UNINASSAU

Curso: Superior de Sistema da Informação

TRABALHO DE SISTEMAS DISTRIBUÍDOS

LibraryApi

SISTEMA DE GERENCIAMENTO DE BIBLIOTECA

EQUIPE:

Deangellis Berg Bezerra da Silva - 11024358

Cauã Siqueira Carneiro da Cunha - 01432916

Jonathas Xavier Freitas - 01403089

Pedro Emanuel Alencar de Melo - 01482712

Matheus Henrique Chagas - 01477597

Arnaldo Willian Duarte do Nascimento - 01401248

Ítalo Bessa e Souza - 01200094

Sumário

Introdução

- **Descrição do Sistema:**
  O Sistema de Gerenciamento de Biblioteca (SGB) é uma aplicação desenvolvida para facilitar o controle de empréstimos, devoluções, cadastro de livros, gerenciamento de usuários e outros processos associados à operação de uma biblioteca.

- **Objetivo:**
  Automatizar as operações da biblioteca, proporcionando eficiência, redução de erros e melhor experiência aos usuários e administradores.

.

Requisitos do Sistema

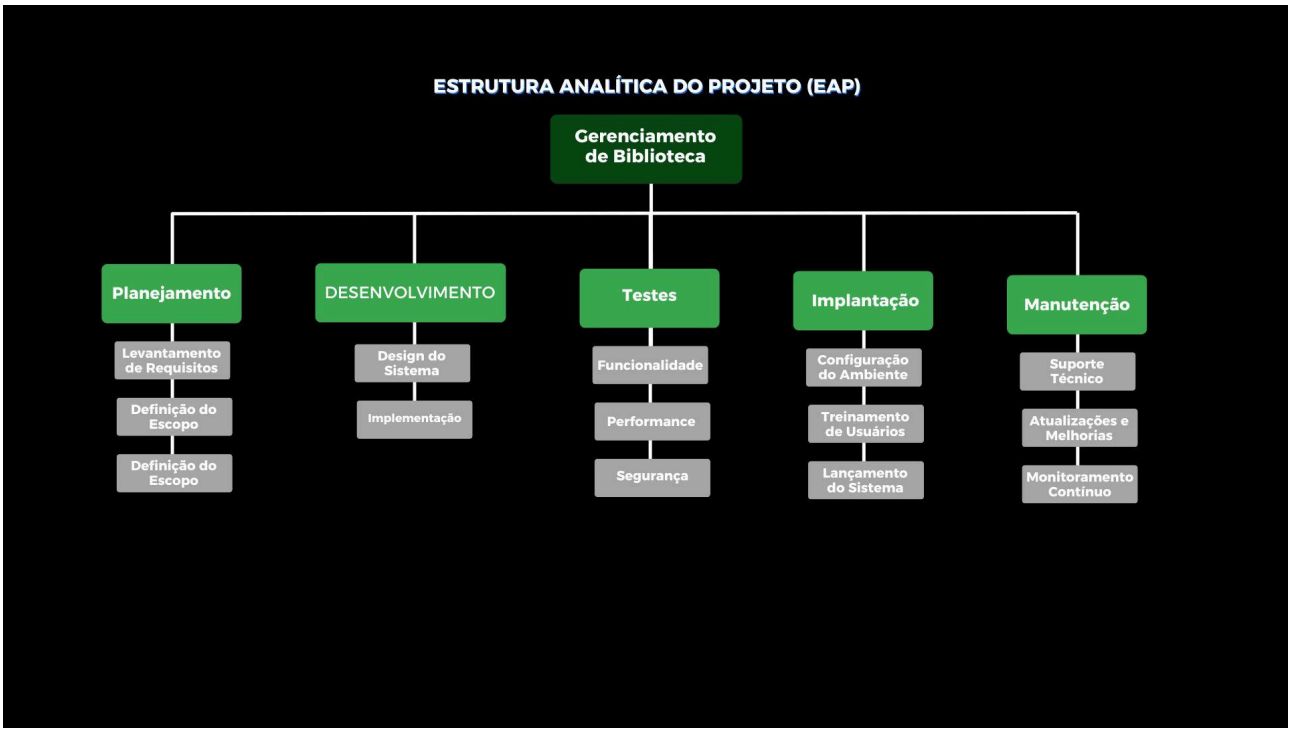
Requisitos Funcionais

- Cadastro, atualização, consulta e remoção de livros.
- Registro de usuários (estudantes, professores, visitantes).
- Sistema de empréstimos e devoluções com controle de prazos.
- Geração de relatórios (livros emprestados, atrasos, usuários frequentes).
- Sistema de reserva de livros.
- Login com perfis de administrador e usuário.

Requisitos Não Funcionais

- Interface amigável e responsiva.
- Tempo de resposta para operações inferior a 3 segundos.
- Backup automático dos dados.
- Suporte a múltiplos usuários simultaneamente.

Estrutura Analítica do Projeto (EAP)



Atribuição de Tarefas

Scrum Master/Líder da Equipe: Deangellis Berg Bezerra da Silva - 11024358

Desenvolvedor Backend: Cauã Siqueira Carneiro da Cunha - 01432916

Desenvolvedor Frontend: Jonathas Xavier Freitas - 01403089 / Pedro Emanuel Alencar de Melo - 01482712

Especialista UX/UI: Matheus Henrique Chagas 01477597 / Ítalo Bessa e Souza - 01200094

Analista de Dados: Deangellis Berg Bezerra 11024358 / Arnaldo Willian Duarte do Nascimento 01401248

Planilha de Custo

- **Funcionários:**

RESUMO - FOLHA DE PAGAMENTO		
CUSTO TOTAL MENSAL	**R$ 55.467,72**	*Custo total para a empresa*
SALÁRIOS BASE TOTAIS	R$ 37.800,00	*Salário base do funcionário*
IMPOSTOS TOTAIS A PAGAR	R$ 13.230,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 3.780,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 657,72	*Processos, seguros, despesas*

NOME	**Deangellis Berg Bezerra da Silva**	
CARGO	**Scrum Master**	
VALOR POR HORA	R$ 50,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 9.000,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 3.150,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 900,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 156,60	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 13.206,60**	*Custo total para a empresa*

NOME	**Ítalo Bessa e Souza**	
CARGO	**Gerente de configuração**	
VALOR POR HORA	R$ 50,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 9.000,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 3.150,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 900,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 156,60	*Processos, seguros, despesas*

CUSTO TOTAL MENSAL	**R$ 13.206,60**	*Custo total para a empresa*

NOME	**Pedro Emanuel Alencar de Melo**	
CARGO	**Documentador 1**	
VALOR POR HORA	R$ 35,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 6.300,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 2.205,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 630,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 109,62	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 9.244,62**	*Custo total para a empresa*

NOME	**Arnaldo Willian Duarte do Nascimento**	
CARGO	**Analista de Dados**	
VALOR POR HORA	R$ 25,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 4.500,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 1.575,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 450,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 78,30	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 6.603,30**	*Custo total para a empresa*

NOME	**Matheus Henrique Chagas**	
CARGO	**Especialista UX/U**	
VALOR POR HORA	R$ 25,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 4.500,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A	R$ 1.575,00	*Valor dos impostos sobre salário*

PAGAR		
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 450,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 78,30	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 6.603,30**	*Custo total para a empresa*
NOME	**Jonathas Xavier Freitas**	
CARGO	**Desenvolvedor 2**	
VALOR POR HORA	R$ 25,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 4.500,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 1.575,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 450,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 78,30	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 6.603,30**	*Custo total para a empresa*

