<!doctype html>

<html lang="pt-br">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 100 100%22><text y=%22.9em%22 font-size=%2290%22>📚</text></svg>">

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<meta name="description" content="Um estúdio de escrita com IA para criar, gerenciar e escrever livros, capítulo por capítulo." />

<title>Estúdio de Histórias IA</title>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="/index.css">

<script type="importmap">

{

"imports": {

"react": "https://aistudiocdn.com/react@^18.3.1",

"react-dom/": "https://aistudiocdn.com/react-dom@^18.3.1/",

"@google/genai": "https://aistudiocdn.com/@google/genai@^1.24.0",

"react/": "https://aistudiocdn.com/react@^18.3.1/",

"vite": "https://aistudiocdn.com/vite@^7.1.10",

"@vitejs/plugin-react": "https://aistudiocdn.com/@vitejs/plugin-react@^5.0.4",

"@netlify/functions": "https://aistudiocdn.com/@netlify/functions@^5.0.0"

}

}

</script>

</head>

<body>

<div id="root"></div>

<script type="module" src="/index.tsx"></script>

</body>

</html>:root {

--background-dark: #1a1a1a;

--surface-color: #242424;

--surface-light: #2c2c2c;

--primary-accent: #8e44ad;

--primary-accent-hover: #9b59b6;

--secondary-accent: #3498db;

--secondary-accent-hover: #5dade2;

--text-primary: #f0f0f0;

--text-secondary: #b3b3b3;

--border-color: #3a3a3a;

--danger-color: #e74c3c;

--font-family: 'Inter', sans-serif;

--border-radius: 12px;

--shadow: 0 4px 6px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.08);

}

\* {

box-sizing: border-box;

}

html, body {

height: 100%;

margin: 0;

padding: 0;

}

body {

font-family: var(--font-family);

background-color: var(--background-dark);

color: var(--text-primary);

display: flex;

justify-content: center;

padding: 2rem 1rem;

overflow-y: auto;

line-height: 1.6;

}

#root {

width: 100%;

max-width: 900px;

}

.app-container {

display: flex;

flex-direction: column;

gap: 2.5rem;

}

/\* --- Animations --- \*/

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

@keyframes fadeInMessage {

from { opacity: 0; transform: translateY(5px); }

to { opacity: 1; transform: translateY(0); }

}

.fade-in {

animation: fadeIn 0.5s ease-in-out forwards;

}

/\* --- Header --- \*/

header {

text-align: center;

animation: fadeIn 0.5s ease;

}

header h1 {

font-size: 2.8rem;

font-weight: 700;

margin: 0;

color: var(--text-primary);

}

header p {

font-size: 1.2rem;

color: var(--text-secondary);

margin-top: 0.5rem;

max-width: 600px;

margin-left: auto;

margin-right: auto;

}

/\* --- General UI Elements --- \*/

.action-btn {

background-color: var(--primary-accent);

color: #fff;

border: none;

padding: 0.8rem 1.2rem;

font-size: 1rem;

font-weight: 600;

border-radius: var(--border-radius);

cursor: pointer;

transition: background-color 0.2s ease, transform 0.1s ease;

display: inline-flex;

align-items: center;

justify-content: center;

gap: 0.5rem;

text-decoration: none;

}

.action-btn:hover:not(:disabled) {

background-color: var(--primary-accent-hover);

transform: translateY(-2px);

}

.action-btn:disabled {

background-color: #555;

cursor: not-allowed;

opacity: 0.7;

}

.secondary-btn {

background-color: var(--surface-light);

color: var(--text-primary);

border: 1px solid var(--border-color);

}

.secondary-btn:hover:not(:disabled) {

background-color: var(--border-color);

}

.small-btn {

padding: 0.4rem 0.8rem;

font-size: 0.85rem;

}

/\* --- Library View --- \*/

.library-header {

display: flex;

justify-content: space-between;

align-items: center;

flex-wrap: wrap;

gap: 1rem;

margin-bottom: 1.5rem;

}

.library-header h2 {

margin: 0;

}

.library-actions {

display: flex;

gap: 0.75rem;

}

.book-grid {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(180px, 1fr));

gap: 1.5rem;

}

.book-card {

background-color: var(--surface-color);

border-radius: var(--border-radius);

overflow: hidden;

cursor: pointer;

transition: transform 0.2s ease, box-shadow 0.2s ease;

border: 1px solid var(--border-color);

display: flex;

flex-direction: column;

}

.book-card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);

}

.book-card-cover {

width: 100%;

aspect-ratio: 2 / 3;

background-color: var(--background-dark);

display: flex;

align-items: center;

justify-content: center;

color: var(--text-secondary);

font-size: 2.5rem;

}

.book-card-cover img {

width: 100%;

height: 100%;

object-fit: cover;

}

.book-card-title {

padding: 1rem;

font-weight: 600;

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

text-align: center;

}

.no-books {

grid-column: 1 / -1;

text-align: center;

padding: 3rem;

color: var(--text-secondary);

background-color: var(--surface-color);

border-radius: var(--border-radius);

border: 2px dashed var(--border-color);

}

/\* --- Editor View --- \*/

.editor-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.editor-section {

background-color: var(--surface-color);

padding: 1.5rem 2rem;

border-radius: var(--border-radius);

border: 1px solid var(--border-color);

margin-bottom: 2rem;

}

.editor-section h3 {

margin-top: 0;

margin-bottom: 1.5rem;

border-bottom: 1px solid var(--border-color);

padding-bottom: 0.75rem;

}

.form-grid {

display: grid;

grid-template-columns: 1fr;

gap: 1.5rem;

}

@media (min-width: 768px) {

.form-grid {

grid-template-columns: 1fr 1fr;

}

}

.form-group {

display: flex;

flex-direction: column;

gap: 0.5rem;

}

.form-group.full-width {

grid-column: 1 / -1;

}

.form-group label {

font-weight: 500;

color: var(--text-secondary);

}

.form-group > input, .form-group > textarea {

width: 100%;

padding: 0.85rem;

background-color: var(--background-dark);

border: 1px solid var(--border-color);

border-radius: var(--border-radius);

color: var(--text-primary);

font-size: 1rem;

font-family: var(--font-family);

transition: border-color 0.2s, box-shadow 0.2s;

resize: vertical;

}

.form-group > textarea {

min-height: 120px;

}

.form-group > input:focus, .form-group > textarea:focus {

outline: none;

border-color: var(--primary-accent);

box-shadow: 0 0 0 3px rgba(142, 68, 173, 0.5);

}

.cover-art-container {

display: grid;

grid-template-columns: 200px 1fr;

gap: 1.5rem;

align-items: start;

}

.cover-preview {

width: 200px;

aspect-ratio: 2 / 3;

background-color: var(--background-dark);

border-radius: var(--border-radius);

display: flex;

align-items: center;

justify-content: center;

overflow: hidden;

border: 1px solid var(--border-color);

}

.cover-preview img {

width: 100%;

height: 100%;

object-fit: cover;

}

.cover-actions {

display: flex;

flex-direction: column;

gap: 1rem;

}

input[type="file"] {

display: none;

}

.tag-list {

display: flex;

flex-wrap: wrap;

gap: 0.5rem;

padding: 0;

margin: 0;

list-style-type: none;

}

.tag-item {

background-color: var(--surface-light);

color: var(--text-secondary);

padding: 0.25rem 0.75rem;

border-radius: 1rem;

font-size: 0.9rem;

