



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	3
Requisitos do Sistema	3
Estrutura Analítica do Projeto (EAP)	4
Atribuição de Tarefas	4
Planilha de Custo	5
Cronograma de Gantt	9
Resumo do Trello	9
Casos de Uso	10
Protótipos	14
1. Tela Inicial - Gerenciamento de Biblioteca	14
2. Cadastro/Lista de Usuário(s)	14
3. Registro/Lista/Atualização/Deletar de Livro	15
4. Registro/Lista de Empréstimo e Devolução de Livro	15
5. Relatórios de Empréstimo e Usuários com Empréstimo Pendentes	16
Extratos do Código	17
Demonstração de Uso	20
1. Tela Inicial - Gerenciamento de Biblioteca	20
2. Registro/Lista/Atualização/Deletar de Livro	20
3. Registro/Lista de Empréstimo e Devolução de Livro	21
4. Relatórios de Empréstimo e Usuários com Empréstimo Pendentes	21
5. Cadastro/Lista de Usuário(s)	22
6. Cadastro/Lista de Usuário(s)	22
7. Registro/Lista de Empréstimo e Devolução de Livro	23
8. Registro/Lista/Atualização/Deletar de Livro	23
9. Registro/Lista/Atualização/Deletar de Livro	24
Lições Aprendidas	25
Agradecimentos	26
Referências	26

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

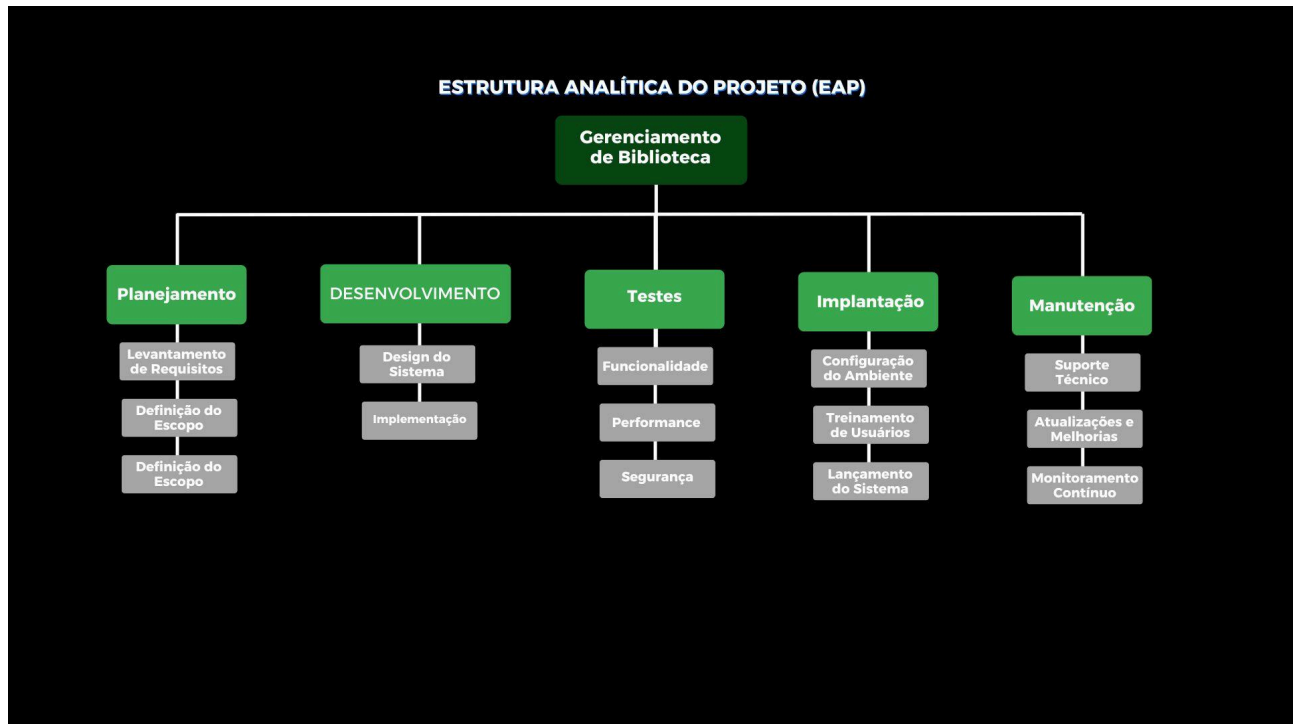
Requisitos Funcionais

- 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	<i>Custo total para a empresa</i>
SALÁRIOS BASE TOTAIS	R\$ 37.800,00	<i>Salário base do funcionário</i>
IMPOSTOS TOTAIS A PAGAR	R\$ 13.230,00	<i>Valor dos impostos sobre salário</i>
BENEFÍCIOS (VR, VT, SAÚDE)	R\$ 3.780,00	<i>Benefícios ao funcionário</i>
APROVISIONAMENTOS	R\$ 657,72	<i>Processos, seguros, despesas</i>