NOME	**Cauã Siqueira Carneiro da Cunha**	
CARGO	**Desenvolvedor 3**	
VALOR POR HORA	R$ 45,00	*Inserir um valor médio*
TOTAL DE HORAS NO MÊS	180 Horas	*Horas mensais dedicadas*
SALÁRIO BASE	R$ 8.100,00	*Salário base do funcionário*
IMPOSTOS TOTAIS (% MÉDIO)	35%	*Média dos impostos sobre salário*
IMPOSTOS TOTAIS A PAGAR	R$ 2.835,00	*Valor dos impostos sobre salário*
BENEFÍCIOS (VR, VT, SAÚDE)	R$ 810,00	*Benefícios ao funcionário*
APROVISIONAMENTOS	R$ 140,94	*Processos, seguros, despesas*
CUSTO TOTAL MENSAL	**R$ 11.885,94**	*Custo total para a empresa*

- **Insumos:**

LICENÇAS DE SOFTWARE			
Microsoft Visual Studio	R$ 225,00	*Valor mensal*	
JetBrains IntelliJ IDE	R$ 245,00	*Valor mensal*	
MySQL Enterprise	R$ 25.000,00	*Valor anual*	
Adobe Creative Cloud	R$ 224,00	*Valor mensal*	

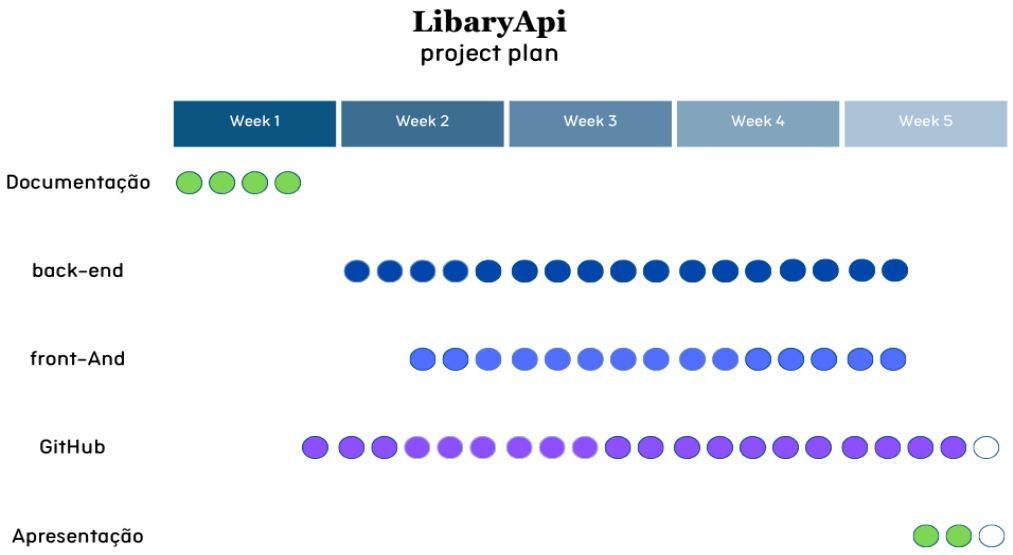
Postman	R$ 60,00	*Valor mensal*	
	R$ -		
	R$ -		
	R$ -		
CUSTO TOTAL MENSAL	**R$ 25.754,00**	*Custo total para a empresa*	

ASSINATURA DE SERVIÇOS			
Udemy Business	R$ 918,00	*Valor anual*	
GitHub Enterprise	R$ 107,00	*Valor mensal*	
Amazon Web Services	R$ 1.500,00	*Valor mensal*	
Trello	R$ 63,00	*Valor mensal*	
Hostinger	R$ 41,00	*Valor mensal*	
CUSTO TOTAL MENSAL	**R$ 2.629,00**	*Custo total para a empresa*	

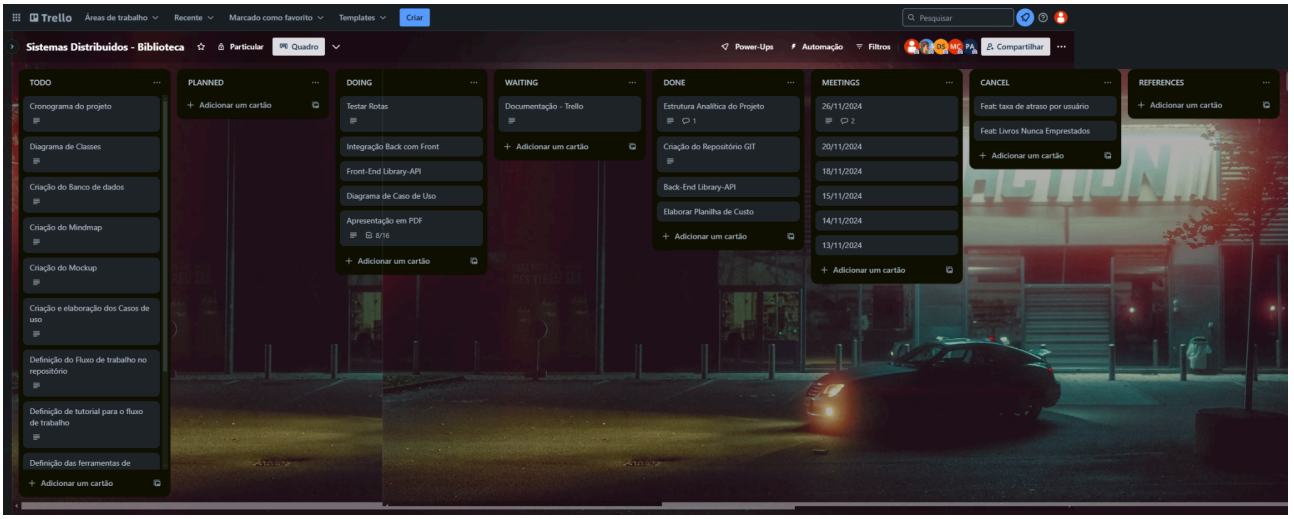
INFRAESTRUTURA E HARDWARE			
Link Internet 1Gbps	R$ 150,00	*Valor mensal*	
Roteador ASUS ROG WIFI6	R$ 1.600,00	*Valor único*	
Notebooks i7	R$ 5.500,00	*Valor único*	
Notebooks i3	R$ 2.500,00	*Valor único*	
Mesas de escritório 4 unidades	R$ 1.600,00	*Valor único*	
Cadeiras de escritório 4 unidades	R$ 1.200,00	*Valor único*	
Notebooks i7	R$ 5.500,00	*Valor único*	
Notebooks i7	R$ 5.500,00	*Valor único*	
CUSTO TOTAL ÚNICO	**R$ 23.400,00**	*Custo total para a empresa*	

CUSTO TOTAL MENSAL	**R$ 150,00**	*Custo total para a empresa*	
CUSTO TOTAL INFRAESTRUTURA	**R$ 23.550,00**	*Custo total para a empresa*	

Cronograma de Gantt



Resumo do Trello



O Sistema de Gerenciamento de Biblioteca está em progresso, com várias funcionalidades já em funcionamento. Desenvolvido para facilitar o controle de livros, empréstimos, devoluções, cadastro de usuários e reservas de livros.

As funções, como: o cadastro de livros e o controle de empréstimos, já estão funcionando. O time está trabalhando no sistema de reservas e no login de usuários, garantindo com que tudo funcione de forma segura e eficiente. A integração entre o que foi feito e o que falta fazer está sendo ajustada por nosso scrum master.

