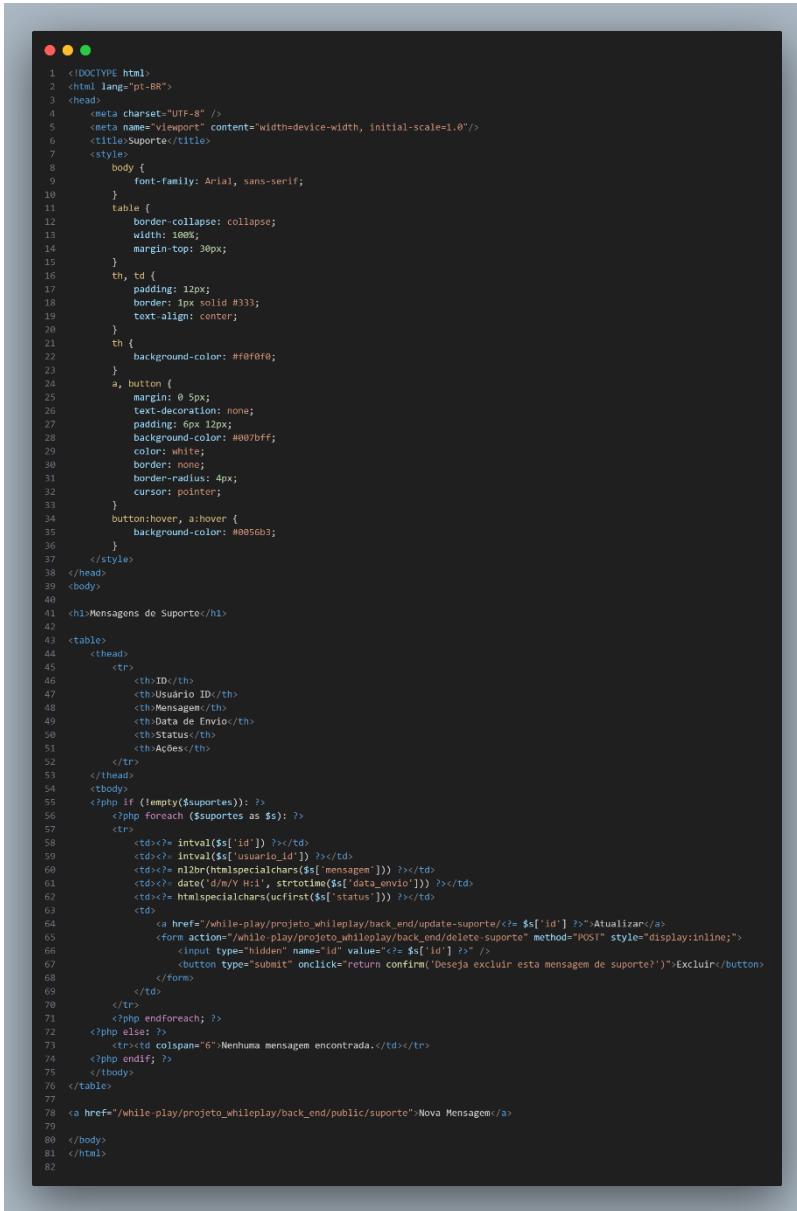
**Suporte – form:** Esse código define uma página HTML com um formulário para envio de mensagens de suporte. A página possui um título e um estilo CSS interno para deixar o formulário visualmente organizado e agradável. O formulário está centralizado na tela, com uma largura fixa, bordas arredondadas e cores suaves.

No formulário, há três campos principais: um campo de texto para o "ID do Usuário", uma área de texto para a "Mensagem" de suporte e um campo de data para a "Data de Envio". Todos esses campos são obrigatórios, garantidos pelo atributo `required`, para que o usuário não possa enviar o formulário sem preenchê-los.

Quando o usuário preenche o formulário e clica no botão "Enviar", os dados são enviados via método POST para a URL /while-play/projeto\_whileplay/backend/save-suporte, onde provavelmente o backend processará e salvará essa informação.

Além disso, há um link na parte inferior da página que permite voltar para a lista de suportes já cadastrados, facilitando a navegação entre as páginas do sistema.

O botão de envio tem um estilo verde que escurece quando o mouse passa por cima, indicando interatividade. Os campos do formulário são estilizados para ocupar toda a largura disponível e terem espaçamento confortável para facilitar o preenchimento.



```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <title>Suporte</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10    }
11   table {
12     border-collapse: collapse;
13     width: 100%;
14     margin-top: 30px;
15   }
16   th, td {
17     padding: 12px;
18     border: 1px solid #333;
19     text-align: center;
20   }
21   th {
22     background-color: #e0e0e0;
23   }
24   a, button {
25     margin: 0 5px;
26     text-decoration: none;
27     padding: 6px 12px;
28     background-color: #007bff;
29     color: white;
30     border: none;
31     border-radius: 4px;
32     cursor: pointer;
33   }
34   button:hover, a:hover {
35     background-color: #0056b3;
36   }
37 </style>
38 </head>
39 <body>
40
41 <h1>Mensagens de Suporte</h1>
42
43 <table>
44   <thead>
45     <tr>
46       <th>ID</th>
47       <th>Usuário ID</th>
48       <th>Mensagem</th>
49       <th>Data de Envio</th>
50       <th>Status</th>
51       <th>Ações</th>
52     </tr>
53   </thead>
54   <tbody>
55     <?php if (!empty($suportes)): ?>
56     <?php foreach ($suportes as $s): ?>
57       <tr>
58         <td><?= intval($s['id']) ?></td>
59         <td><?= intval($s['usuario_id']) ?></td>
60         <td><?= nl2br(htmlspecialchars($s['mensagem'])) ?></td>
61         <td><?= date('d/m/Y H:i', strtotime($s['data_envio'])) ?></td>
62         <td><?= htmlspecialchars(ucfirst($s['status'])) ?></td>
63         <td>
64           <a href="/while-play/projeto_whileplay/back_end/update-suporte/?=>$s['id'] ?>">Atualizar</a>
65           <form action="/while-play/projeto_whileplay/back_end/delete-suporte" method="POST" style="display:inline;">
66             <input type="hidden" name="id" value=<?= $s['id'] ?>" />
67             <button type="submit" onclick="return confirm('Deseja excluir esta mensagem de suporte?')">Excluir</button>
68           </form>
69         </td>
70       </tr>
71     <?php endforeach; ?>
72     <?php else: ?>
73       <tr><td colspan="6">Nenhuma mensagem encontrada.</td></tr>
74     <?php endif; ?>
75   </tbody>
76 </table>
77
78 <a href="/while-play/projeto_whileplay/back_end/public/suporte">Nova Mensagem</a>
79
80 </body>
81 </html>
82

```

**Suporte – list:** Esse código gera uma página HTML que lista as mensagens de suporte cadastradas no sistema, exibindo-as em uma tabela organizada. Cada linha

da tabela mostra o ID da mensagem, o ID do usuário que enviou, o conteúdo da mensagem, a data e hora de envio formatada, o status atual e ações disponíveis.

O código PHP dentro da tabela verifica se existem mensagens na variável `$suportes`. Se houver, ele percorre cada suporte e exibe os dados correspondentes. O conteúdo da mensagem é protegido com `htmlspecialchars` para evitar ataques XSS, e quebras de linha são mantidas com `\n\n`. A data é formatada para o padrão brasileiro `dd/mm/aaaa hh:mm`.

Na coluna de ações, o usuário pode clicar para "Atualizar" a mensagem, acessando uma rota específica com o ID da mensagem, ou pode excluir a mensagem por meio de um formulário com método POST, que confirma a exclusão antes de enviar.