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

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

CUSTO TOTAL MENSAL	R\$ 13.206,60	<i>Custo total para a empresa</i>
---------------------------	----------------------	-----------------------------------

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

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

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

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:**

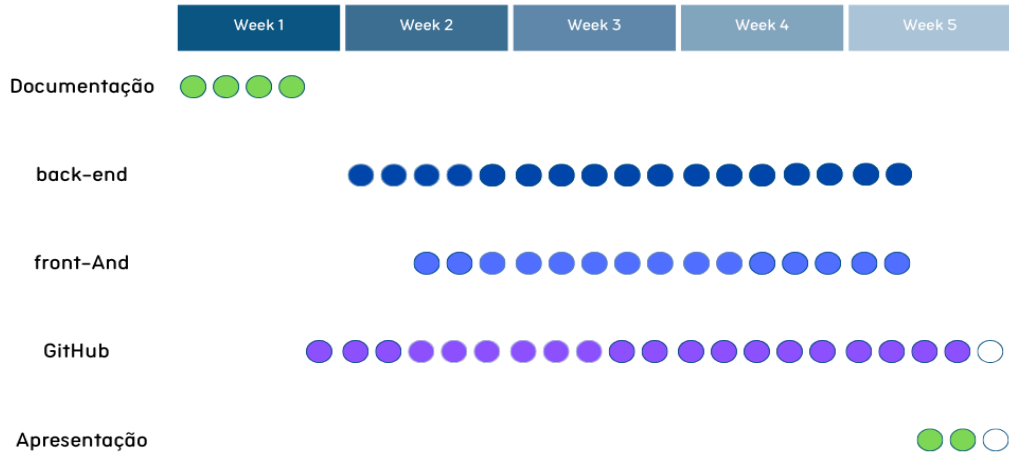
LICENÇAS DE SOFTWARE			
Microsoft Visual Studio	R\$ 225,00	Valor mensal	
JetBrains IntelliJ IDE	R\$ 245,00	Valor mensal	
MySQL Enterprise	R\$ 25.000,00	Valor anual	
Adobe Creative Cloud	R\$ 224,00	Valor mensal	