}

.helper-text {

font-size: 0.9rem;

color: var(--text-secondary);

margin-top: 0.75rem;

margin-bottom: 0;

}

/\* --- Character Editor --- \*/

.character-list {

display: flex;

flex-direction: column;

gap: 1rem;

margin-bottom: 1rem;

}

.character-card {

display: flex;

gap: 1rem;

align-items: flex-start;

background-color: var(--surface-light);

padding: 1rem;

border-radius: var(--border-radius);

}

.character-card-inputs {

flex-grow: 1;

display: grid;

grid-template-columns: 1fr 1fr;

gap: 0.75rem;

}

.character-card-inputs input,

.character-card-inputs select,

.character-card-inputs textarea {

width: 100%;

padding: 0.75rem;

background-color: var(--background-dark);

border: 1px solid var(--border-color);

border-radius: var(--border-radius);

color: var(--text-primary);

font-size: 0.95rem;

font-family: var(--font-family);

transition: border-color 0.2s, box-shadow 0.2s;

}

.character-card-inputs input:focus,

.character-card-inputs select:focus,

.character-card-inputs textarea:focus {

outline: none;

border-color: var(--primary-accent);

box-shadow: 0 0 0 3px rgba(142, 68, 173, 0.5);

}

.character-card-inputs textarea {

grid-column: 1 / -1;

min-height: 80px;

resize: vertical;

}

.remove-char-btn {

background: var(--surface-color);

border: 1px solid var(--border-color);

color: var(--text-secondary);

width: 36px;

height: 36px;

border-radius: 50%;

cursor: pointer;

font-size: 1.5rem;

font-weight: bold;

line-height: 32px;

padding: 0;

flex-shrink: 0;

transition: background-color 0.2s, color 0.2s;

display: flex;

align-items: center;

justify-content: center;

}

.remove-char-btn:hover {

background-color: var(--danger-color);

color: #fff;

border-color: var(--danger-color);

}

/\* --- Outline List --- \*/

.outline-list {

list-style-type: none;

padding: 0;

margin: 1.5rem 0 0 0;

display: flex;

flex-direction: column;

gap: 1rem;

}

.outline-item {

background-color: var(--surface-light);

padding: 1rem 1.5rem;

border-radius: var(--border-radius);

border-left: 4px solid var(--secondary-accent);

display: flex;

justify-content: space-between;

align-items: flex-start;

gap: 1rem;

transition: border-left-color 0.3s ease;

}

.outline-item.completed {

border-left-color: var(--primary-accent);

}

.outline-item.completed .outline-content h4::before {

content: '✅';

margin-right: 0.75rem;

}

.outline-content h4 {

margin: 0 0 0.5rem 0;

}

.outline-content p {

margin: 0;

color: var(--text-secondary);

}

.outline-item.is-loading {

justify-content: center;

align-items: center;

min-height: 120px;

padding: 1rem;

border-left-color: var(--secondary-accent-hover);

}

.outline-item.is-loading .loading-container {

padding: 0;

width: 100%;

}

.outline-item.is-loading .loader {

width: 30px;

height: 30px;

border-width: 4px;

margin-bottom: 0.75rem;

}

.outline-item.is-loading .loading-message {

font-size: 0.95rem;

}

.chapter-list {

list-style-type: none;

padding: 0;

margin: 0;

display: flex;

flex-direction: column;

gap: 1.5rem;

}

.chapter-item {

background-color: var(--surface-light);

padding: 1rem 1.5rem;

border-radius: var(--border-radius);

border-left: 4px solid var(--primary-accent);

transition: background-color 0.2s ease;

}

.chapter-item:hover {

background-color: var(--surface-light);

}

.chapter-item-header {

display: flex;

justify-content: space-between;

align-items: center;

}

.chapter-item-header h4 {

margin: 0;

font-weight: 700;

font-size: 1.2rem;

}

.chapter-item-content {

margin-top: 1rem;

padding-top: 1rem;

border-top: 1px solid var(--border-color);

color: var(--text-secondary);

word-wrap: break-word;

}

.empty-chapters {

text-align: center;

color: var(--text-secondary);

padding: 2rem;

border: 2px dashed var(--border-color);

border-radius: var(--border-radius);

}

/\* --- Chapter Editing --- \*/

.chapter-edit-view {

margin-top: 1rem;

padding-top: 1rem;

border-top: 1px solid var(--border-color);

}

.chapter-edit-textarea {

width: 100%;

min-height: 400px; /\* Generous height for editing \*/

padding: 0.85rem;

background-color: var(--background-dark);

border: 1px solid var(--border-color);

border-radius: var(--border-radius);

color: var(--text-primary);

font-size: 1rem;

font-family: var(--font-family);

line-height: 1.6;

resize: vertical;

}

.chapter-edit-textarea:focus {

outline: none;

border-color: var(--primary-accent);

box-shadow: 0 0 0 3px rgba(142, 68, 173, 0.5);

}

.chapter-edit-actions {

margin-top: 1rem;

display: flex;

gap: 0.75rem;

}

/\* --- Reader View --- \*/

.reader-container {

background-color: var(--surface-color);

padding: 2rem 3rem;

border-radius: var(--border-radius);

border: 1px solid var(--border-color);

}

.reader-header {

text-align: center;

margin-bottom: 2rem;

padding-bottom: 1rem;

border-bottom: 1px solid var(--border-color);

}

.reader-cover {

max-width: 250px;

margin: 0 auto 1.5rem auto;

border-radius: var(--border-radius);

display: block;

}

.narrate-btn {

margin-top: 1rem;

}

.reader-content .reader-chapter {

padding: 1rem;

border-radius: var(--border-radius);

margin-bottom: 1.5rem;

transition: background-color 0.3s ease;

}

.current-chapter-highlight {

background-color: rgba(142, 68, 173, 0.15);

}

.chapter-header-with-controls {

display: flex;

justify-content: space-between;

align-items: center;

gap: 1rem;

}

.chapter-header-with-controls h3 {

margin: 0;

color: var(--primary-accent);

}

.audio-controls {

display: flex;

align-items: center;

justify-content: center;

min-width: 36px;

min-height: 36px;

}

.play-pause-btn {

background: var(--surface-light);

color: var(--text-primary);

border: 1px solid var(--border-color);

border-radius: 50%;

width: 36px;

height: 36px;

cursor: pointer;

font-size: 1.2rem;

display: flex;

align-items: center;

justify-content: center;

transition: background-color 0.2s ease;

}

.play-pause-btn:hover {

background-color: var(--primary-accent);

}

.reader-content p {

white-space: pre-wrap;

text-align: justify;

margin-top: 1rem;

margin-bottom: 1rem;

}

/\* --- Loading Spinner --- \*/

.loading-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

gap: 1rem;

padding: 2rem;

color: var(--text-secondary);

}

.loading-container .loading-message {

animation: fadeInMessage 0.5s ease-in-out forwards;

text-align: center;

}

.loader {

border: 5px solid var(--surface-light);

border-top: 5px solid var(--primary-accent);

border-radius: 50%;

width: 50px;

height: 50px;

animation: spin 1s linear infinite;

}

.loader-small {

border: 3px solid var(--surface-light);

border-top: 3px solid var(--primary-accent);

border-radius: 50%;

width: 24px;

height: 24px;

animation: spin 0.8s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

.sr-only {

position: absolute;

width: 1px;

height: 1px;

padding: 0;

margin: -1px;

overflow: hidden;

clip: rect(0, 0, 0, 0);

white-space: nowrap;

border-width: 0;