Se não houver mensagens cadastradas, a tabela exibe uma linha única informando que não foi encontrada nenhuma mensagem.



```

1 <?php
2
3 require_once '../models/Suporte.php';
4
5 class SuporteController {
6
7     public function showForm() {
8         require '../views/suporte_form.php';
9     }
10
11    public function saveSuporte() {
12        $usuario_id = $_POST['usuario_id'] ?? '';
13        $mensagem = $_POST['mensagem'] ?? '';
14        $data_envio = $_POST['data_envio'] ?? '';
15
16        $suporte = new Suporte();
17        $suporte->save($usuario_id, $mensagem, $data_envio);
18
19        header('Location: /while-play/projeto_whileplay/back-end/list-suportes');
20    }
21
22    public function listSuportes() {
23        $suporte = new Suporte();
24        $suportes = $suporte->getAll();
25        require '../views/suporte_list.php';
26    }
27
28    public function deleteSuporteByTitle() {
29        $usuario_id = $_POST['usuario_id'] ?? null;
30        if ($usuario_id) {
31            $suporte = new Suporte();
32            $suporte->deleteByTitle($usuario_id);
33        }
34        header('Location: /while-play/projeto_whileplay/back-end/list-suportes');
35    }
36
37    public function showUpdateForm($id) {
38        $suporte = new Suporte();
39        $suporteInfo = $suporte->getById($id);
40        require '../views/update_suporte_form.php';
41    }
42
43    public function updateSuporte() {
44        $id = $_POST['id'];
45        $usuario_id = $_POST['usuario_id'];
46        $mensagem = $_POST['mensagem'];
47        $data_envio = $_POST['data_envio'];
48
49        $suporte = new Suporte();
50        $suporte->update($id, $usuario_id, $mensagem, $data_envio);
51
52        header('Location: /while-play/projeto_whileplay/back-end/list-suportes');
53    }
54 }
```

**Suporte – Controller:** Esse código define a classe SuporteController, que gerencia as operações relacionadas às mensagens de suporte no sistema. Ela carrega os formulários para criar ou atualizar mensagens, salva os dados enviados pelo usuário, lista todas as mensagens cadastradas e também permite excluir mensagens específicas. Para isso, utiliza o modelo Suporte para acessar e modificar os dados no banco.

Os métodos do controlador seguem o padrão MVC, cuidando do fluxo entre as views e o modelo. Após executar operações como salvar, atualizar ou deletar uma mensagem, o controlador redireciona o usuário para a lista atualizada de suportes, garantindo uma navegação fluida e organizada dentro do sistema.

```

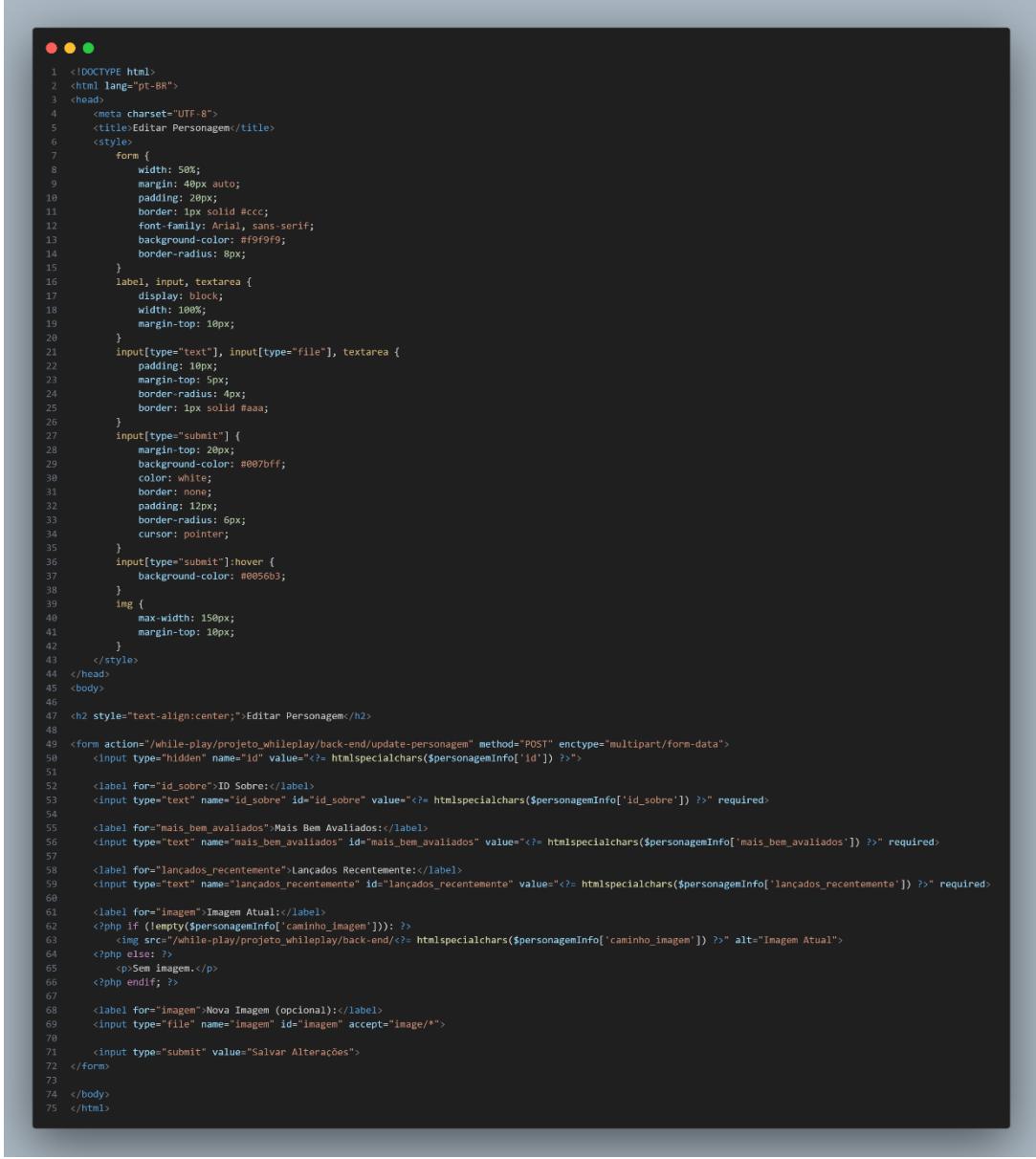
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <title>Alvalular Perfil</title>
6   <style>
7     form {
8       width: 400px;
9       margin: 40px auto;
10      padding: 20px;
11      border: 1px solid #ccc;
12      border-radius: 4px;
13      background-color: #f0f0f0;
14    }
15
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21
22    input, textarea {
23      width: 200px;
24      padding: 10px;
25      margin-top: 5px;
26      border-radius: 4px;
27      border: 1px solid #ccc;
28      box-sizing: border-box;
29    }
30
31    button {
32      margin-top: 20px;
33      padding: 12px 25px;
34      background-color: #007bff;
35      border: none;
36      color: white;
37      font-weight: bold;
38      border-radius: 4px;
39      cursor: pointer;
40    }
41
42    button:hover {
43      background-color: #0066cc;
44    }
45
46    .back-link {
47      display: block;
48      margin: 20px auto;
49      text-align: center;
50      color: black;
51      text-decoration: none;
52    }
53
54    .back-link:hover {
55      text-decoration: underline;
56    }
57
58    .current-photo {
59      margin-top: 10px;
60      max-width: 150px;
61      max-height: 150px;
62      border-radius: 4px;
63      border: 1px solid #ccc;
64      object-fit: cover;
65    }
66  </style>
67 </head>
68 <body>
69
70   <form action="/while-play/projeto_unilead/back-end/update-perfil" method="POST" enctype="multipart/form-data">
71     <input type="hidden" name="perfilInfo[id]" value="<${perfilInfo[id]}>" />
72
73     <!-- Campo oculto para identificar o perfil -->
74     <input type="hidden" name="id" value="<${htmlspecialchars($perfilInfo[id])}>" />
75
76     <label for="nome_completo">Nome Completo:</label>
77     <input type="text" id="nome_completo" name="nome_completo" value="<${htmlspecialchars($perfilInfo['nome_completo'])}>" required>
78
79     <label for="username">Username:</label>
80     <input type="text" id="username" name="username" value="<${htmlspecialchars($perfilInfo['username'])}>" required>
81
82     <label for="email">Email:</label>
83     <input type="email" id="email" name="email" value="<${htmlspecialchars($perfilInfo['email'])}>" required>
84
85     <label for="senha">Senha:</label> (Deixe em branco para manter a atual)<input type="password" id="senha" name="senha" placeholder="Nova senha">
86
87     <label for="biografia">Biografia:</label>
88     <textarea id="biografia" name="biografia" rows="4"><${htmlspecialchars($perfilInfo['biografia'])}</textarea>
89
90     <label for="data_criacao">Data de Criação:</label>
91     <input type="date" id="data_criacao" name="data_criacao" value="<${htmlspecialchars($perfilInfo['data_criacao'])}>" required>
92
93     <label for="foto_perfil">Foto do Perfil:</label>
94     
95
96     <input type="file" name="foto_perfil" value="" />
97     <input type="button" value="Alterar foto..." />
98
99     
100    
101
102    <label for="imagem">Alterar foto:</label>
103    <input type="file" id="imagem" name="imagem" accept="image/*">
104
105    <button type="submit">Salvar Alterações</button>
106
107  </form>
108
109  <a class="back-link" href="/while-play/projeto_unilead/back-end/list_perfil"> Voltar para lista de perfis </a>
110
111 </body>
112 </html>