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	

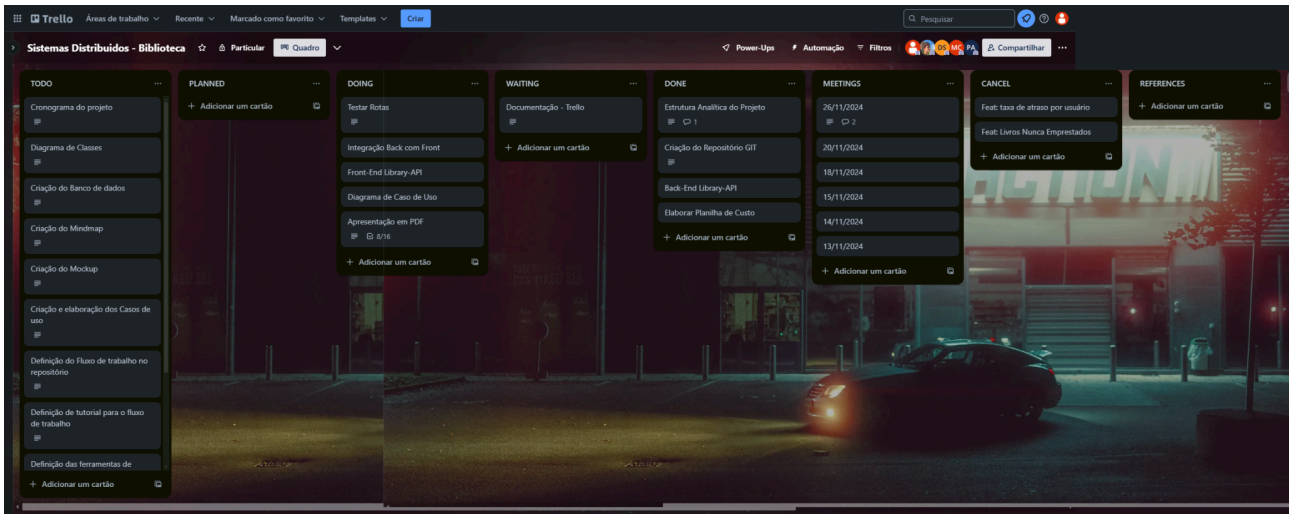
Cronograma de Gantt

LibaryApi

project plan



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

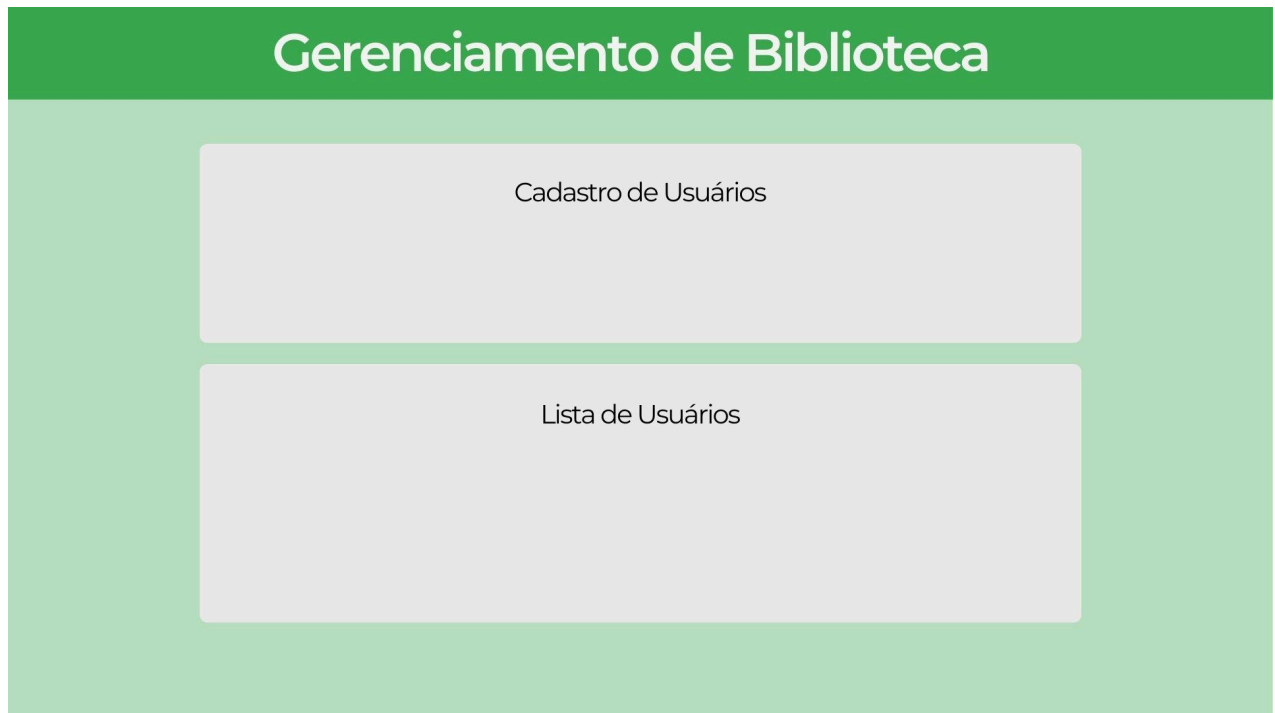
Seção	Descrição
ID	UC008
Título	Devolver Livros
Ator Principal	Usuário
Descrição	Permite ao usuário registrar a devolução de livros emprestados.
Pré-condições	O usuário deve estar autenticado e o livro deve estar em posse do mesmo.
Fluxo Principal	1. O usuário seleciona o livro a ser devolvido. 2. Confirma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

Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda

libraryweb

EXPLORADOR

EDITORIOS ABERTOS

LIBRARYWEB

- node_modules
- public
- src
 - assets
 - css
 - pages
 - BooksPage.jsx
 - Dashboard.jsx
 - LoansPage.jsx
 - ReportPage.jsx
 - UsersPage.jsx
 - App.jsx
 - index.js
 - dockerignore
 - gitignore
 - gitkeep
 - Dockerfile
 - package-lock.json
 - package.json
 - README.md