}

.error-message {

color: var(--danger-color);

background-color: rgba(231, 76, 60, 0.1);

border: 1px solid var(--danger-color);

padding: 1rem;

border-radius: var(--border-radius);

text-align: center;

}

/\* --- API Key Error View --- \*/

.api-key-error-container {

background-color: var(--surface-color);

border: 1px solid var(--danger-color);

border-radius: var(--border-radius);

padding: 2rem 3rem;

margin: 2rem auto 0 auto;

max-width: 700px;

text-align: left;

}

.api-key-error-container h2 {

color: var(--text-primary);

margin-top: 0;

text-align: center;

}

.api-key-instructions {

background-color: var(--surface-light);

border-radius: var(--border-radius);

padding: 1.5rem;

margin-top: 1.5rem;

text-align: left;

}

.api-key-instructions h4 {

margin-top: 0;

margin-bottom: 1rem;

border-bottom: 1px solid var(--border-color);

padding-bottom: 0.5rem;

}

.api-key-instructions ol {

padding-left: 20px;

margin: 0;

}

.api-key-instructions li {

margin-bottom: 0.75rem;

}

.api-key-instructions code {

background-color: var(--background-dark);

color: var(--primary-accent);

padding: 0.2em 0.4em;

margin: 0 0.2em;

font-size: 0.9em;

border-radius: 4px;

}

.api-key-instructions a {

color: var(--secondary-accent);

text-decoration: none;

}

.api-key-instructions a:hover {

text-decoration: underline;

}

.error-footer {

margin-top: 2rem;

color: var(--text-secondary);

font-size: 0.9rem;

text-align: center;

}

.server-error-box {

background-color: rgba(231, 76, 60, 0.1);

border: 1px solid var(--danger-color);

padding: 1rem;

border-radius: var(--border-radius);

margin: 1.5rem 0;

text-align: left;

}

.server-error-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 0.75rem;

}

.server-error-box strong {

color: var(--danger-color);

display: block;

}

.server-error-box p {

margin: 0;

word-break: break-all;

}

.server-error-box code {

background-color: transparent;

color: var(--text-primary);

padding: 0;

}

.copy-btn {

background-color: var(--surface-light);

border: 1px solid var(--border-color);

color: var(--text-secondary);

padding: 0.25rem 0.6rem;

font-size: 0.8rem;

border-radius: 6px;

cursor: pointer;

transition: background-color 0.2s, color 0.2s;

}

.copy-btn:hover {

background-color: var(--secondary-accent);

color: #fff;

}

.troubleshooting-section {

background-color: var(--surface-light);

border-radius: var(--border-radius);

padding: 1.5rem;

margin-top: 2rem;

border-left: 4px solid var(--secondary-accent);

}

.troubleshooting-section h4 {

margin-top: 0;

margin-bottom: 1rem;

color: var(--secondary-accent);

}

.troubleshooting-section ul {

padding-left: 20px;

margin: 0;

list-style-type: '❓ ';

}

.troubleshooting-section li {

margin-bottom: 0.75rem;

}

.troubleshooting-section code {

background-color: var(--background-dark);

color: var(--primary-accent);

padding: 0.2em 0.4em;

margin: 0 0.2em;

font-size: 0.9em;

border-radius: 4px;

}import React, { useState, useRef, useEffect, useCallback } from 'react';

import { createRoot } from 'react-dom/client';

// --- INTERFACES ---

interface Character {

id: string;

name: string;

description: string;

voice: string;

}

interface ChapterOutline {

id: string;

title: string;

plan: string;

completed: boolean;

}

interface Chapter {

id: string; // Should match a ChapterOutline id

title: string;

content: string;

}

interface Book {

id: string;

title: string;

characters: Character[];

setting: string;

outline: ChapterOutline[];

coverImageUrl: string;

synopsis: string;

tags: string[];

chapters: Chapter[];

}

// Enum for schema types, since the original is no longer imported

const GenAIType = {

STRING: 'STRING',

NUMBER: 'NUMBER',

INTEGER: 'INTEGER',

BOOLEAN: 'BOOLEAN',

ARRAY: 'ARRAY',

OBJECT: 'OBJECT',

};

const AVAILABLE\_VOICES = ['Kore', 'Puck', 'Charon', 'Fenrir', 'Zephyr'];

// --- AUDIO HELPERS (Based on Gemini API guidelines) ---

function decode(base64: string): Uint8Array {

const binaryString = atob(base64);

const len = binaryString.length;

const bytes = new Uint8Array(len);

for (let i = 0; i < len; i++) {

bytes[i] = binaryString.charCodeAt(i);

}

return bytes;

}

async function decodeAudioData(

data: Uint8Array,

ctx: AudioContext,

sampleRate: number,

numChannels: number,

): Promise<AudioBuffer> {

const dataInt16 = new Int16Array(data.buffer);

const frameCount = dataInt16.length / numChannels;

const buffer = ctx.createBuffer(numChannels, frameCount, sampleRate);

for (let channel = 0; channel < numChannels; channel++) {

const channelData = buffer.getChannelData(channel);

for (let i = 0; i < frameCount; i++) {

channelData[i] = dataInt16[i \* numChannels + channel] / 32768.0;

}

}

return buffer;

}

// --- HELPER & UI COMPONENTS ---

const LoadingSpinner = ({ message, messages }: { message: string; messages?: string[] }) => {

const [displayedMessage, setDisplayedMessage] = useState(message);

// Use the message as a key to force re-render/re-animation of the paragraph

const key = displayedMessage;

useEffect(() => {

if (messages && messages.length > 0) {

let index = 0;

setDisplayedMessage(messages[0]);

const intervalId = setInterval(() => {

index = (index + 1) % messages.length;

setDisplayedMessage(messages[index]);

}, 3500); // Change message every 3.5 seconds

return () => clearInterval(intervalId);

} else {

setDisplayedMessage(message);

}

}, [messages, message]);

return (

<div className="loading-container fade-in">

<div className="loader" role="status"><span className="sr-only">Gerando...</span></div>

<p key={key} className="loading-message">{displayedMessage}</p>

</div>

);

};

const ApiKeyErrorView = ({ serverError }: { serverError?: string | null }) => {

const [copied, setCopied] = useState(false);

const handleCopy = () => {

if (serverError) {

navigator.clipboard.writeText(serverError);

setCopied(true);

setTimeout(() => setCopied(false), 2000); // Reset after 2 seconds

}

};

return (

<div className="api-key-error-container fade-in">

<h2><span role="img" aria-label="lock">🔑</span> Quase lá! Só falta conectar a sua chave de API.</h2>

{serverError && (

<div className="server-error-box">

<div className="server-error-header">

<strong>Diagnóstico do Servidor:</strong>

<button onClick={handleCopy} className="copy-btn">

{copied ? 'Copiado!' : 'Copiar'}

</button>

</div>

<p><code>{serverError}</code></p>

</div>

)}

<p>

Não se preocupe, este é o passo final e mais comum da configuração. O servidor confirmou que só precisa da chave para desbloquear a magia da IA.