```

**Update – perfil – form:** Esse código cria uma página HTML com um formulário para atualizar os dados de um perfil de usuário. O formulário exibe campos para nome completo, username, email, senha (com instrução para deixar em branco se não quiser alterar), biografia, data de criação e uma opção para alterar a foto do perfil. Também mostra a foto atual do usuário, se houver, para referência. Todos os campos essenciais são obrigatórios, e o formulário envia os dados via POST para uma rota no backend responsável por salvar as alterações.

O estilo do formulário é simples e organizado, com campos que ocupam toda a largura disponível e um botão azul que muda de tom ao passar o mouse. Há ainda um link para voltar para a lista de perfis, facilitando a navegação. Essa página oferece uma

interface clara e funcional para que o usuário possa editar seu perfil de forma prática e segura.



```

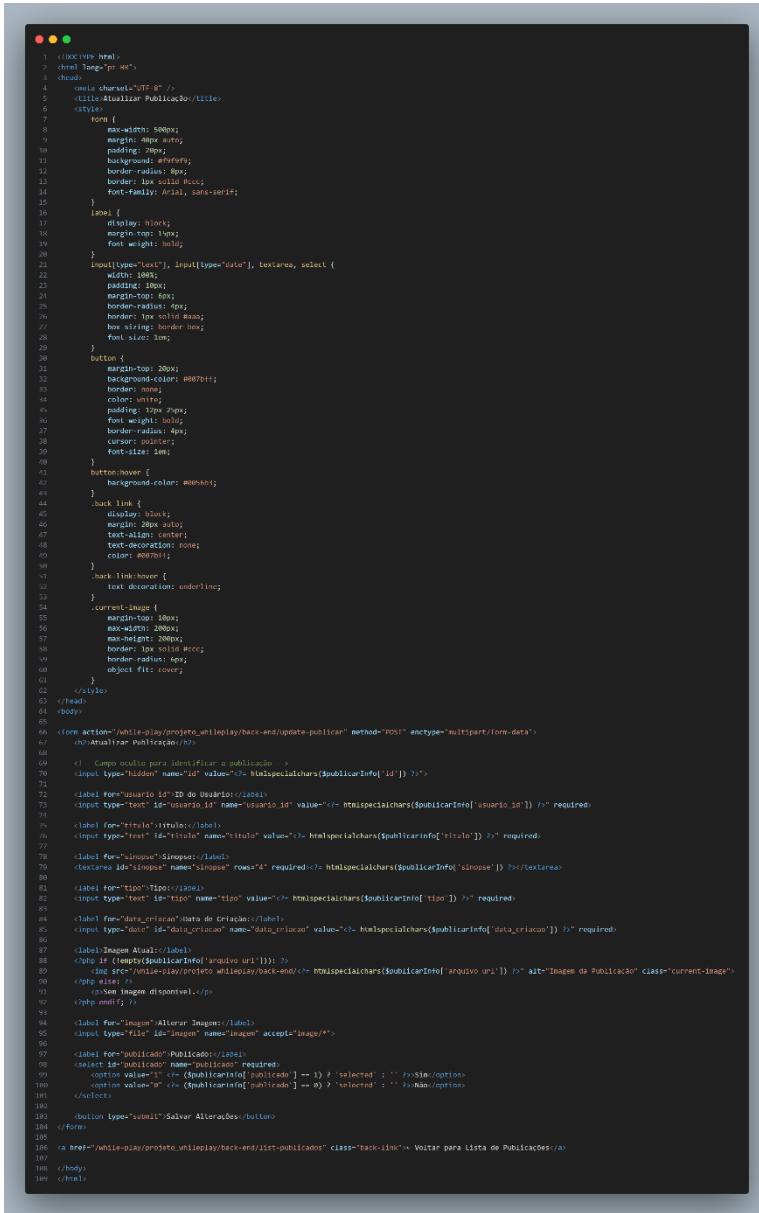
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <title>Editar Personagem</title>
6      <style>
7          form {
8              width: 50%;
9              margin: 40px auto;
10             padding: 20px;
11             border: 1px solid #cccc;
12             font-family: Arial, sans-serif;
13             background-color: #f9f9f9;
14             border-radius: 10px;
15         }
16         label, input, textarea {
17             display: block;
18             width: 100%;
19             margin-top: 10px;
20         }
21         input[type="text"], input[type="file"], textarea {
22             padding: 10px;
23             margin-top: 5px;
24             border-radius: 4px;
25             border: 1px solid #aaa;
26         }
27         input[type="submit"] {
28             margin-top: 20px;
29             background-color: #007bff;
30             color: white;
31             border: none;
32             padding: 12px;
33             border-radius: 6px;
34             cursor: pointer;
35         }
36         input[type="submit"]:hover {
37             background-color: #0056b3;
38         }
39         img {
40             max-width: 150px;
41             margin-top: 10px;
42         }
43     </style>
44 </head>
45 <body>
46     <h2 style="text-align:center;">Editar Personagem</h2>
47     <form action="/while-play/projeto_whileplay/back-end/update-personagem" method="POST" enctype="multipart/form-data">
48         <input type="hidden" name="id" value="= htmlspecialchars($personagemInfo['id']) ?" />>>
49         <label for="id_sobre">ID Sobre:</label>
50         <input type="text" name="id_sobre" id="id_sobre" value="= htmlspecialchars($personagemInfo['id_sobre']) ?" required>
51
52         <label for="mais_bem_avaliados">Mais Bem Avaliados:</label>
53         <input type="text" name="mais_bem_avaliados" id="mais_bem_avaliados" value="= htmlspecialchars($personagemInfo['mais_bem_avaliados']) ?" required>
54
55         <label for="lançados_recentemente">Lançados Recentemente:</label>
56         <input type="text" name="lançados_recentemente" id="lançados_recentemente" value="= htmlspecialchars($personagemInfo['lançados_recentemente']) ?" required>
57
58         <label for="imagem">Imagen Atual:</label>
59         <?php if (!empty($personagemInfo['caminho_imagem'])): ?>
60             
61         <?php else: ?>
62             <p>Sem imagen.</p>
63         <?php endif; ?>
64
65         <label for="nova_imagem">Nova Imagem (opcional):</label>
66         <input type="file" name="nova_imagem" id="nova_imagem" accept="image/*">
67
68         <input type="submit" value="Salvar Alterações">
69     </form>
70
71 </body>
72 </html>