Ainda faltam algumas tarefas, como a criação de relatórios sobre livros emprestados e a realização de backups automáticos para garantir a segurança dos dados. Essas tarefas serão feitas nas próximas fases do projeto. O projeto está avançando bem com as principais funcionalidades já implementadas.

Casos de Uso

UC001 - Gerenciar Livros

Seção	Descrição
ID	UC001
Título	Gerenciar Livros
Ator Principal	Bibliotecário
Descrição	Permite ao bibliotecário realizar operações de CRUD (Criar, Ler, Atualizar, Deletar) para os livros.
Pré-condições	O bibliotecário deve estar autenticado no sistema.
Fluxo Principal	1. O bibliotecário acessa a funcionalidade de gerenciamento de livros. 2. Escolhe entre as opções de criar, editar, visualizar ou excluir um livro. 3. O sistema realiza a operação solicitada e confirma a execução.
Fluxo Alternativo	1. Caso as informações estejam incompletas ou inválidas, o sistema solicita correções.
Pós-condições	O banco de dados de livros é atualizado conforme a operação realizada.
Regras de Negócio	1. O título e o autor do livro são campos obrigatórios. 2. Não é permitido excluir um livro com empréstimos ativos.

UC002 - Gerenciar Usuários

Seção	Descrição
ID	UC002
Título	Gerenciar Usuários
Ator Principal	Bibliotecário
Descrição	Permite ao bibliotecário gerenciar os dados dos usuários registrados no sistema.
Pré-condições	O bibliotecário deve estar autenticado no sistema.
Fluxo Principal	1. O bibliotecário acessa a funcionalidade de gerenciamento de usuários. 2. Escolhe entre criar, editar, visualizar ou excluir um usuário. 3. O sistema realiza a operação e confirma a execução.
Fluxo Alternativo	1. Caso os dados do usuário sejam inválidos, o sistema solicita ajustes.
Pós-condições	Os dados do usuário são atualizados no sistema.
Regras de Negócio	1. O e-mail do usuário deve ser único. 2. Não é permitido excluir usuários com empréstimos ativos.

UC003 - Registrar Empréstimos

Seção	Descrição
ID	UC003
Título	Registrar Empréstimos
Ator Principal	Bibliotecário
Descrição	Permite ao bibliotecário registrar um empréstimo de livro para um usuário.
Pré-condições	O bibliotecário e o usuário devem estar cadastrados no sistema.
Fluxo Principal	1. O bibliotecário seleciona o usuário e o livro a ser emprestado. 2. Define a data de devolução. 3. O sistema registra o empréstimo e atualiza a disponibilidade do livro.
Fluxo Alternativo	1. Caso o usuário tenha atingido o limite de empréstimos, o sistema impede a operação.
Pós-condições	O empréstimo é registrado, e o livro fica indisponível para outros usuários.
Regras de Negócio	1. Cada usuário pode pegar emprestado até 5 livros ao mesmo tempo. 2. O prazo máximo de empréstimo é de 30 dias.

UC004 - Registrar Devoluções

Seção	Descrição
ID	UC004
Título	Registrar Devoluções
Ator Principal	Bibliotecário
Descrição	Permite ao bibliotecário registrar a devolução de um livro emprestado.
Pré-condições	O empréstimo do livro deve estar registrado no sistema.
Fluxo Principal	1. O bibliotecário seleciona o usuário e o livro devolvido. 2. O sistema atualiza o status do empréstimo e torna o livro disponível novamente.
Fluxo Alternativo	1. Caso o livro não seja encontrado no sistema, o bibliotecário é notificado do erro.
Pós-condições	O livro está disponível para novos empréstimos.
Regras de Negócio	1. Caso a devolução ultrapasse o prazo, o sistema deve registrar multa para o usuário.

UC005 - Gerar Relatórios

Seção	Descrição
ID	UC005
Título	Gerar Relatórios
Ator Principal	Bibliotecário
Descrição	Permite ao bibliotecário gerar relatórios sobre o sistema, como livros mais emprestados e usuários com pendências.
Pré-condições	O bibliotecário deve estar autenticado no sistema.
Fluxo Principal	1. O bibliotecário acessa a funcionalidade de relatórios. 2. Escolhe o tipo de relatório desejado. 3. O sistema gera o relatório e apresenta os dados.
Fluxo Alternativo	1. Caso não haja dados suficientes, o sistema exibe uma mensagem de aviso.
Pós-condições	O relatório é gerado e pode ser exportado pelo bibliotecário.
Regras de Negócio	1. Relatórios devem incluir informações atualizadas e precisas.

UC006 - Visualizar Livros

Seção	Descrição
ID	UC006
Título	Visualizar Livros
Ator Principal	Usuário
Descrição	Permite ao usuário consultar informações sobre os livros disponíveis no sistema.
Pré-condições	O usuário deve estar autenticado.
Fluxo Principal	1. O usuário acessa a funcionalidade de consulta. 2. Filtra ou pesquisa por livros de interesse. 3. O sistema exibe as informações dos livros.
Pós-condições	O usuário tem acesso às informações dos livros disponíveis.

UC007 - Realizar Empréstimos

Seção	Descrição
ID	UC007
Título	Realizar Empréstimos
Ator Principal	Usuário
Descrição	Permite ao usuário solicitar um empréstimo de livro.
Pré-condições	O usuário deve estar autenticado e não ter atingido o limite de empréstimos.
Fluxo Principal	1. O usuário seleciona o livro desejado. 2. Confirma o pedido de empréstimo. 3. O sistema registra o empréstimo.
Pós-condições	O livro é marcado como emprestado ao usuário.