</p>

<div className="api-key-instructions">

<h4>Guia de Configuração Rápida na Netlify</h4>

<ol>

<li>Obtenha sua chave de API no <a href="https://makersuite.google.com/app/apikey" target="\_blank" rel="noopener noreferrer">Google AI Studio</a>.</li>

<li>Vá para o painel do seu site na <a href="https://app.netlify.com/" target="\_blank" rel="noopener noreferrer">Netlify</a>.</li>

<li>Navegue para: <strong>Site configuration</strong> &rarr; <strong>Build & deploy</strong> &rarr; <strong>Environment</strong>.</li>

<li>Clique em <strong>New variable</strong> e insira <strong>exatamente</strong>:

<ul>

<li><strong>Key:</strong> <code>API\_KEY</code> (maiúsculas e com underscore)</li>

<li><strong>Value:</strong> Cole a chave que você copiou do Google.</li>

</ul>

</li>

<li><strong>O PASSO MAIS IMPORTANTE:</strong> Vá para a aba <strong>Deploys</strong> e clique em <strong>"Trigger deploy" &rarr; "Deploy site"</strong>. O aplicativo só lerá a nova chave após um novo deploy.</li>

</ol>

</div>

<div className="troubleshooting-section">

<h4>Ainda não funcionou? Vamos verificar os pontos comuns:</h4>

<ul>

<li><strong>O Deploy foi feito?</strong> A alteração só tem efeito <strong>após</strong> um novo deploy ser concluído com sucesso. Recarregar a página não é suficiente.</li>

<li><strong>Erro de digitação na Chave?</strong> A "Key" deve ser <code>API\_KEY</code>, exatamente assim. Nem <code>api\_key</code>, nem <code>API KEY</code>.</li>

<li><strong>A Chave foi colada corretamente?</strong> Às vezes, um espaço extra no início ou no fim pode ser copiado. Verifique se o valor colado está limpo.</li>

<li><strong>Múltiplos Sites?</strong> Se você tiver mais de um site na Netlify, confirme se está adicionando a variável ao site correto.</li>

</ul>

</div>

<p className="error-footer">

Assim que a chave estiver no lugar e o novo deploy for concluído, esta tela desaparecerá e você poderá começar a criar!

</p>

</div>

);

};

// FIX: Added `style` prop to allow inline styling and resolve TypeScript errors.

const ActionButton: React.FC<{

onClick?: (e?: React.MouseEvent) => void;

disabled?: boolean;

children: React.ReactNode;

className?: string;

title?: string;

as?: 'button' | 'label';

htmlFor?: string;

style?: React.CSSProperties;

}> = ({ onClick, disabled, children, className = '', title, as = 'button', htmlFor, style }) => {

const Component = as;

return (

<Component

onClick={onClick}

disabled={disabled}

className={`action-btn ${className}`}

title={title}

htmlFor={htmlFor}

style={style}

>

{children}

</Component>

);

};

// --- MAIN VIEWS ---

const LibraryView = ({ books, setBooks, onSelectBook }: {

books: Book[];

setBooks: (books: Book[]) => void;

onSelectBook: (bookId: string) => void;

}) => {

const fileInputRef = useRef<HTMLInputElement>(null);

const handleCreateNewBook = () => {

const newBook: Book = {

id: Date.now().toString(),

title: 'Novo Livro Sem Título',

characters: [],

setting: '',

outline: [],

coverImageUrl: '',

synopsis: '',

tags: [],

chapters: [],

};

setBooks([newBook, ...books]);

onSelectBook(newBook.id);

};

const handleExport = () => {

const jsonString = JSON.stringify(books, null, 2);

const blob = new Blob([jsonString], { type: 'application/json' });

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = 'biblioteca\_ia\_historias.json';

a.click();

URL.revokeObjectURL(url);

};

const handleImportClick = () => {

fileInputRef.current?.click();

};

const handleImportFile = (event: React.ChangeEvent<HTMLInputElement>) => {

const file = event.target.files?.[0];

if (file) {

const reader = new FileReader();

reader.onload = (e) => {

try {

const importedBooks = JSON.parse(e.target?.result as string);

// Basic validation

if (Array.isArray(importedBooks) && importedBooks.every(b => b.id && b.title)) {

setBooks(importedBooks);

} else {

alert('Arquivo de importação inválido.');

}

} catch (error) {

alert('Erro ao ler o arquivo de importação.');

console.error("Import error:", error);

}

};

reader.readAsText(file);

}

};

return (

<div className="fade-in">

<div className="library-header">

<h2>Minha Biblioteca</h2>

<div className="library-actions">

<input type="file" ref={fileInputRef} onChange={handleImportFile} accept="application/json" style={{ display: 'none' }}/>

<ActionButton onClick={handleImportClick} className="secondary-btn">Importar</ActionButton>

<ActionButton onClick={handleExport} className="secondary-btn">Exportar</ActionButton>

<ActionButton onClick={handleCreateNewBook}>+ Criar Novo Livro</ActionButton>

</div>

</div>

{books.length > 0 ? (

<div className="book-grid">

{books.map(book => (

<div key={book.id} className="book-card" onClick={() => onSelectBook(book.id)}>

<div className="book-card-cover">

{book.coverImageUrl ? <img src={book.coverImageUrl} alt={`Capa de ${book.title}`} /> : '📚'}

</div>

<div className="book-card-title">{book.title}</div>

</div>

))}

</div>

) : (

<div className="no-books">

<h3>Sua biblioteca está vazia.</h3>

<p>Clique em "Criar Novo Livro" para começar sua primeira história!</p>

</div>

)}

</div>

);

};

const BookEditorView = ({ book, setBooks, onBack, onRead }: {

book: Book;

setBooks: React.Dispatch<React.SetStateAction<Book[]>>;

onBack: () => void;

onRead: () => void;

}) => {

const [loadingStates, setLoadingStates] = useState({

cover: false,

synopsis: false,

outline: false,

chapter: '', // Now holds the ID of the chapter being generated

});

const [error, setError] = useState<string | null>(null);

const [expandedChapterId, setExpandedChapterId] = useState<string | null>(null);

const [editingChapterId, setEditingChapterId] = useState<string | null>(null);

const [editedContent, setEditedContent] = useState('');

const outlineLoadingMessages = [

"Analisando o conceito do seu livro...",

"Esboçando os arcos narrativos principais...",

"Criando pontos de virada emocionantes...",

"Definindo o clímax e a resolução...",

"Estruturando os capítulos para máximo impacto...",

];

const chapterLoadingMessages = [

"A IA está mergulhando no seu universo...",

"Escrevendo o primeiro rascunho do capítulo...",

"Adicionando descrições sensoriais e detalhes vívidos...",

"Desenvolvendo diálogos e pensamentos internos dos personagens...",

"Expandindo a narrativa para atingir a profundidade desejada...",

"Revisando a coesão e o ritmo do capítulo... Quase pronto!",

];

const handleUpdateBook = (updatedFields: Partial<Book>) => {

setBooks(prevBooks =>

prevBooks.map(b => b.id === book.id ? { ...b, ...updatedFields } : b)

);

};

const handleGenerate = async (

type: 'cover' | 'synopsis' | 'outline' | 'chapter',

payload: any,

chapterOutlineId?: string

) => {

setError(null);

const loadingKey = type === 'chapter' ? 'chapter' : type;

setLoadingStates(prev => ({ ...prev, [loadingKey]: chapterOutlineId || true }));

try {

const response = await fetch('/.netlify/functions/gemini', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ type, payload }),

});

if (!response.ok) {

const errorData = await response.json().catch(() => ({ error: `HTTP error! status: ${response.status}` }));

throw new Error(errorData.error || `HTTP error! status: ${response.status}`);