```

**Update – personagens – form:** Esse código define um formulário HTML para edição dos atributos de um personagem, incluindo `id_sobre`, `mais_bem_avaliados` e `lançados_recentemente`, todos enviados via método POST para a rota `/update-personagem`. O formulário utiliza `enctype="multipart/form-data"` para possibilitar o upload opcional de uma nova imagem, exibindo a imagem atual quando disponível.

O layout é estruturado com CSS básico para garantir responsividade e usabilidade, incluindo inputs com validação `required`. O formulário usa PHP para preencher dinamicamente os valores atuais dos campos e a imagem existente, garantindo que

os dados sejam mantidos durante a edição antes da submissão para atualização no backend.



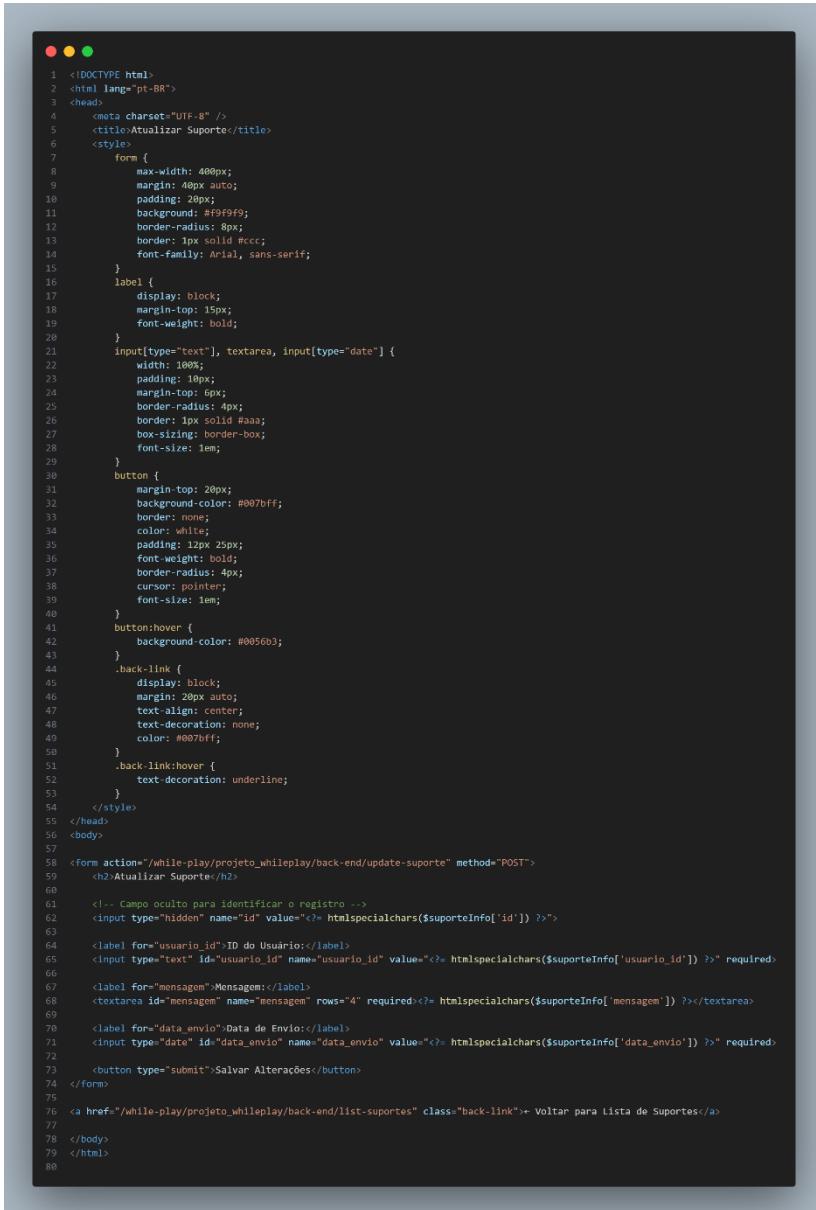
```

1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   meta charset="UTF-8" />
5   <title>Atualizar Publicação</title>
6   <style>
7     form {
8       margin: 50px;
9       margin-left: auto;
10      padding: 20px;
11      background: #f0f0f0;
12      border-radius: 10px;
13      border: 1px solid #ccc;
14      font-family: Arial, sans-serif;
15    }
16    input {
17      display: block;
18      margin-top: 10px;
19      font-weight: bold;
20    }
21    input[type="text"], input[type="date"], textarea, select {
22      width: 100px;
23      padding: 10px;
24      margin-bottom: 10px;
25      border-radius: 5px;
26      border: 1px solid #ccc;
27      box-sizing: border-box;
28      font-size: 1em;
29    }
30    button {
31      margin-top: 20px;
32      background-color: #007bff;
33      border: none;
34      color: white;
35      padding: 10px 20px;
36      font-size: 1em;
37      border-radius: 4px;
38      cursor: pointer;
39      font-size: 1em;
40    }
41    button:hover {
42      background-color: #0056b3;
43    }
44    .back-link {
45      display: block;
46      margin: 20px auto;
47      text-align: center;
48      text-decoration: none;
49      color: #007bff;
50    }
51    .back-link:hover {
52      text-decoration: underline;
53    }
54    .current-image {
55      margin-top: 10px;
56      max-width: 200px;
57      max-height: 200px;
58      border: 1px solid #ccc;
59      border-radius: 10px;
60      object-fit: cover;
61    }
62  </style>
63 </head>
64 <body>
65   <form action="/whiteplay/projeto/whiteplay/back-end/update-publicar" method="POST" enctype="multipart/form-data">
66     <input type="hidden" name="publicarInfo[id]" value="<?php htmlspecialchars($publicarInfo['id']) ?>" />
67     <label> Campo usado para identificar a publicação </label>
68     <input type="hidden" name="publicarInfo[usuario_id]" value="<?php htmlspecialchars($publicarInfo['usuario_id']) ?>" required>
69     <label> ID do Usuário:</label>
70     <input type="text" id="usuario_id" name="usuario_id" value="<?php htmlspecialchars($publicarInfo['usuario_id']) ?>" required>
71     <label> Nome do Usuário:</label>
72     <input type="text" id="nome" name="nome" value="<?php htmlspecialchars($publicarInfo['nome']) ?>" required>
73     <label> Título da publicação:</label>
74     <input type="text" id="titulo" name="titulo" value="<?php htmlspecialchars($publicarInfo['titulo']) ?>" required>
75     <label> Subtítulo da publicação:</label>
76     <input type="text" id="subtitulo" name="subtitulo" value="<?php htmlspecialchars($publicarInfo['subtitulo']) ?>" required>
77     <label> Sinopse da publicação:</label>
78     <input type="text" id="sinopse" name="sinopse" rows="4" required><?php htmlspecialchars($publicarInfo['sinopse']) ?></input>
79     <label> Tipo da publicação:</label>
80     <input type="text" id="tipo" name="tipo" value="<?php htmlspecialchars($publicarInfo['tipo']) ?>" required>
81     <label> Data de criação da publicação:</label>
82     <input type="date" id="data_criacao" name="data_criacao" value="<?php htmlspecialchars($publicarInfo['data_criacao']) ?>" required>
83     <label> Imagem Atual:</label>
84     <input type="file" id="arquivo" name="arquivo" value="<?php if (!empty($publicarInfo['arquivo'])) { echo $publicarInfo['arquivo']; } ?>" alt="Imagem da Publicação" class="current-image">
85     
86     <?php if ($publicarInfo['arquivo']) { >
87       
88     <?php } else { >
89       
90     <?php } ?>
91     <label> Alterar Imagem:</label>
92     <input type="file" id="image" name="image" accept="image/*">
93     <label> Status da publicação:</label>
94     <select type="radio" name="publicado" required>
95       <option value="1" ><?php if ($publicarInfo['publicado'] == 1) { echo "selected"; } ?> Sim</option>
96       <option value="0" ><?php if ($publicarInfo['publicado'] == 0) { echo "selected"; } ?> Não</option>
97     </select>
98   </form>
99   <a href="/whiteplay/projeto/whiteplay/back-end/list-publicados" class="back-link"> Voltar para Lista de Publicações </a>
100 </body>
101 </html>