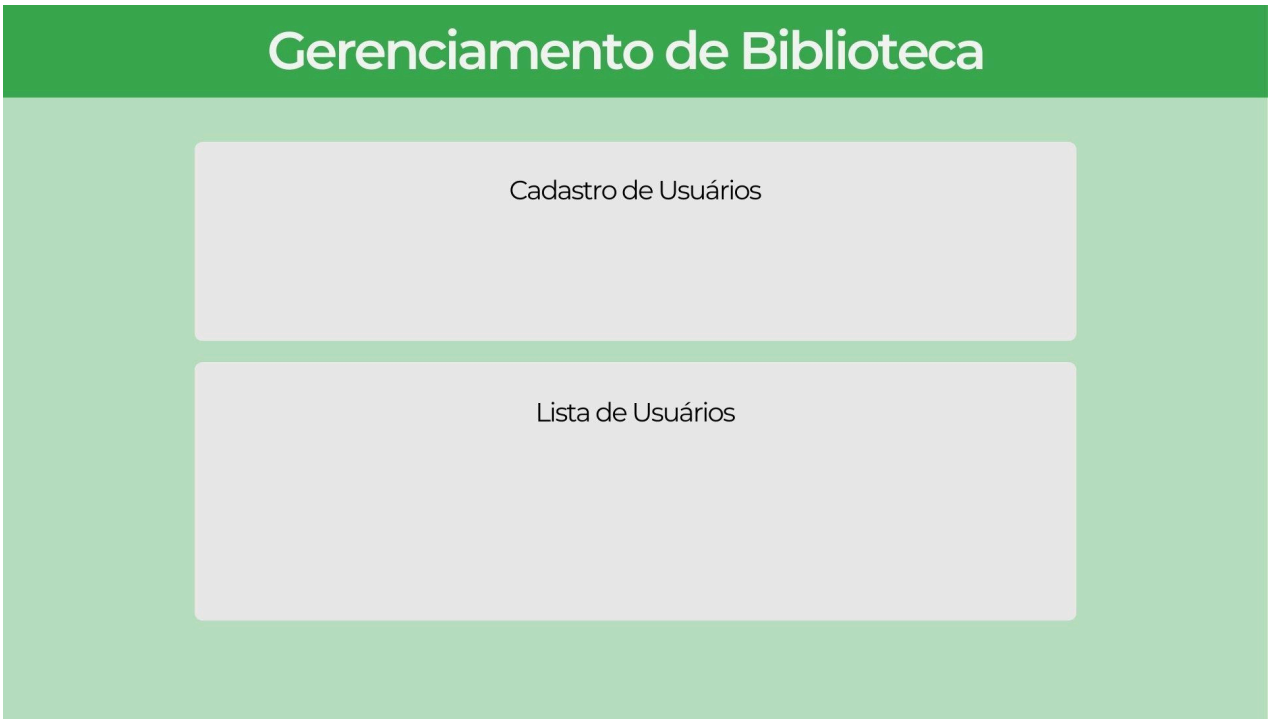
UC008 - Devolver Livros

Seção	Descrição
ID	UC008
Título	Devolver Livros
Ator Principal	Usuário
Descrição	Permite ao usuário registrar a devolução de livros emprestados.
Pré-condições	O usuário deve estar autenticado e o livro deve estar em posse do mesmo.
Fluxo Principal	1. O usuário seleciona o livro a ser devolvido. 2. Confirma a devolução. 3. O sistema registra a devolução.
Pós-condições	O livro está disponível para outros usuários.

Protótipos

1. Tela Inicial - Gerenciamento de Biblioteca

2. Cadastro/Lista de Usuário(s)

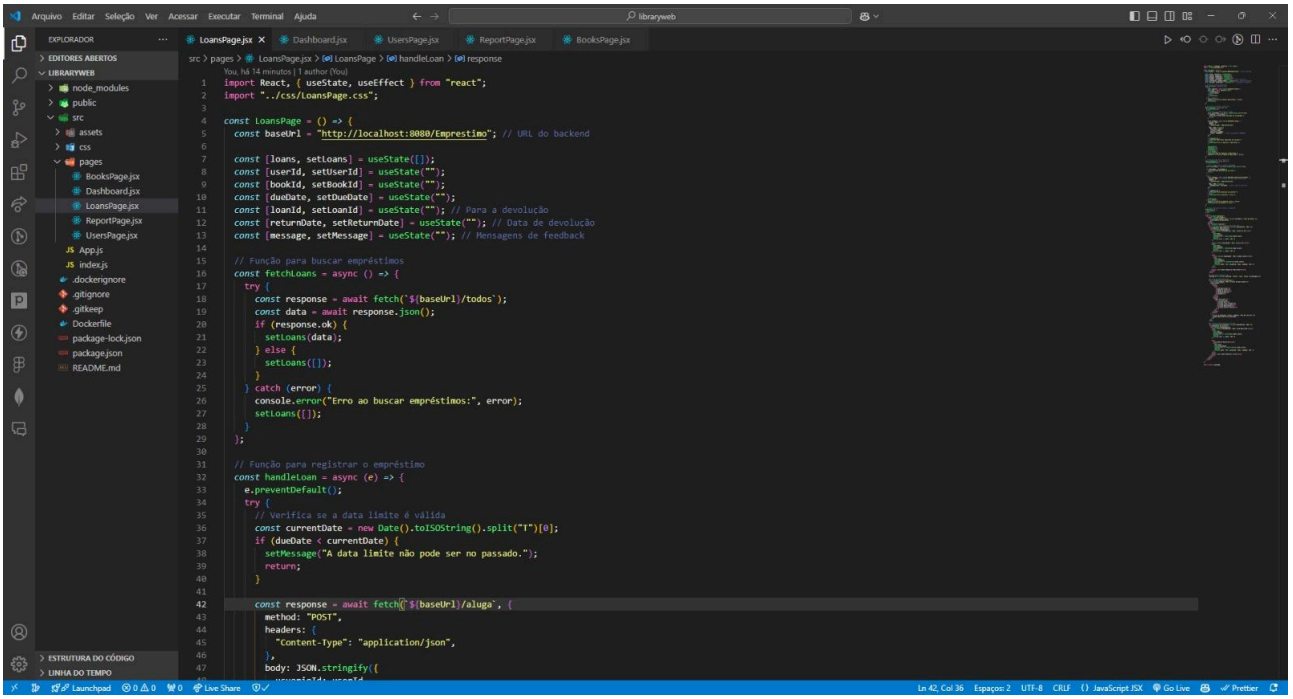


3. Registro/Lista/Atualização/Deletar de Livro

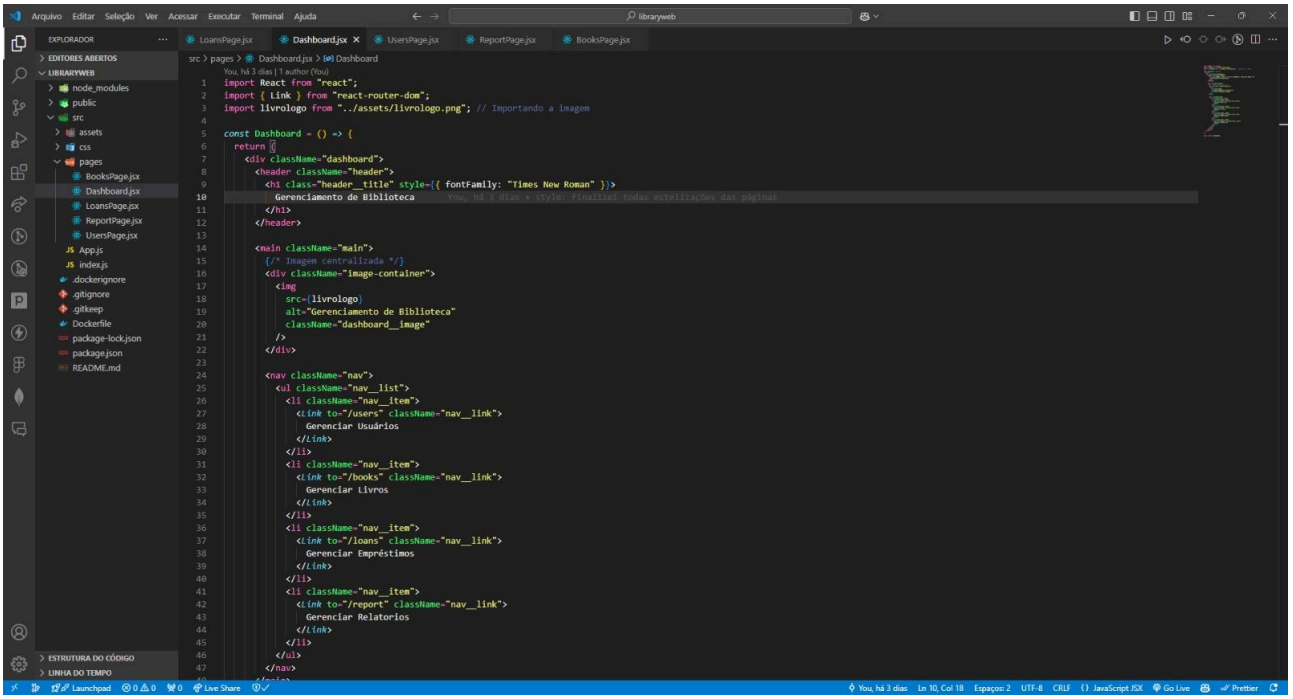
4. Registro/Lista de Empréstimo e Devolução de Livro