}

if (type === 'cover') {

const { base64Image } = await response.json();

handleUpdateBook({ coverImageUrl: `data:image/png;base64,${base64Image}` });

} else {

const resultText = await response.text();

const parsed = JSON.parse(resultText);

if (type === 'synopsis') {

handleUpdateBook({ synopsis: parsed.synopsis, tags: parsed.tags });

} else if (type === 'outline') {

const newOutline: ChapterOutline[] = parsed.map((item: any, index: number) => ({

id: `${Date.now()}-${index}`,

title: item.title,

plan: item.plan,

completed: false,

}));

handleUpdateBook({ outline: newOutline });

} else if (type === 'chapter' && chapterOutlineId) {

const newChapter: Chapter = { id: chapterOutlineId, title: parsed.title, content: parsed.content };

const updatedOutline = book.outline.map(item => item.id === chapterOutlineId ? {...item, completed: true} : item);

handleUpdateBook({ chapters: [...book.chapters.filter(c => c.id !== chapterOutlineId), newChapter], outline: updatedOutline });

}

}

} catch (err: any) {

console.error(`Error generating ${type}:`, err);

setError(err.message || `Ocorreu um erro ao gerar ${type}. Tente novamente.`);

} finally {

setLoadingStates(prev => ({ ...prev, [loadingKey]: type === 'chapter' ? '' : false }));

}

};

const characterPromptInfo = book.characters.map(c => c.name).join(', ');

const characterDetailsForPrompt = book.characters.map(c => `${c.name}: ${c.description}`).join('; ');

const generateCover = () => handleGenerate('cover', { prompt: `Capa de livro fotorrealista, imagem de alta resolução com qualidade de fotografia para uma história com o título "${book.title}". A imagem deve parecer uma cena de um filme, com iluminação dramática e detalhes nítidos. Personagens principais: ${characterPromptInfo}. Cenário: ${book.setting}. Evite qualquer estilo de ilustração, desenho animado ou arte digital óbvia. O objetivo é o realismo absoluto.` });

const generateSynopsis = () => handleGenerate('synopsis', { prompt: `Baseado no conceito: Título: "${book.title}", Personagens: "${characterDetailsForPrompt}", Cenário: "${book.setting}", crie uma sinopse de história envolvente (cerca de 150 palavras) e uma lista de exatamente 5 tags relevantes.`, schema: { type: GenAIType.OBJECT, properties: { synopsis: { type: GenAIType.STRING }, tags: { type: GenAIType.ARRAY, items: { type: GenAIType.STRING } } }, required: ["synopsis", "tags"] } });

const generateOutline = () => handleGenerate('outline', { prompt: `Crie um esboço de história detalhado e expansivo para uma saga épica intitulada "${book.title}". Personagens principais: ${characterPromptInfo}. Cenário: ${book.setting}. REQUISITO CRÍTICO: O esboço DEVE conter um MÍNIMO de 25 capítulos. Cada capítulo será posteriormente expandido para mais de 4.000 palavras, então o plano para cada um deve ser substancial o suficiente para suportar tal profundidade. Para cada capítulo, forneça um título cativante e um "plano" detalhado descrevendo os eventos principais, desenvolvimento de personagens e pontos da trama.`, schema: { type: GenAIType.ARRAY, items: { type: GenAIType.OBJECT, properties: { title: { type: GenAIType.STRING }, plan: { type: GenAIType.STRING } } } } });

const generateChapter = (outlineItem: ChapterOutline) => {

const characterListForPrompt = book.characters.map(c => c.name).join(', ');

const firstCharName = book.characters[0]?.name || 'Personagem1';

const prompt = `INSTRUÇÃO CRÍTICA E NÃO NEGOCIÁVEL: Escreva um capítulo de romance completo, denso e extremamente detalhado com um MÍNIMO ABSOLUTO de 4.000 palavras.

REQUISITOS DE FORMATAÇÃO PARA ÁUDIO MULTI-FALANTE:

1. Todo o texto narrativo (descrições, pensamentos internos, etc.) DEVE ser prefixado com "Narrador: ".

2. Todo diálogo DEVE ser prefixado com o nome do personagem falando, seguido por dois pontos. Exemplo: "${firstCharName}: Olá, mundo."

3. NÃO inclua o nome do personagem na mesma linha após o prefixo.

CORRETO:

Narrador: O sol se punha no horizonte.

${firstCharName}: Você viu aquilo?

INCORRETO:

O sol se punha. ${firstCharName} disse, "Você viu aquilo?"

DETALHES DO CAPÍTULO:

- Título da Saga: "${book.title}"

- Título do Capítulo: "${outlineItem.title}"

- Plano do Capítulo: "${outlineItem.plan}"

- Personagens Principais Envolvidos: "${characterListForPrompt}"

- Cenário: "${book.setting}"

PROCESSO DE ESCRITA ITERATIVO OBRIGATÓRIO:

1. Escreva um primeiro rascunho.

2. Revise e expanda, adicionando mais profundidade, descrições sensoriais, diálogo interno dos personagens, subtramas e interações detalhadas, sempre seguindo as regras de formatação.

3. Continue este processo de expansão até que a contagem de palavras exceda 4.000 palavras.

PROIBIDO: Não use repetição ou preenchimento. O conteúdo deve ser conciso e envolvente. Apenas retorne o conteúdo final do capítulo, sem o título.`;

handleGenerate('chapter', { prompt, schema: { type: GenAIType.OBJECT, properties: { title: { type: GenAIType.STRING }, content: { type: GenAIType.STRING } }, required: ["title", "content"] } }, outlineItem.id);

};

const handleCoverUpload = (event: React.ChangeEvent<HTMLInputElement>) => {

const file = event.target.files?.[0];

if (file) {

const reader = new FileReader();

reader.onload = (e) => {

handleUpdateBook({ coverImageUrl: e.target?.result as string });

};

reader.readAsDataURL(file);

}

};

const handleStartEdit = (chapter: Chapter) => {

setEditingChapterId(chapter.id);

setEditedContent(chapter.content);

};

const handleCancelEdit = () => {

setEditingChapterId(null);

setEditedContent('');

};

const handleSaveEdit = (chapterId: string) => {

const updatedChapters = book.chapters.map(ch =>

ch.id === chapterId ? { ...ch, content: editedContent } : ch

);

handleUpdateBook({ chapters: updatedChapters });

handleCancelEdit();

};

const handleAddCharacter = () => {

const newChar: Character = {

id: `char-${Date.now()}`,

name: '',

description: '',

voice: 'Kore',

};

handleUpdateBook({ characters: [...book.characters, newChar] });

};

const handleUpdateCharacter = (id: string, field: keyof Omit<Character, 'id'>, value: string) => {

const updatedCharacters = book.characters.map(char =>

char.id === id ? { ...char, [field]: value } : char

);

handleUpdateBook({ characters: updatedCharacters });

};

const handleRemoveCharacter = (id: string) => {

const characterToRemove = book.characters.find(char => char.id === id);

const characterName = characterToRemove?.name || 'este personagem';

if (window.confirm(`Tem certeza de que deseja remover ${characterName}? Esta ação não pode ser desfeita.`)) {

handleUpdateBook({ characters: book.characters.filter(char => char.id !== id) });

}

};

return (

<div className="fade-in">

<div className="editor-header">

<ActionButton onClick={onBack} className="secondary-btn">← Voltar para a Biblioteca</ActionButton>