```

**Update – publicar – form:** Este código HTML/PHP constrói um formulário para atualização de uma publicação no sistema. Ele inclui campos para `usuario_id`, `titulo`, `sinopse`, `tipo`, `data_criacao` e `publicado`, todos preenchidos dinamicamente com os dados atuais usando PHP. Também permite atualizar a imagem vinculada à publicação, utilizando `enctype="multipart/form-data"` para suportar o upload de arquivos. O status de publicação é gerenciado por meio de um campo `select`, que define se a publicação está ativa (1) ou não (0).

O layout é responsivo, com formatação CSS aplicada para melhorar a usabilidade. O botão de envio possui efeito de hover, e a imagem atual da publicação é exibida como referência ao usuário. Todos os dados, ao serem enviados, são processados pela rota /update-publicar, responsável pela atualização no backend. O formulário mantém segurança básica utilizando `htmlspecialchars` para evitar injeções de código nos campos exibidos.



The screenshot shows a web browser window with a dark theme. Inside, there's a form for updating a support ticket. At the top, there's a title "Atualizar Suporte". The form contains fields for "usuario\_id" (a hidden input), "mensagem" (a text area), and "data\_envio" (a date input). Below the form is a button labeled "Salvar Alterações". At the bottom, there's a link "Voltar para Lista de Suportes". The code block below the screenshot is the HTML and CSS for this page.