5. Relatórios de Empréstimo e Usuários com Empréstimo Pendentes

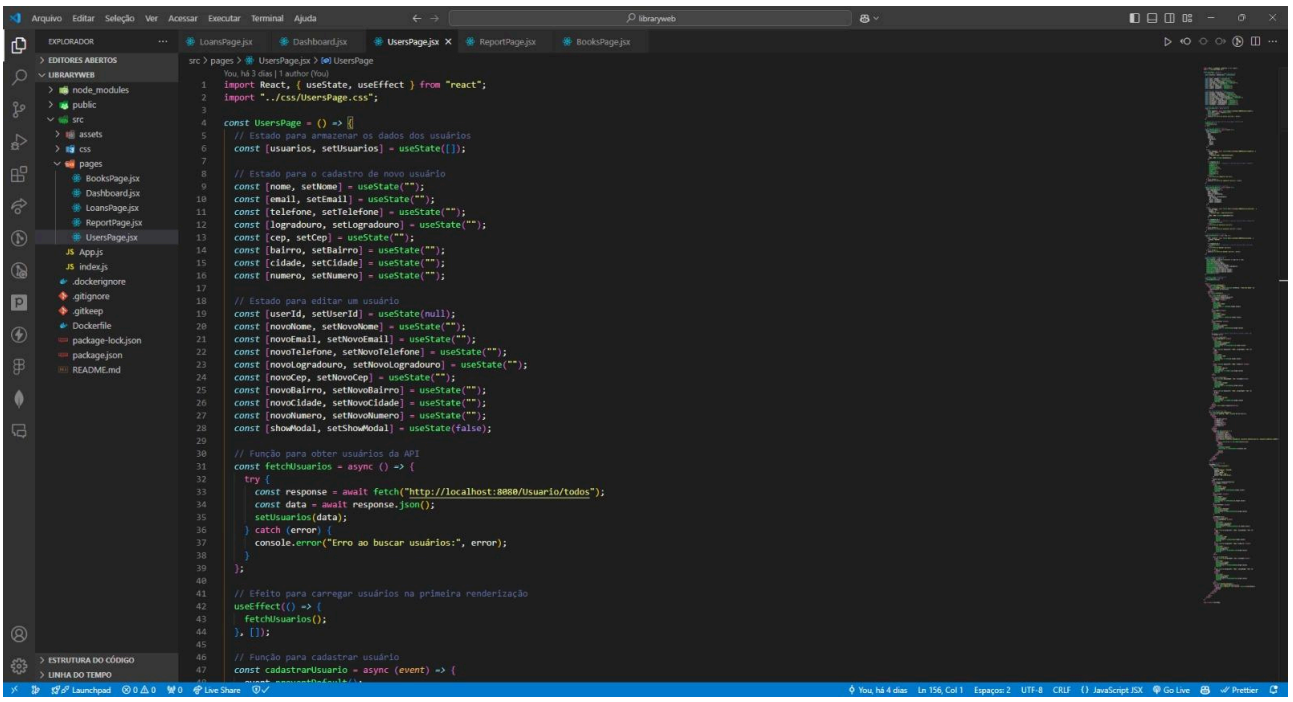
Extratos do Código



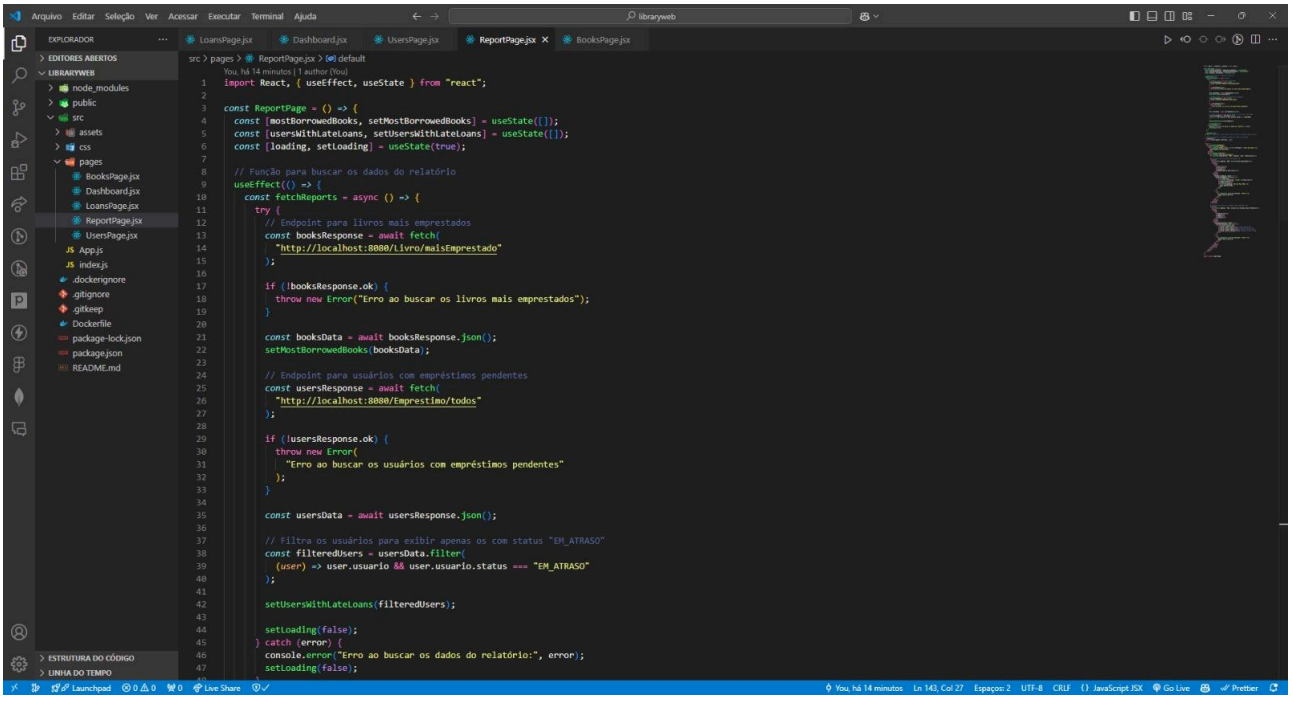
```jsx
import React, { useState, useEffect } from "react";
import "../css/LoansPage.css";

const LoansPage = () => {
  const baseUrl = "http://localhost:8080/Emprestimo"; // URL do backend

  const [loans, setLoans] = useState([]);
  const [userId, setUserId] = useState("");
  const [bookId, setBookId] = useState("");
  const [dueDate, setDueDate] = useState("");
  const [loanId, setLoanId] = useState(""); // Para a devolução
  const [returnDate, setReturnDate] = useState(""); // Data de devolução
  const [message, setMessage] = useState(""); // Mensagens de feedback

  // Função para buscar empréstimos
  const fetchLoans = async () => {
    try {
      const response = await fetch(`${baseUrl}/todos`);
      const data = await response.json();
      if (response.ok) {
        setLoans(data);
      } else {
        setLoans([]);
      }
    } catch (error) {
      console.error("Erro ao buscar empréstimos:", error);
      setLoans([]);
    }
  };

  // Função para registrar o empréstimo
  const handleLoan = async (e) => {
    e.preventDefault();
    try {
      // Verifica se a data limite é válida
      const currentDate = new Date().toISOString().split("T")[0];
      if (dueDate < currentDate) {
        setMessage("A data limite não pode ser no passado.");
        return;
      }

      const response = await fetch(`${baseUrl}/aluga`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
```