ESTRUTURA DO CÓDIGO

Linha do tempo

Launchpad

Live Share

Ln 42, Col 36 Espaço: 2 UTF-8 CRUF (1) JavaScript JSX Go Live Prettier

```
src > pages > LoansPage.jsx > [60] LoansPage > [60] handleLoan > [60] response
You, há 14 minutos | 1 autor (You)
1 import React, { useState, useEffect } from "react";
2 import "../css/LoansPage.css";
3
4 const LoansPage = () => {
5   const baseUrl = "http://localhost:8080/emprestimo"; // URL do backend
6
7   const [loans, setLoans] = useState([]);
8   const [userId, setUserId] = useState("");
9   const [bookId, setBookId] = useState("");
10  const [dueDate, setDueDate] = useState("");
11  const [loanId, setLoanId] = useState(""); // Para a devolução
12  const [returnDate, setReturnDate] = useState(""); // Data de devolução
13  const [message, setMessage] = useState(""); // Mensagens de feedback
14
15  // Função para buscar empréstimos
16  const fetchLoans = async () => {
17    try {
18      const response = await fetch(`${baseUrl}/todos`);
19      const data = await response.json();
20      if (response.ok) {
21        setLoans(data);
22      } else {
23        setLoans([]);
24      }
25    } catch (error) {
26      console.error("Erro ao buscar empréstimos:", error);
27      setLoans([]);
28    }
29  };
30
31  // Função para registrar o empréstimo
32  const handleLoan = async (e) => {
33    e.preventDefault();
34    try {
35      // Verifica se a data limite é válida
36      const currentDate = new Date().toISOString().split("T")[0];
37      if (dueDate < currentDate) {
38        setMessage("A data limite não pode ser no passado.");
39        return;
40      }
41
42      const response = await fetch(`${baseUrl}/aluga`, {
43        method: "POST",
44        headers: {
45          "Content-Type": "application/json",
46        },
47        body: JSON.stringify({
48          userId,
49          bookId,
50          dueDate,
51          loanId,
52        })
53      });
54      const data = await response.json();
55      if (response.ok) {
56        setLoans([...loans, data]);
57        setMessage("Empréstimo registrado com sucesso.");
58      } else {
59        setMessage("Erro ao registrar o empréstimo.");
60      }
61    } catch (error) {
62      console.error("Erro ao registrar o empréstimo:", error);
63      setMessage("Erro ao registrar o empréstimo.");
64    }
65  };
66
67  // Função para devolver o empréstimo
68  const handleReturn = async () => {
69    try {
70      const response = await fetch(`${baseUrl}/devolve/${loanId}`);
71      const data = await response.json();
72      if (response.ok) {
73        setLoans(loans.filter((loan) => loan.id !== loanId));
74        setMessage("Empréstimo devolvido com sucesso.");
75      } else {
76        setMessage("Erro ao devolver o empréstimo.");
77      }
78    } catch (error) {
79      console.error("Erro ao devolver o empréstimo:", error);
80      setMessage("Erro ao devolver o empréstimo.");
81    }
82  };
83
84  // Função para cancelar o empréstimo
85  const handleCancel = async () => {
86    try {
87      const response = await fetch(`${baseUrl}/cancela/${loanId}`);
88      const data = await response.json();
89      if (response.ok) {
90        setLoans(loans.filter((loan) => loan.id !== loanId));
91        setMessage("Empréstimo cancelado com sucesso.");
92      } else {
93        setMessage("Erro ao cancelar o empréstimo.");
94      }
95    } catch (error) {
96      console.error("Erro ao cancelar o empréstimo:", error);
97      setMessage("Erro ao cancelar o empréstimo.");
98    }
99  };
100
101  // Função para atualizar o status do empréstimo
102  const handleUpdateStatus = async (loanId, status) => {
103    try {
104      const response = await fetch(`${baseUrl}/atualiza/${loanId}/status`, {
105        method: "PUT",
106        headers: {
107          "Content-Type": "application/json",
108        },
109        body: JSON.stringify({ status })
110      });
111      const data = await response.json();
112      if (response.ok) {
113        setLoans(loans.map((loan) => (loan.id === loanId ? { ...loan, status }, {})));
114        setMessage("Status do empréstimo atualizado com sucesso.");
115      } else {
116        setMessage("Erro ao atualizar o status do empréstimo.");
117      }
118    } catch (error) {
119      console.error("Erro ao atualizar o status do empréstimo:", error);
120      setMessage("Erro ao atualizar o status do empréstimo.");
121    }
122  };
123
124  // Função para buscar o usuário logado
125  const fetchUser = async () => {
126    try {