<ActionButton onClick={onRead} disabled={book.chapters.length === 0}>Ler Livro</ActionButton>

</div>

{error && <div className="error-message">{error}</div>}

{/\* --- Section 1: Book Concept --- \*/}

<div className="editor-section">

<h3>1. Conceito do Livro</h3>

<div className="form-grid">

<div className="form-group full-width">

<label htmlFor="bookTitle">Título do Livro</label>

<input id="bookTitle" type="text" value={book.title} onChange={e => handleUpdateBook({ title: e.target.value })} placeholder="Ex: A Sombra da Montanha de Cristal"/>

</div>

<div className="form-group">

<label>Personagens e Vozes</label>

<div className="character-list">

{book.characters.map(char => (

<div key={char.id} className="character-card">

<div className="character-card-inputs">

<input

type="text"

placeholder="Nome do Personagem"

value={char.name}

onChange={e => handleUpdateCharacter(char.id, 'name', e.target.value)}

/>

<select value={char.voice} onChange={e => handleUpdateCharacter(char.id, 'voice', e.target.value)}>

{AVAILABLE\_VOICES.map(v => <option key={v} value={v}>{v}</option>)}

</select>

<textarea

placeholder="Descrição (traços, motivações, etc.)"

value={char.description}

onChange={e => handleUpdateCharacter(char.id, 'description', e.target.value)}

/>

</div>

<button onClick={() => handleRemoveCharacter(char.id)} className="remove-char-btn" title="Remover Personagem">&times;</button>

</div>

))}

</div>

<ActionButton onClick={handleAddCharacter} className="secondary-btn small-btn" style={{alignSelf: 'flex-start'}}>

+ Adicionar Personagem

</ActionButton>

</div>

<div className="form-group">

<label htmlFor="bookSetting">Cenário</label>

<textarea id="bookSetting" value={book.setting} onChange={e => handleUpdateBook({ setting: e.target.value })} placeholder="Ex: O reino de Eldoria, uma terra de picos imponentes, florestas antigas e cidades flutuantes."/>

</div>

</div>

</div>

{/\* --- Section 2: Cover & Synopsis --- \*/}

<div className="editor-section">

<h3>2. Capa e Sinopse</h3>

<div className="cover-art-container">

<div className="cover-preview">

{loadingStates.cover ? <LoadingSpinner message="Criando capa..."/> :

book.coverImageUrl ? <img src={book.coverImageUrl} alt="Capa do livro"/> : '🎨'

}

</div>

<div className="cover-actions">

<ActionButton onClick={generateCover} disabled={!book.title || loadingStates.cover}>

{loadingStates.cover ? 'Gerando...' : 'Gerar Capa com IA'}

</ActionButton>

<ActionButton as="label" htmlFor="cover-upload" className="secondary-btn">Carregar Imagem</ActionButton>

<input id="cover-upload" type="file" accept="image/\*" onChange={handleCoverUpload} />

</div>

</div>

<hr style={{border: 0, borderTop: `1px solid ${'var(--border-color)'}`, margin: '2rem 0'}}/>

<div className="form-group full-width">

<label htmlFor="synopsis">Sinopse & Tags</label>

{loadingStates.synopsis ? <LoadingSpinner message="Criando sinopse..."/> :

<>

<textarea id="synopsis" value={book.synopsis} onChange={e => handleUpdateBook({ synopsis: e.target.value })} placeholder="A sinopse da sua história aparecerá aqui."/>

<ul className="tag-list">

{book.tags.map(tag => <li key={tag} className="tag-item">{tag}</li>)}

</ul>

</>

}

<ActionButton onClick={generateSynopsis} disabled={!book.title || loadingStates.synopsis} className="small-btn" style={{alignSelf: 'flex-start'}}>

{loadingStates.synopsis ? 'Gerando...' : 'Gerar Sinopse e Tags com IA'}

</ActionButton>

</div>

</div>

{/\* --- Section 3: Story Outline --- \*/}

<div className="editor-section">

<h3>3. Esboço da História</h3>

<ActionButton onClick={generateOutline} disabled={!book.title || loadingStates.outline}>

{loadingStates.outline ? 'Gerando...' : 'Gerar Esboço com IA'}

</ActionButton>

<p className="helper-text">A IA irá gerar um esboço de pelo menos 25 capítulos, cada um projetado para ser expandido para mais de 4.000 palavras.</p>

{loadingStates.outline && <LoadingSpinner message="Construindo a estrutura da sua história..." messages={outlineLoadingMessages} />}

{book.outline.length > 0 && (

<ul className="outline-list">

{book.outline.map(item => (

<li key={item.id} className={`outline-item ${item.completed ? 'completed' : ''} ${loadingStates.chapter === item.id ? 'is-loading' : ''}`}>

{loadingStates.chapter === item.id ? (

<LoadingSpinner message={chapterLoadingMessages[0]} messages={chapterLoadingMessages} />

) : (

<>

<div className="outline-content">

<h4>{item.title}</h4>

<p>{item.plan}</p>

</div>

<ActionButton

onClick={() => generateChapter(item)}

disabled={!!loadingStates.chapter || item.completed}

className="small-btn"

title={item.completed ? "Capítulo já escrito" : "Escrever este capítulo"}

>

{item.completed ? 'Concluído' : 'Escrever'}

</ActionButton>

</>

)}

</li>

))}

</ul>

)}

</div>

{/\* --- Section 4: Chapters --- \*/}

<div className="editor-section">

<h3>4. Capítulos</h3>

{book.chapters.length > 0 ? (

<ul className="chapter-list">

{book.outline

.filter(outlineItem => book.chapters.some(ch => ch.id === outlineItem.id))

.map(outlineItem => {

const chapter = book.chapters.find(ch => ch.id === outlineItem.id)!;

return (

<li key={chapter.id} className="chapter-item">

<div className="chapter-item-header" onClick={() => setExpandedChapterId(expandedChapterId === chapter.id ? null : chapter.id)} style={{cursor: 'pointer'}}>

<h4>{chapter.title}</h4>

<span>{expandedChapterId === chapter.id ? '▲' : '▼'}</span>

</div>

{expandedChapterId === chapter.id && (

<>

{editingChapterId === chapter.id ? (

<div className="chapter-edit-view">

<textarea

className="chapter-edit-textarea"

value={editedContent}

onChange={(e) => setEditedContent(e.target.value)}

/>

<div className="chapter-edit-actions">

<ActionButton onClick={() => handleSaveEdit(chapter.id)} className="small-btn">Salvar</ActionButton>

<ActionButton onClick={handleCancelEdit} className="secondary-btn small-btn">Cancelar</ActionButton>

</div>

</div>

) : (

<div className="chapter-item-content">

<div dangerouslySetInnerHTML={{ \_\_html: chapter.content.replace(/\n/g, '<br />') }} />

<ActionButton onClick={() => handleStartEdit(chapter)} className="secondary-btn small-btn" style={{marginTop: '1rem'}}>

Editar Capítulo

</ActionButton>

</div>

)}

</>

)}

</li>

);

})}

</ul>

) : (

<div className="empty-chapters">

<p>Seus capítulos escritos aparecerão aqui.</p>

</div>

)}

</div>

</div>

);