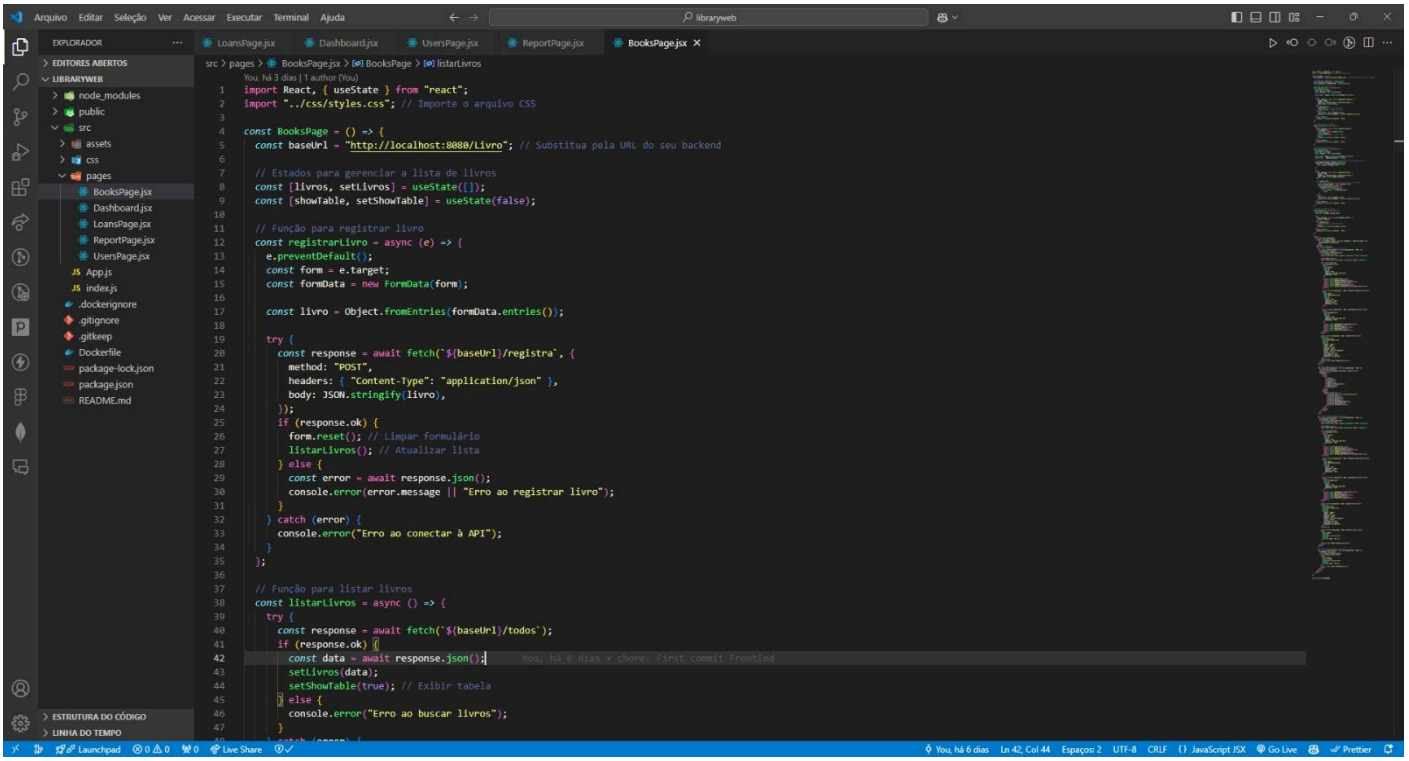
```jsx
import React from "react";
import { Link } from "react-router-dom";
import livrologo from "../assets/livrologo.png"; // Importando a imagem

const Dashboard = () => {
  return (
    <div className="dashboard">
      <header className="header">
        <h1 class="header__title" style={{ fontFamily: "Times New Roman" }}>
          Gerenciamento de Biblioteca
        </h1>
      </header>

      <main className="main">
        {/* Imagem centralizada */}
        <div className="image-container">
          <img
            src={livrologo}
            alt="Gerenciamento de Biblioteca"
            className="dashboard__image"
          />
        </div>

        <nav className="nav">
          <ul className="nav__list">
            <li className="nav__item">
              <Link to="/users" className="nav__link">
                Gerenciar Usuários
              </Link>
            </li>
            <li className="nav__item">
              <Link to="/books" className="nav__link">
                Gerenciar Livros
              </Link>
            </li>
            <li className="nav__item">
              <Link to="/loans" className="nav__link">
                Gerenciar Empréstimos
              </Link>
            </li>
            <li className="nav__item">
              <Link to="/report" className="nav__link">
                Gerenciar Relatorios
              </Link>
            </li>
          </ul>
        </nav>
```



```jsx
import React, { useState, useEffect } from "react";
import "../css/UsersPage.css";

const UsersPage = () => {
  // Estado para armazenar os dados dos usuários
  const [usuarios, setUsuarios] = useState([]);

  // Estado para o cadastro de novo usuário
  const [nome, setNome] = useState("");
  const [email, setEmail] = useState("");
  const [telefone, setTelefone] = useState("");
  const [logradouro, setLogradouro] = useState("");
  const [cep, setCep] = useState("");
  const [bairro, setBairro] = useState("");
  const [cidade, setCidade] = useState("");
  const [numero, setNumero] = useState("");

  // Estado para editar um usuário
  const [userId, setUserId] = useState(null);
  const [novoNome, setNovoNome] = useState("");
  const [novoEmail, setNovoEmail] = useState("");
  const [novoTelefone, setNovoTelefone] = useState("");
  const [novoLogradouro, setNovoLogradouro] = useState("");
  const [novoCep, setNovoCep] = useState("");
  const [novoBairro, setNovoBairro] = useState("");
  const [novoCidade, setNovoCidade] = useState("");
  const [novoNumero, setNovoNumero] = useState("");
  const [showModal, setShowModal] = useState(false);

  // Função para obter usuários da API
  const fetchUsuarios = async () => {
    try {
      const response = await fetch("http://localhost:8080/Usuario/todos");
      const data = await response.json();
      setUsuarios(data);
    } catch (error) {
      console.error("Erro ao buscar usuários:", error);
    }
  };

  // Efeito para carregar usuários na primeira renderização
  useEffect(() => {
    fetchUsuarios();
  }, []);

  // Função para cadastrar usuário
  const cadastrarUsuario = async (event) => {
```