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8" />
5   <title>Atualizar Suporte</title>
6   <style>
7     form {
8       max-width: 400px;
9       margin: 40px auto;
10      padding: 20px;
11      background: #f9f9f9;
12      border-radius: 4px;
13      border: 1px solid #ccc;
14      font-family: Arial, sans-serif;
15    }
16    label {
17      display: block;
18      margin-top: 15px;
19      font-weight: bold;
20    }
21    input[type="text"], textarea, input[type="date"] {
22      width: 100%;
23      padding: 10px;
24      margin-top: 6px;
25      border-radius: 4px;
26      border: 1px solid #aaa;
27      box-sizing: border-box;
28      font-size: 1em;
29    }
30    button {
31      margin-top: 20px;
32      background-color: #007bff;
33      border: none;
34      color: white;
35      padding: 12px 25px;
36      font-weight: bold;
37      border-radius: 4px;
38      cursor: pointer;
39      font-size: 1em;
40    }
41    button:hover {
42      background-color: #0056b3;
43    }
44    .back-link {
45      display: block;
46      margin: 20px auto;
47      text-align: center;
48      text-decoration: none;
49      color: #007bff;
50    }
51    .back-link:hover {
52      text-decoration: underline;
53    }
54  </style>
55 </head>
56 <body>
57   <form action="/while-play/projeto_whileplay/back-end/update-suporte" method="POST">
58     <h2>Atualizar Suporte</h2>
59
60     <!-- Campo oculto para identificar o registro -->
61     <input type="hidden" name="id" value="php htmlspecialchars($suporteInfo['id']) ??" />
62
63     <label for="usuario_id">ID do Usuário:</label>
64     <input type="text" id="usuario_id" name="usuario_id" value="php htmlspecialchars($suporteInfo['usuario_id']) ??" required>
65
66     <label for="mensagem">Mensagem:</label>
67     <textarea id="mensagem" name="mensagem" rows="4" required>php htmlspecialchars($suporteInfo['mensagem']) ?&lt;/textarea&gt;
68
69     &lt;label for="data_envio"&gt;Data de Envio:&lt;/label&gt;
70     &lt;input type="date" id="data_envio" name="data_envio" value="<?php htmlspecialchars($suporteInfo['data_envio']) ??" required>
71
72     <button type="submit">Salvar Alterações</button>
73
74   </form>
75
76   <a href="/while-play/projeto_whileplay/back-end/list-suportes" class="back-link"> Voltar para Lista de Suportes</a>
77
78 </body>
79 </html>
80

```

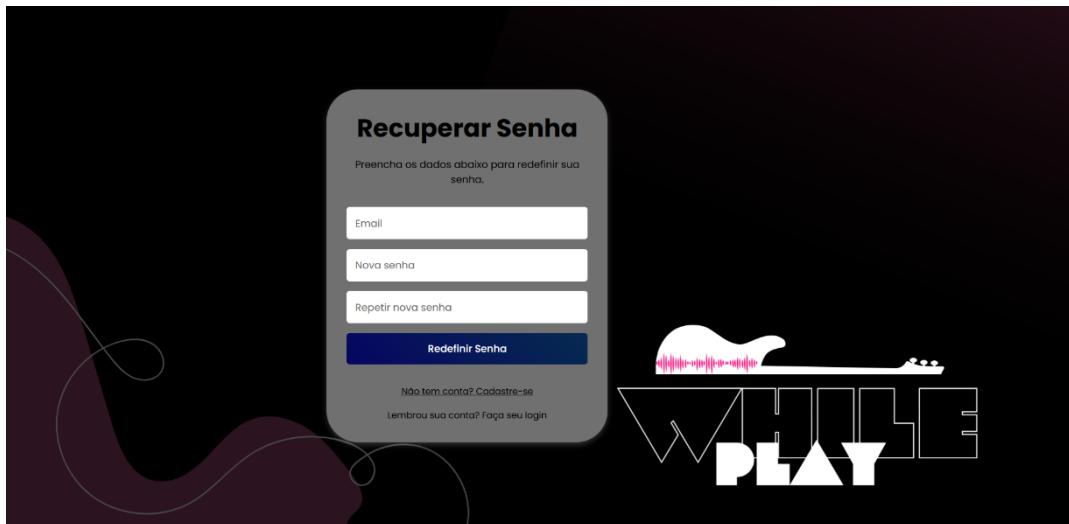
**Update – suporte – form:** Este código HTML/PHP gera um formulário para atualização de registros de suporte no sistema. Ele permite editar os campos `usuario_id`, `mensagem` e `data_envio`, preenchidos automaticamente com os dados atuais usando PHP e a função `htmlspecialchars()` para garantir segurança contra injeções de código. O formulário utiliza o método POST para enviar as alterações para a rota `/update-suporte`, responsável por processar e atualizar os dados no backend.

O layout é simples e funcional, com estilização CSS aplicada para centralizar o formulário e oferecer uma interface clara. O botão de envio possui efeito hover para melhorar a interação, e há também um link na parte inferior que permite retornar à lista de registros de suporte. A estrutura garante que a atualização dos dados seja feita de forma segura, organizada e responsiva para o usuário.

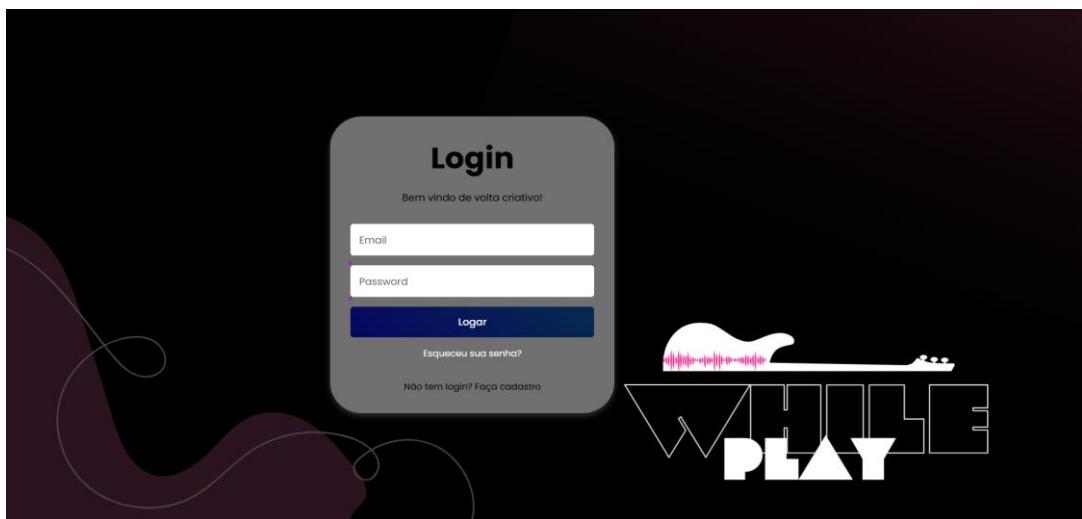