};

const ReaderView = ({ book, onBack }: { book: Book; onBack: () => void }) => {

type NarratingState = 'stopped' | 'playing' | 'loading';

const [narratingState, setNarratingState] = useState<NarratingState>('stopped');

const [currentChapterId, setCurrentChapterId] = useState<string | null>(null);

const audioContextRef = useRef<AudioContext | null>(null);

const audioSourceRef = useRef<AudioBufferSourceNode | null>(null);

const chapterQueueRef = useRef<Chapter[]>([]);

const isMountedRef = useRef(true);

const sortedChapters = book.chapters

.slice() // Create a shallow copy before sorting

.sort((a, b) => {

const outlineAIndex = book.outline.findIndex(o => o.id === a.id);

const outlineBIndex = book.outline.findIndex(o => o.id === b.id);

return outlineAIndex - outlineBIndex;

});

const getAudioContext = () => {

if (!audioContextRef.current) {

audioContextRef.current = new (window.AudioContext || (window as any).webkitAudioContext)({ sampleRate: 24000 });

}

return audioContextRef.current;

};

const stopNarration = useCallback(() => {

if (audioSourceRef.current) {

audioSourceRef.current.onended = null;

audioSourceRef.current.stop();

audioSourceRef.current = null;

}

chapterQueueRef.current = [];

if (isMountedRef.current) {

setNarratingState('stopped');

setCurrentChapterId(null);

}

}, []);

useEffect(() => {

isMountedRef.current = true;

return () => {

isMountedRef.current = false;

stopNarration();

};

}, [stopNarration]);

const generateAndPlay = async (chapter: Chapter) => {

if (!isMountedRef.current) return;

setCurrentChapterId(chapter.id);

setNarratingState('loading');

try {

const response = await fetch('/.netlify/functions/gemini', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ type: 'tts', payload: { prompt: chapter.content, characters: book.characters } }),

});

if (!response.ok) {

const errorData = await response.json().catch(() => ({ error: `HTTP error! status: ${response.status}` }));

throw new Error(errorData.error || `HTTP error! status: ${response.status}`);

}

const { base64Audio } = await response.json();

if (base64Audio && isMountedRef.current) {

const audioContext = getAudioContext();

const audioBuffer = await decodeAudioData(decode(base64Audio), audioContext, 24000, 1);

if (audioSourceRef.current) {

audioSourceRef.current.stop();

}

const source = audioContext.createBufferSource();

source.buffer = audioBuffer;

source.connect(audioContext.destination);

audioSourceRef.current = source;

setNarratingState('playing');

source.start();

source.onended = () => {

if (isMountedRef.current) {

playNextInQueue();

}

};

} else {

playNextInQueue();

}

} catch (error) {

console.error("Error generating or playing audio:", error);

if (isMountedRef.current) {

playNextInQueue();

}

}

};

const playNextInQueue = () => {

if (chapterQueueRef.current.length > 0) {

const nextChapter = chapterQueueRef.current.shift();

if (nextChapter) {

generateAndPlay(nextChapter);

}

} else {

stopNarration();

}

};

const handleToggleNarration = () => {

if (narratingState === 'playing' || narratingState === 'loading') {

stopNarration();

} else {

chapterQueueRef.current = [...sortedChapters];

playNextInQueue();

}

};

const handlePlaySingleChapter = (chapter: Chapter) => {

if (currentChapterId === chapter.id && (narratingState === 'playing' || narratingState === 'loading')) {

stopNarration();

} else {

chapterQueueRef.current = [chapter];

playNextInQueue();

}

};

return (

<div className="reader-container fade-in">

<ActionButton onClick={onBack} className="secondary-btn" style={{marginBottom: '2rem'}}>← Voltar ao Editor</ActionButton>

<div className="reader-header">

{book.coverImageUrl && <img src={book.coverImageUrl} alt={`Capa de ${book.title}`} className="reader-cover"/>}

<h1>{book.title}</h1>

<ActionButton onClick={handleToggleNarration} disabled={book.chapters.length === 0} className="narrate-btn">

{narratingState === 'loading' ? 'Carregando...' : narratingState === 'playing' ? '⏹️ Parar Narração' : '▶️ Ouvir a História'}

</ActionButton>

</div>

<div className="reader-content">

{sortedChapters.map(chapter => (

<div key={chapter.id} className={`reader-chapter ${currentChapterId === chapter.id ? 'current-chapter-highlight' : ''}`}>

<div className="chapter-header-with-controls">

<h3>{chapter.title}</h3>

<div className="audio-controls">

{narratingState === 'loading' && currentChapterId === chapter.id ? (

<div className="loader-small" role="status"><span className="sr-only">Carregando áudio</span></div>

) : (

<button

onClick={() => handlePlaySingleChapter(chapter)}

className="play-pause-btn"

aria-label={narratingState === 'playing' && currentChapterId === chapter.id ? `Pausar capítulo ${chapter.title}` : `Ouvir capítulo ${chapter.title}`}

>

{narratingState === 'playing' && currentChapterId === chapter.id ? '❚❚' : '▶'}

</button>

)}

</div>

</div>

<p dangerouslySetInnerHTML={{ \_\_html: chapter.content.replace(/\n/g, '<br />') }} />

</div>

))}

</div>

</div>

);

};

// --- MAIN APP COMPONENT ---

const App = () => {

const [books, setBooks] = useState<Book[]>(() => {

try {

const savedBooks = localStorage.getItem('ia-story-books');

const parsedBooks = savedBooks ? JSON.parse(savedBooks) : [];

// Data migration for the new character format

return parsedBooks.map((book: any, bookIndex: number) => {

if (!book.characters || typeof book.characters === 'string') {

return {

...book,

characters: book.characters ? [{

id: `char-${Date.now()}-${bookIndex}-0`,

name: 'Personagem Principal',

description: book.characters,

voice: 'Kore'

}] : [],

};

}

if (Array.isArray(book.characters)) {

return {

...book,

characters: book.characters.map((c: any, i: number) => ({

id: c.id || `char-${Date.now()}-${bookIndex}-${i}`,

name: c.name || '',

description: c.description || '',

voice: c.voice || 'Kore'

}))

};

}

return book;

});

} catch (error) {

console.error("Failed to load or migrate books from localStorage", error);

return [];

}

});

const [currentView, setCurrentView] = useState<'library' | 'editor' | 'reader'>('library');

const [selectedBookId, setSelectedBookId] = useState<string | null>(null);

const [apiKeyOk, setApiKeyOk] = useState<boolean | null>(null);

const [apiKeyError, setApiKeyError] = useState<string | null>(null);

useEffect(() => {

try {

localStorage.setItem('ia-story-books', JSON.stringify(books));

} catch (error) {

console.error("Failed to save books to localStorage", error);

}

}, [books]);

useEffect(() => {

const checkApiKey = async () => {

try {

const response = await fetch('/.netlify/functions/gemini', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ type: 'health' }),

});

if (response.ok) {

setApiKeyOk(true);

} else {

const errorData = await response.json().catch(() => null);

const errorMessage = errorData?.error || `Falha na verificação da API (HTTP ${response.status})`;

console.error("API Key check failed:", errorMessage);

setApiKeyOk(false);

setApiKeyError(errorMessage);

}

} catch (error) {

const errorMessage = error instanceof Error ? error.message : "Erro de conexão desconhecido.";

console.error("API key health check failed:", errorMessage);

setApiKeyOk(false);

setApiKeyError(errorMessage);

}

};

checkApiKey();

}, []);

const handleSelectBook = (bookId: string) => {

setSelectedBookId(bookId);

setCurrentView('editor');

};

const handleBackToLibrary = () => {

setSelectedBookId(null);

setCurrentView('library');