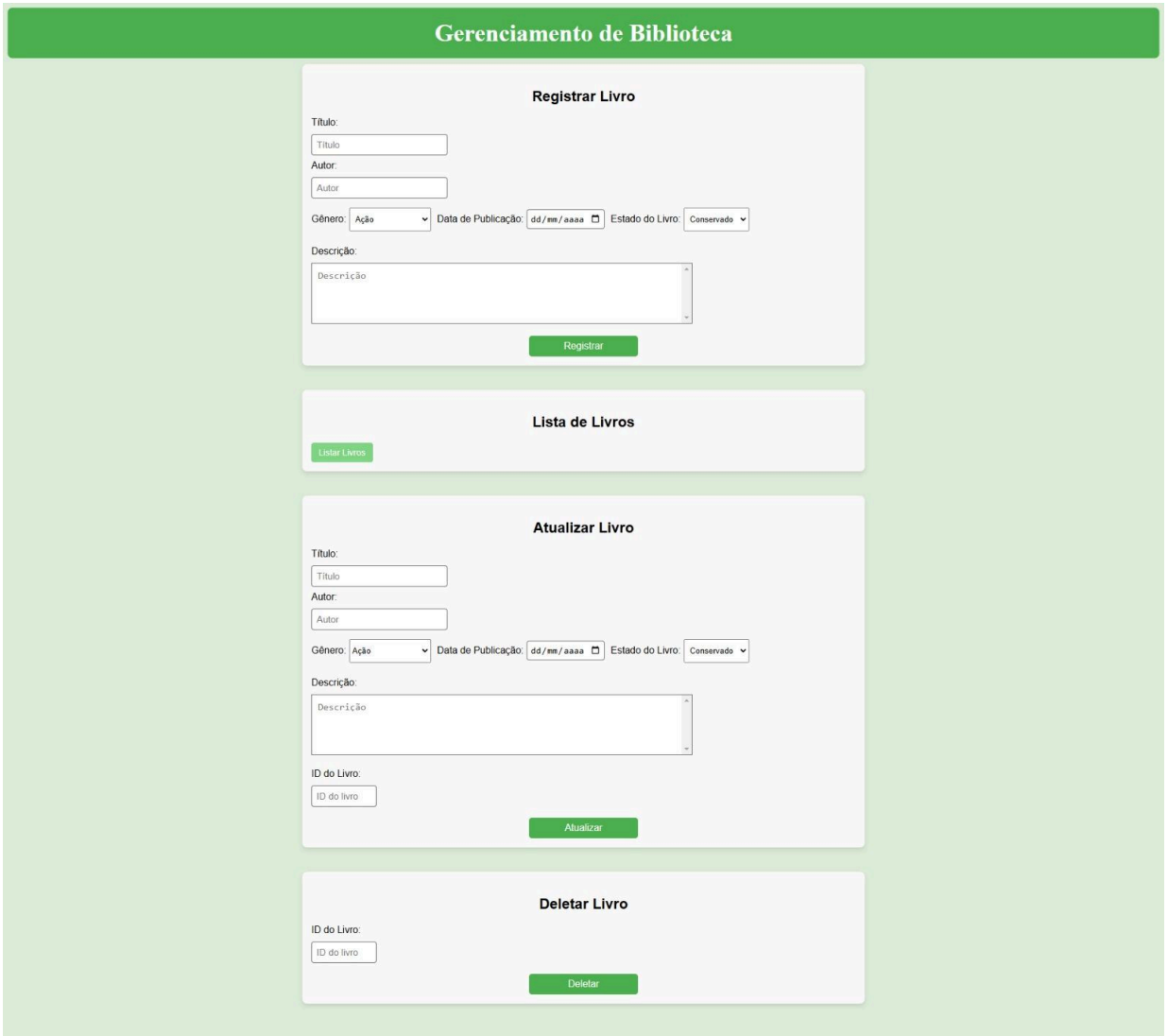
```jsx
import React, { useEffect, useState } from "react";

const ReportPage = () => {
  const [mostBorrowedBooks, setMostBorrowedBooks] = useState([]);
  const [usersWithLateLoans, setUsersWithLateLoans] = useState([]);
  const [loading, setLoading] = useState(true);

  // Função para buscar os dados do relatório
  useEffect(() => {
    const fetchReports = async () => {
      try {
        // Endpoint para livros mais emprestados
        const booksResponse = await fetch(
          "http://localhost:8080/Livro/maisEmprestado"
        );

        if (!booksResponse.ok) {
          throw new Error("Erro ao buscar os livros mais emprestados");
        }

        const booksData = await booksResponse.json();
        setMostBorrowedBooks(booksData);

        // Endpoint para usuários com empréstimos pendentes
        const usersResponse = await fetch(
          "http://localhost:8080/Emprestimo/todos"
        );

        if (!usersResponse.ok) {
          throw new Error(
            "Erro ao buscar os usuários com empréstimos pendentes"
          );
        }

        const usersData = await usersResponse.json();

        // Filtra os usuários para exibir apenas os com status "EM_ATRASO"
        const filteredUsers = usersData.filter(
          (user) => user.usuario && user.usuario.status === "EM_ATRASO"
        );

        setUsersWithLateLoans(filteredUsers);

        setLoading(false);
      } catch (error) {
        console.error("Erro ao buscar os dados do relatório:", error);
        setLoading(false);
```


Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
libraryweb
EXPLORADOR
EDITORES ABERTOS
LIBRARYWEB
node_modules
public
src
assets
css
pages
BooksPage.jsx
Dashboard.jsx
LoansPage.jsx
ReportPage.jsx
UsersPage.jsx
App.js
index.js
.dockerignore
.gitignore
.gitkeep
Dockerfile
package-lock.json
package.json
README.md
ESTRUTURA DO CÓDIGO
LINHA DO TEMPO
LoansPage.jsx
Dashboard.jsx
UsersPage.jsx
ReportPage.jsx
BooksPage.jsx
src > pages > BooksPage.jsx > BooksPage > listarLivros
You, há 3 dias | 1 author (You)
import React, { useState } from "react";
import "../css/styles.css"; // Importe o arquivo CSS
const BooksPage = () => {
const baseUrl = "http://localhost:8080/Livro"; // Substitua pela URL do seu backend
// Estados para gerenciar a lista de livros
const [livros, setLivros] = useState([]);
const [showTable, setShowTable] = useState(false);
// Função para registrar livro
const registrarLivro = async (e) => {
e.preventDefault();
const form = e.target;
const formData = new FormData(form);
const livro = Object.fromEntries(formData.entries());
try {
const response = await fetch(`${baseUrl}/registra`, {
method: "POST",
headers: { "Content-Type": "application/json" },
body: JSON.stringify(livro),
});
if (response.ok) {
form.reset(); // Limpar formulário
listarLivros(); // Atualizar lista
} else {
const error = await response.json();
console.error(error.message || "Erro ao registrar livro");
}
} catch (error) {
console.error("Erro ao conectar à API");
}
};
// Função para listar livros
const listarLivros = async () => {
try {
const response = await fetch(`${baseUrl}/todos`);
if (response.ok) {
const data = await response.json();
You, há 6 dias • chore: First commit FrontEnd
setLivros(data);
setShowTable(true); // Exibir tabela
} else {
console.error("Erro ao buscar livros");
}
Launchpad
Live Share
You, há 6 dias
Ln 42, Col 44
Espaços: 2
UTF-8
CRLF
JavaScript JSX
Go Live
Prettier