### 13.3 Integração com APIs

- Quais APIs foram usadas
- Como foram integradas no sistema

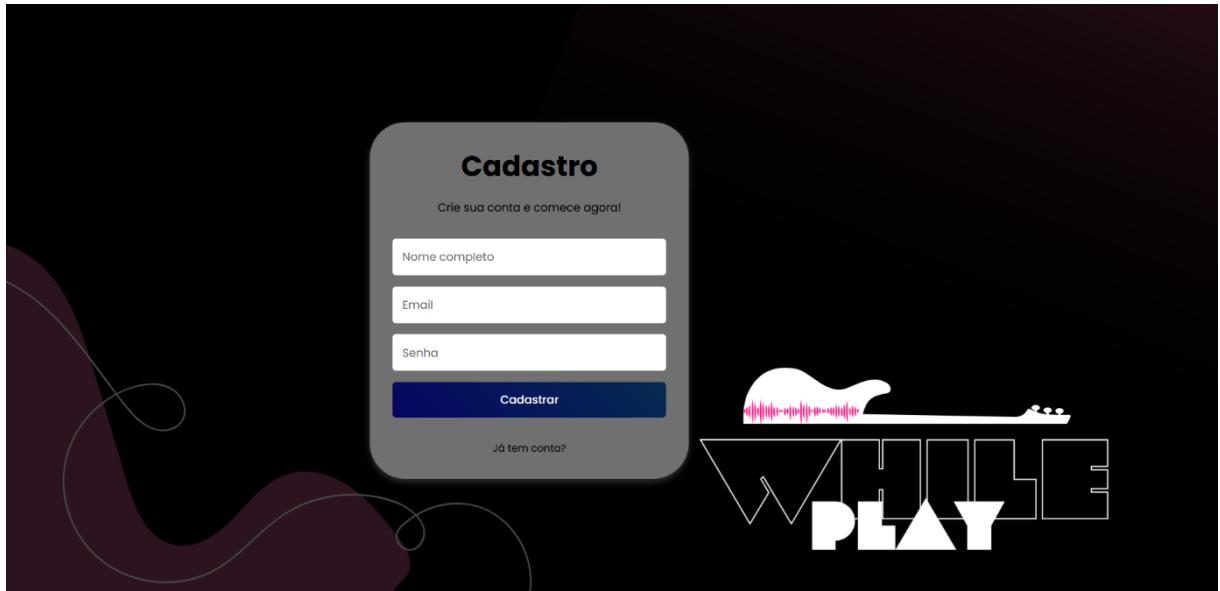
## 14 TELAS EXECUTADAS E FUNCIONALIDADES



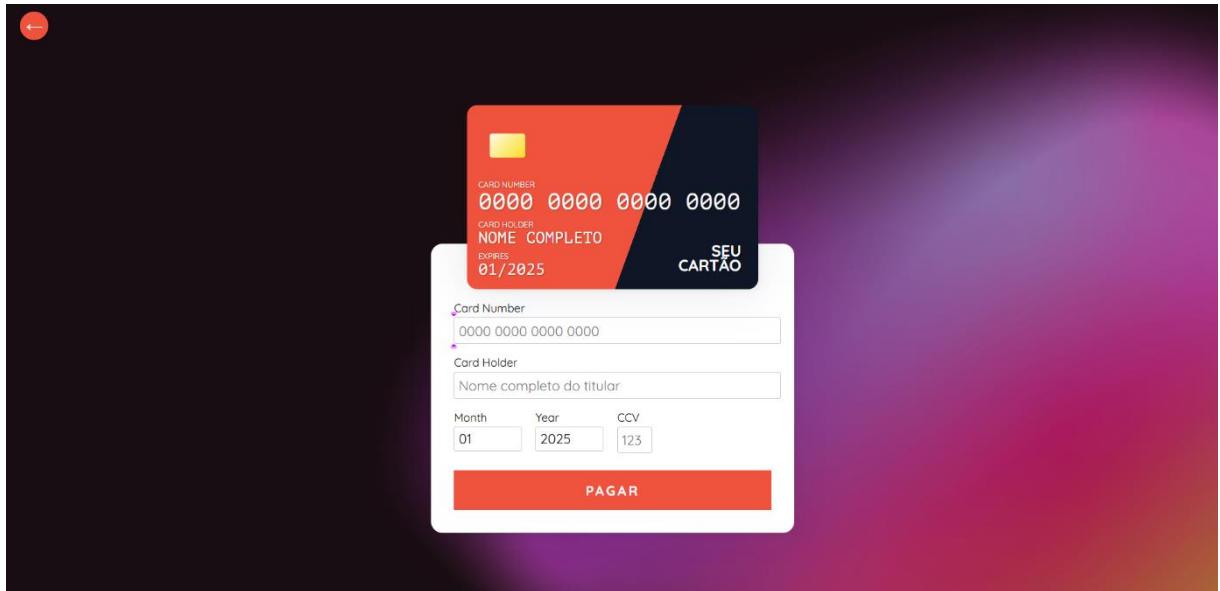
Esse código redefini sua senha, onde deve inserir o email correto seu, e sua nova senha e escrevê-la novamente igual. Se clicar n background você será questionado se quer voltar a página anterior de ir para a de login, ou apenas selecionar as opções abaixo da redefinir senha.



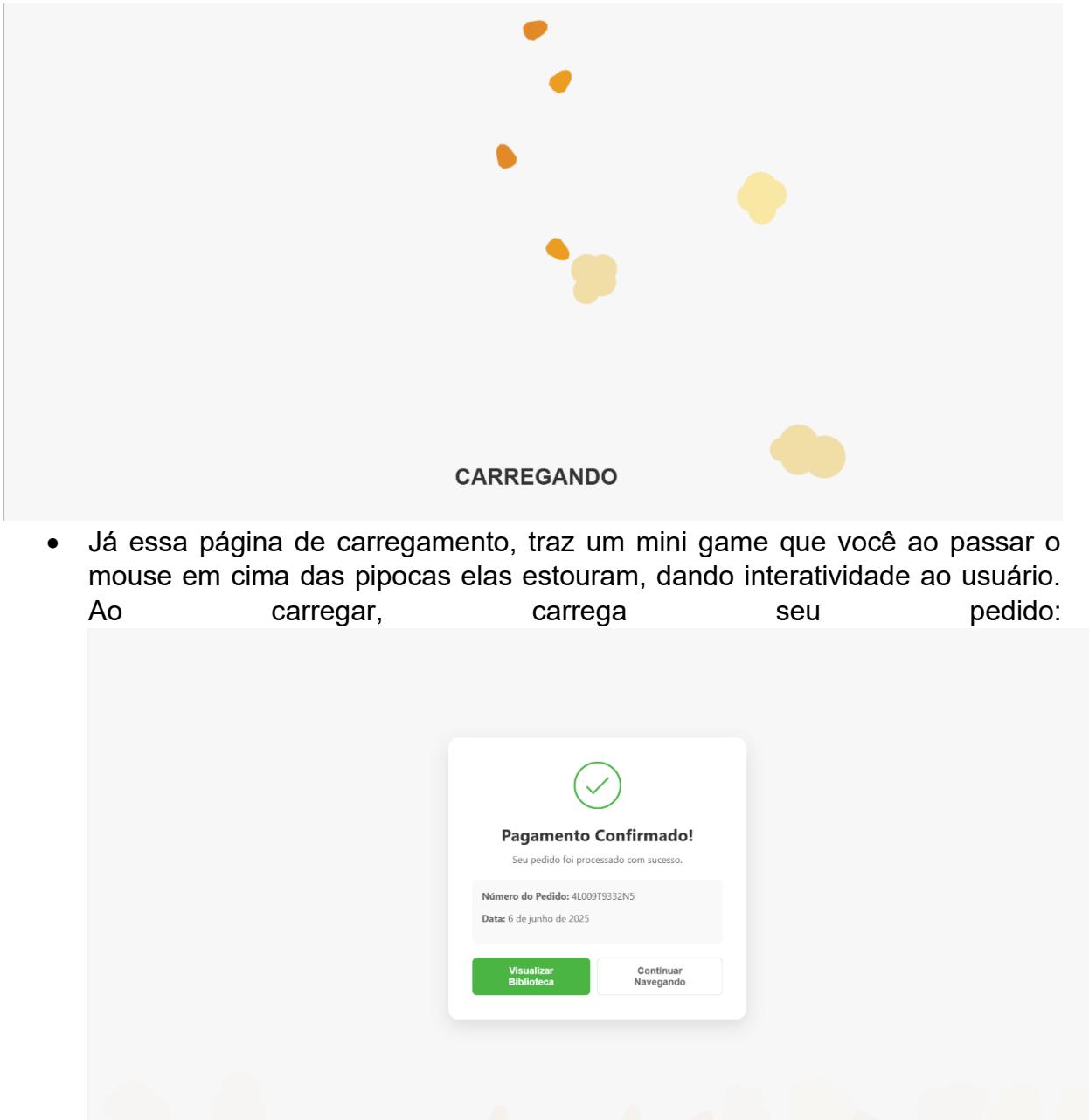
- Seguindo o mesmo padrão visual, segue a lógica onde insere seu email e senha atual, podendo acessar as páginas principais do site. Aqui também temos o background que te envia para a homepage e as opções embaixo do botão logar.



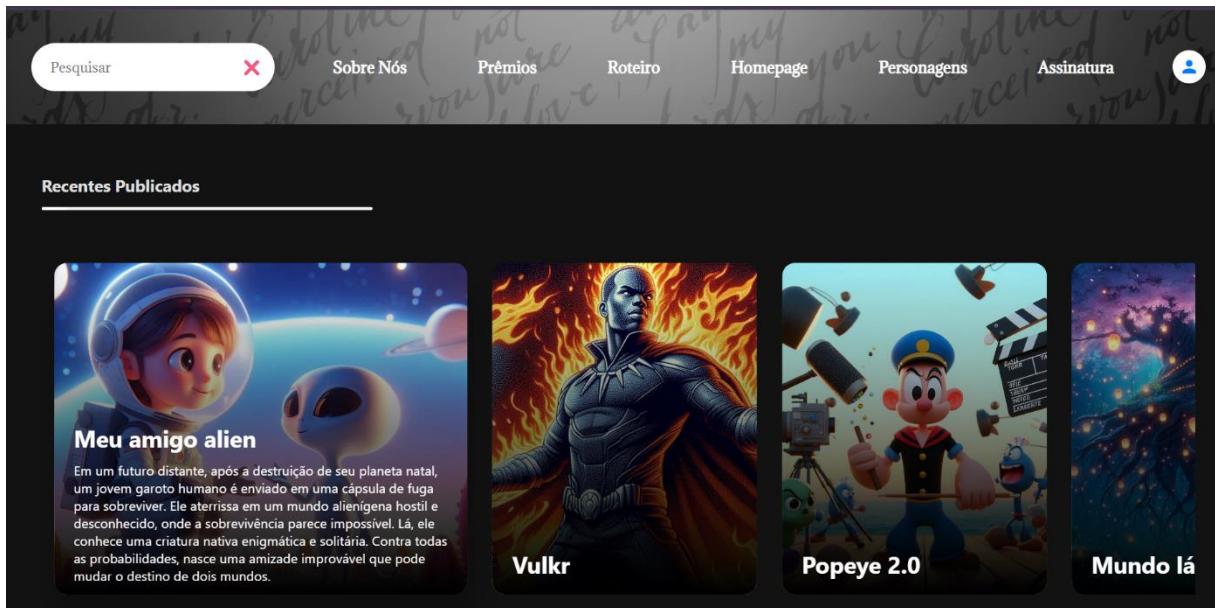
- Como os anteriores, também há a opção de background que volta para a homepage e as opções embaixo do botão, além de inserir nome, email e senha.



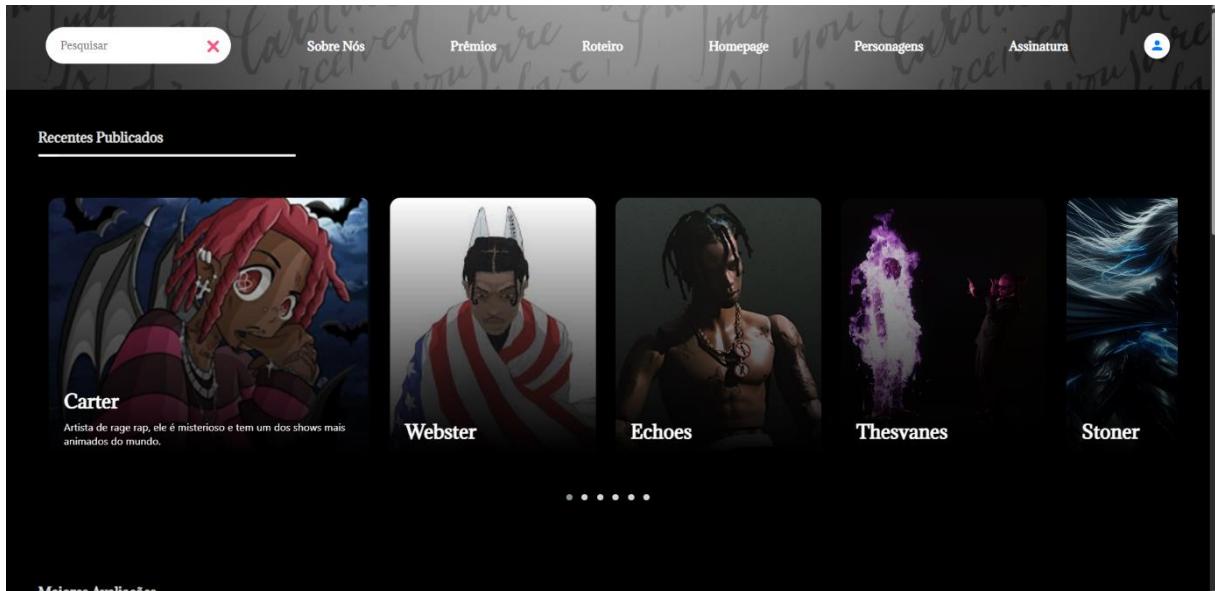
- Já essa é a página de pagamento, onde insere seus dados que preenchem automaticamente no cartão online. Ao clicar na seta, volta e cancela sua compra atua. Quando inserir os dados e clicar o pagamento irá ser direcionado à página de carregamento.



- Já essa página de carregamento, traz um mini game que você ao passar o mouse em cima das pipocas elas estouram, dando interatividade ao usuário. Ao carregar, carrega seu pedido:



- Provavelmente a principal página do site, a página de roteiro traz roteiros de pessoas online que querem vender seu produto. Com os cards, você pode passar o cursor em cima e ler uma breve descrição, assim clicando em cima, você pode baixar o arquivo para ler o roteiro completo e comprar os direitos da obra.



- Seguindo os mesmos padrões da página de roteiro, temos a mesma proposta que o anterior. Ao clicar pode visualizar o projeto e comprar, e pode ler uma breve descrição nos cards.

**Prêmios**

Celebrando os clientes que conquistaram premiações especiais através das compras em nosso site, como roteiros e personagens.

**OSCAR**

Vencedores do maior prêmio de cinema da história utilizando produtos vendidos pelo site.

| MELHOR ROTEIRO ORIGINAL                                                | MELHOR ATOR                                                                                               | MELHOR DIRETOR                                                                                               | MELHOR FILME                                                                  | MELHOR ATOR                                                                               |
|------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <b>Sean S. Baker</b><br>Dirigiu filme premiado comprado em nosso site. | <b>Adrien Brody</b><br>Interpretou Michael (personagem criado e vendido pelo site) em seu último projeto. | <b>Christopher Nolan</b><br>Adaptação recente comprada pelo nosso site e deu a luz ao filme "A Última nota". | <b>Oppenheimer</b><br>Filme premiado que adaptou um roteiro comprado no site. | <b>Colin Murphy</b><br>Premiado com mais de 100 personagens criados por clientes do site. |