};

const handleReadBook = () => {

setCurrentView('reader');

}

const selectedBook = books.find(b => b.id === selectedBookId);

return (

<div className="app-container">

<header>

<h1>Estúdio de Histórias IA</h1>

<p>Sua assistente criativa para dar vida a mundos, personagens e narrativas épicas.</p>

</header>

<main>

{apiKeyOk === null && <LoadingSpinner message="Verificando configuração do servidor..." />}

{apiKeyOk === false && <ApiKeyErrorView serverError={apiKeyError} />}

{apiKeyOk === true && (

<>

{currentView === 'library' && <LibraryView books={books} setBooks={setBooks} onSelectBook={handleSelectBook} />}

{currentView === 'editor' && selectedBook && <BookEditorView book={selectedBook} setBooks={setBooks} onBack={handleBackToLibrary} onRead={handleReadBook}/>}

{currentView === 'reader' && selectedBook && <ReaderView book={selectedBook} onBack={() => setCurrentView('editor')} />}

</>

)}

</main>

</div>

);

};

const container = document.getElementById('root');

const root = createRoot(container!);

root.render(<App />);{

"name": "escritor-de-historias-ia",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "tsc && vite build",

"preview": "vite preview"

},

"dependencies": {

"@google/genai": "^1.24.0",

"react": "^18.3.1",

"react-dom": "^18.3.1"

},

"devDependencies": {

"@netlify/functions": "^2.8.1",

"@types/react": "^18.3.3",

"@types/react-dom": "^18.3.0",

"@vitejs/plugin-react": "^4.3.1",

"typescript": "^5.5.3",

"vite": "^5.3.3"

}

}# Estúdio de Histórias IA 📚

Bem-vindo ao Estúdio de Histórias IA, sua assistente criativa para dar vida a mundos, personagens e narrativas épicas. Este aplicativo web utiliza o poder da IA generativa da Google (Gemini) para ajudar escritores a criar livros completos, desde o conceito inicial até os capítulos detalhados e a narração em áudio.

## ✨ Funcionalidades

- \*\*Biblioteca Pessoal:\*\* Crie e gerencie múltiplos livros em sua própria biblioteca digital.

- \*\*Importação e Exportação:\*\* Salve e carregue toda a sua biblioteca com um único arquivo JSON.

- \*\*Conceituação Detalhada:\*\* Defina o título, o cenário e crie personagens únicos com descrições e vozes de narração distintas.

- \*\*Geração por IA:\*\*

- \*\*Capas de Livro:\*\* Gere capas de livro fotorrealistas e impressionantes com base no conceito da sua história.

- \*\*Sinopse e Tags:\*\* Crie uma sinopse envolvente e tags relevantes para o seu livro com um clique.

- \*\*Esboço da História:\*\* Desenvolva um esboço detalhado com dezenas de capítulos, cada um com um plano de eventos e desenvolvimento de personagens.

- \*\*Escrita de Capítulos:\*\* Expanda cada item do esboço em um capítulo completo e detalhado, com mais de 4.000 palavras.

- \*\*Editor Integrado:\*\* Edite e refine o conteúdo gerado pela IA diretamente no aplicativo.

- \*\*Leitor com Narração:\*\* Ouça sua história ganhar vida com uma narração em áudio multi-falante, onde cada personagem (e o narrador) tem sua própria voz.

## 🛠️ Tecnologias Utilizadas

- \*\*Frontend:\*\* React, TypeScript, Vite

- \*\*API de IA:\*\* Google Gemini API (modelos Flash, Flash Image e TTS)

- \*\*Backend:\*\* Netlify Functions (Serverless)

- \*\*Estilização:\*\* CSS puro com variáveis

## 🚀 Configuração e Deploy

Para rodar este projeto, você precisará de uma chave de API do Google Gemini e uma conta na Netlify.

### 1. Obtenha sua Chave de API

1. Acesse o [Google AI Studio](https://makersuite.google.com/app/apikey).

2. Clique em "Create API key in new project" para gerar sua chave.

3. Copie a chave. Você precisará dela no próximo passo.

### 2. Configuração na Netlify

Este projeto é configurado para ser implantado facilmente na Netlify.

1. \*\*Faça o Fork ou Clone do Repositório:\*\* Coloque o código em sua conta do GitHub.

2. \*\*Crie um Novo Site na Netlify:\*\*

- Na Netlify, clique em "Add new site" -> "Import an existing project".

- Conecte ao seu provedor Git (GitHub) e selecione o repositório do projeto.

3. \*\*Configure a Variável de Ambiente:\*\*

- Nas configurações do seu site na Netlify, vá para \*\*Site configuration\*\* → \*\*Build & deploy\*\* → \*\*Environment\*\*.

- Clique em \*\*New variable\*\*.

- Insira \*\*exatamente\*\* o seguinte:

- \*\*Key:\*\* `API\_KEY`

- \*\*Value:\*\* Cole a chave de API que você copiou do Google AI Studio.

4. \*\*Acione o Deploy:\*\*

- Vá para a aba \*\*Deploys\*\* e clique em \*\*"Trigger deploy" → "Deploy site"\*\*.

- \*\*Este passo é crucial!\*\* A Netlify só aplicará a nova variável de ambiente após um novo deploy.

O arquivo `netlify.toml` na raiz do projeto já contém todas as configurações de build necessárias, então a Netlify deve construir e implantar o site e a função serverless automaticamente.

## ✍️ Como Usar

1. \*\*Crie um Novo Livro:\*\* Na tela da biblioteca, clique em "+ Criar Novo Livro".

2. \*\*Defina o Conceito:\*\* Dê um título ao seu livro, adicione personagens (com nomes, descrições e vozes) e descreva o cenário.

3. \*\*Gere a Capa e Sinopse:\*\* Use os botões na seção 2 para que a IA crie uma capa e uma sinopse para sua história.

4. \*\*Gere o Esboço:\*\* Clique em "Gerar Esboço com IA". A IA criará uma lista de capítulos com um plano para cada um.

5. \*\*Escreva os Capítulos:\*\* Para cada item no esboço, clique em "Escrever" para que a IA gere um capítulo completo.

6. \*\*Edite e Refine:\*\* Após a geração, você pode expandir, editar e refinar cada capítulo conforme desejar.

7. \*\*Leia e Ouça:\*\* Quando tiver capítulos escritos, vá para a tela de leitura para ler sua história ou ouvi-la sendo narrada pela IA.

{

"compilerOptions": {

"target": "ES2020",

"useDefineForClassFields": true,

"lib": ["ES2020", "DOM", "DOM.Iterable"],

"module": "ESNext",

"skipLibCheck": true,

/\* Bundler mode \*/

"moduleResolution": "bundler",

"allowImportingTsExtensions": true,

"resolveJsonModule": true,

"isolatedModules": true,

"noEmit": true,

"jsx": "react-jsx",

/\* Linting \*/

"strict": true,

"noUnusedLocals": true,

"noUnusedParameters": true,

"noFallthroughCasesInSwitch": true

},

"include": ["index.tsx", "vite.config.ts"],

"references": [{ "path": "./tsconfig.node.json" }]

}{

"compilerOptions": {

"composite": true,

"skipLibCheck": true,

"module": "ESNext",

"moduleResolution": "bundler",

"allowSyntheticDefaultImports": true,

"strict": true

},

"include": ["vite.config.ts"]

}import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

// https://vitejs.dev/config/

export default defineConfig({

plugins: [react()],

});