Demonstração de Uso

1. Tela Inicial - Gerenciamento de Biblioteca

2. Registro/Lista/Atualização/Deletar de Livro



3. Registro/Lista de Empréstimo e Devolução de Livro

4. Relatórios de Empréstimo e Usuários com Empréstimo Pendentes

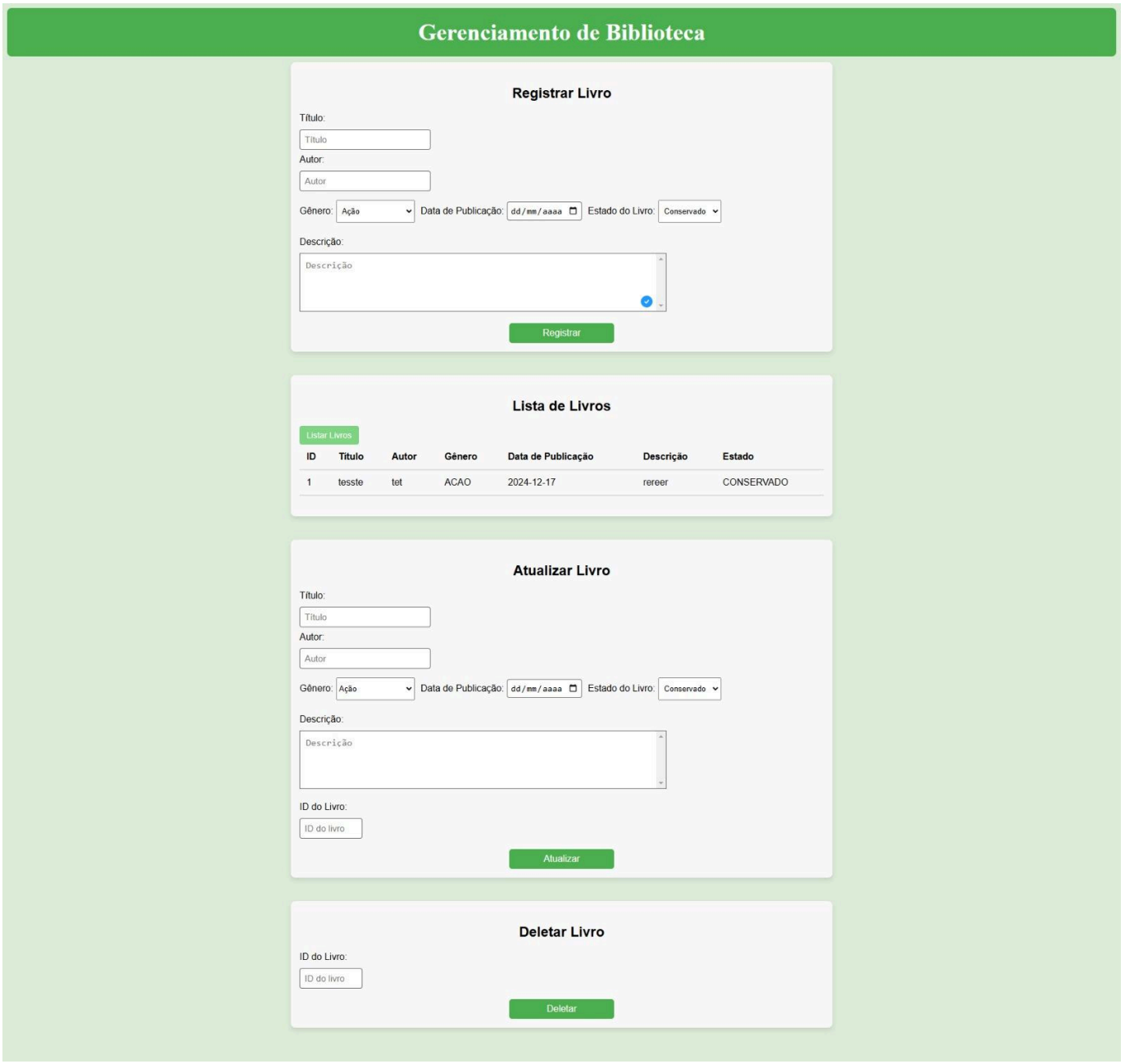
5. Cadastro/Lista de Usuário(s)

6. Cadastro/Lista de Usuário(s)

7. Registro/Lista de Empréstimo e Devolução de Livro

8. Registro/Lista/Atualização/Deletar de Livro

9. Registro/Lista/Atualização/Deletar de Livro



Lições Aprendidas

1. Planejamento Detalhado do Projeto

- Entender as operações da biblioteca e as necessidades de usuários de cada funcionário, usuário e administradores.
- Separar o que o sistema deve fazer e como o sistema deve se comportar.

2. Boas Práticas de Desenvolvimento

- Criar um design responsivo tanto para dispositivos móveis quanto desktop.
- Utilizar ferramentas como Git para rastrear alterações e facilitar colaboração.
- Segurança: Implementar autenticação e autorização (login com perfis).

3. Experiência em Colaboração

- A experiência prática em projetos colaborativos, incluindo a divisão de tarefas e integração de diferentes partes do sistema.
- Com o uso do Trello para organizar tarefas e acompanhar o progresso de cada integrante do projeto.
- Aprender com revisões de código, identificando melhorias em padrões.

4. Integração e Deploy

- APIs e Comunicação: Usar Fetch no frontend para consumir APIs criadas com Spring no backend.
- Configurar os ambientes de desenvolvimento e produção com o Docker.
- Testes: Realizar testes unitários e de integração para evitar falhas.

5. Gestão e Monitoramento

- Gerenciamento de Usuários: Diferenciar permissões de administradores e leitores.
- Suporte a Múltiplos Usuários: Garantir que o sistema funcione bem com vários acessos ao mesmo tempo.

6. Desafios e Soluções

- Manutenção do Código: Escrever documentação clara e comentar código.
- Desempenho: Monitorar tempo de resposta e otimizar consultas ao banco de dados.
- Adaptação às Necessidades: Projetar o sistema de forma escalável para futuras funcionalidades.

7. Aprendizados Adquiridos

- Integração de tecnologias (frontend e backend).
- Desenvolvimento orientado a usuários (UX/UI).
- Soluções para alta disponibilidade e confiabilidade.
- Uso de ferramentas modernas para gestão e deploy.

Agradecimentos

A conclusão do Sistema de Gerenciamento de Biblioteca (SGB) marca o resultado de uma jornada repleta de desafios e aprendizados. Este projeto reflete o esforço coletivo de uma equipe dedicada e talentosa, que trabalharam com comprometimento para transformar o projeto em uma solução prática.

Este projeto também reflete a sinergia e o aprendizado mútuo entre todos os membros. Cada ideia, cada linha de código e cada debate contribuíram para o desenvolvimento de um sistema que atenderá as necessidades das bibliotecas e de seus usuários.

Agradecemos a todos pelo empenho, criatividade e dedicação. Um reflexo do valor da colaboração e do aprendizado compartilhado, demonstrando que o comprometimento de todos podem gerar bons resultados.

Muito obrigado a todos!

Referências

1.Comunidades como Stack Overflow: Verificando erros durante o processo de outros usuários com erros parecidos, perguntas e respostas sobre problemas de programação.

2. Documentações Oficiais para aprender, relembrar e aprofundar seus conhecimentos sobre cada uma delas:

- React: Guia oficial para desenvolvimento de interfaces com React.
- Spring Framework: Tutorial para configurar e implementar APIs.
- MySQL: Referência para modelagem e manipulação de bancos de dados.
- Docker: Guia para virtualizar e gerenciar ambientes de desenvolvimento.
- Git: Para dúvidas de controle de versão e colaboração no código.

3. Cursos Online e Tutoriais, segue exemplos de referências:

- Codecademy - React: Curso prático para iniciantes em React.
- Udemy: Através de curso realizados por membros da equipe;
- Youtube: Aprender React, desde os fundamentos até tópicos avançados e realização de dúvidas pontuais sobre prototipagem até programação.
- Figma (Plataforma Oficial): Recursos gratuitos para aprender prototipagem e design responsivo.