- Já essa página traz todos os premiados por diversas premiações ao redor do mundo, como Oscar, Bafta e Globo de ouro. São ganhadores que adaptaram projetos comprados em nosso site e foram vencedores. Traz artistas que ganharam prêmios fictícios para simular. Isso serve apenas para dar veracidade para o site e confiança que nosso projeto pode fazer você dar certo em seu sonho.

**PLANO FREE**

ACESSO AS PÁGINAS  
VISUALIZAR ROTEIROS E PERSONAGENS

**PLANO ATUAL**

**CONTINUAR PLANO**

**PREMIUM**

COMPRAR ROTEIROS E PERSONAGENS  
VISUALIZAR BIBLIOTECA  
TUDO DO PLANO FREE

**49,99**

**PAGAR**

- Quando você faz login, pode acessar as páginas que mostrei anteriormente menos fazer a publicação e a compra dos projetos. Isso se dá por conta que deve assinar a assinatura do nosso site, onde traz confiança para o vendedor e garante que seu projeto vai ser usado. Você poderá ter a experiência completa do site e finalmente trabalhar com isso.

**Publicar Projeto**

Selecione o tipo de arquivo que deseja disponibilizar para venda

Personagem - Imagem (PNG)

Roteiro - Documento (DOCX)

**Título do Projeto**  
Digite o título do seu projeto

**E-mail para contato**  
Digite seu e-mail

**Sinopse**  
Digite a sinopse do seu projeto

**Imagen do Personagem (PNG)**  
Clique para selecionar o arquivo PNG  
Tamanho máximo: 20MB - Formatos aceitos: PNG

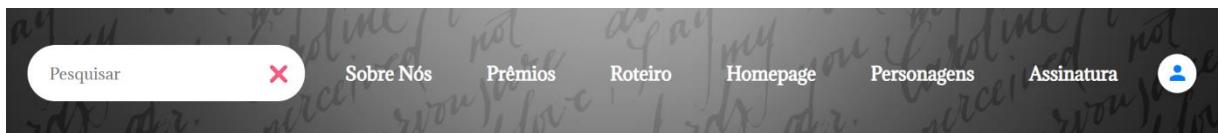
**TERMOS DE COMERCIALIZAÇÃO E VENDA DE DIREITOS**  
Ao submeter esse projeto à plataforma, você ("Autor") concorda com os seguintes termos e condições:

1. **DISPONIBILIZAÇÃO PARA VENDA DE DIREITOS**  
U. O Autor consente expressamente em disponibilizar seu projeto (doravante "Obra") para que terceiros interessados possam adquirir direitos de uso, adaptação, produção ou comercialização mediante compensação financeira.

[Leia os termos](#)

**Publicar e Disponibilizar para Venda**

- Aqui temos a página de publicação de projeto. Ao selecionar personagens traz uma memória máxima de arquivo e troca docx para png. Já ao selecionar roteiro volta para as anteriores medidas e docx. Ali você deve inserir o nome do projeto, email para receber possíveis notificações de vendas, uma breve descrição sobre o seu projeto, selecionar o arquivo na sua máquina e inseri-lo no botão e por fim ler e aceitar os termos do nosso site. Com tudo isso pronto, você pode começar a realmente ver seu esforço sendo respondido.



- Esse é o cabeçalho padrão de login, tendo as opções: sobre nós (traz você a página sobre os criadores do site nosso e a história do projeto), prêmios, roteiros e personagens (páginas que citei anteriormente), homepage (traz você direto para a página principal de nossos projeto), e assinatura (página que dá direito a o usuário trabalhar e ganhar renda com nosso site). Além que ao clicar no ícone de perfil ali, joga você direto ao seu perfil pessoal, podendo editá-lo cria-lo.



- Cabeçalho de pessoas sem login, ao entrar no site sem concluir os passos iniciais. Ele poderá visualizar as páginas sem poder comprar ou baixar nada. Apenas ver as páginas e andar sobre elas.



- Último cabeçalho que podem ter acesso no nosso site, onde quando há a conclusão da assinatura. Nele você tem acesso a biblioteca, onde todos seus arquivos publicados e comprados apareceram lá para você poder visualizar e ter um controle sobre. Além disso, o usuário também pode acessar à área de publicação onde publica o seu projeto.

**15 CONCLUSÃO**

## **REFERÊNCIAS**