127      const response = await fetch(`${baseUrl}/usuario`);
128      const data = await response.json();
129      if (response.ok) {
130        setUserId(data.userId);
131      } else {
132        setUserId("");
133      }
134    } catch (error) {
135      console.error("Erro ao buscar o usuário logado:", error);
136      setUserId("");
137    }
138  };
139
140  // Função para buscar o livro disponível
141  const fetchBooks = async () => {
142    try {
143      const response = await fetch(`${baseUrl}/livros`);
144      const data = await response.json();
145      if (response.ok) {
146        setBookId(data[0].id);
147      } else {
148        setBookId("");
149      }
150    } catch (error) {
151      console.error("Erro ao buscar o livro disponível:", error);
152      setBookId("");
153    }
154  };
155
156  // Função para buscar o calendário
157  const fetchCalendar = async () => {
158    try {
159      const response = await fetch(`${baseUrl}/calendario`);
160      const data = await response.json();
161      if (response.ok) {
162        setDueDate(data[0].dueDate);
163      } else {
164        setDueDate("");
165      }
166    } catch (error) {
167      console.error("Erro ao buscar o calendário:", error);
168      setDueDate("");
169    }
170  };
171
172  // Função para buscar o formulário de feedback
173  const fetchFeedbackForm = async () => {
174    try {
175      const response = await fetch(`${baseUrl}/feedback-form`);
176      const data = await response.json();
177      if (response.ok) {
178        setMessage(data.message);
179      } else {
180        setMessage("");
181      }
182    } catch (error) {
183      console.error("Erro ao buscar o formulário de feedback:", error);
184      setMessage("");
185    }
186  };
187
188  // Função para buscar o formulário de login
189  const fetchLoginForm = async () => {
190    try {
191      const response = await fetch(`${baseUrl}/login-form`);
192      const data = await response.json();
193      if (response.ok) {
194        setUserId(data.userId);
195      } else {
196        setUserId("");
197      }
198    } catch (error) {
199      console.error("Erro ao buscar o formulário de login:", error);
200      setUserId("");
201    }
202  };
203
204  // Função para buscar o formulário de cadastro
205  const fetchSignupForm = async () => {
206    try {
207      const response = await fetch(`${baseUrl}/signup-form`);
208      const data = await response.json();
209      if (response.ok) {
210        setUserId(data.userId);
211      } else {
212        setUserId("");
213      }
214    } catch (error) {
215      console.error("Erro ao buscar o formulário de cadastro:", error);
216      setUserId("");
217    }
218  };
219
220  // Função para buscar o formulário de recuperação de senha
221  const fetchResetPasswordForm = async () => {
222    try {
223      const response = await fetch(`${baseUrl}/reset-password-form`);
224      const data = await response.json();
225      if (response.ok) {
226        setUserId(data.userId);
227      } else {
228        setUserId("");
229      }
230    } catch (error) {
231      console.error("Erro ao buscar o formulário de recuperação de senha:", error);
232      setUserId("");
233    }
234  };
235
236  // Função para buscar o formulário de perfil de usuário
237  const fetchUserProfileForm = async () => {
238    try {
239      const response = await fetch(`${baseUrl}/user-profile-form`);
240      const data = await response.json();
241      if (response.ok) {
242        setUserId(data.userId);
243      } else {
244        setUserId("");
245      }
246    } catch (error) {
247      console.error("Erro ao buscar o formulário de perfil de usuário:", error);
248      setUserId("");
249    }
250  };
251
252  // Função para buscar o formulário de perfil de livro
253  const fetchBookProfileForm = async () => {
254    try {
255      const response = await fetch(`${baseUrl}/book-profile-form`);
256      const data = await response.json();
257      if (response.ok) {
258        setBookId(data.id);
259      } else {
260        setBookId("");
261      }
262    } catch (error) {
263      console.error("Erro ao buscar o formulário de perfil de livro:", error);
264      setBookId("");
265    }
266  };
267
268  // Função para buscar o formulário de perfil de calendário
269  const fetchCalendarProfileForm = async () => {
270    try {
271      const response = await fetch(`${baseUrl}/calendar-profile-form`);
272      const data = await response.json();
273      if (response.ok) {
274        setDueDate(data.dueDate);
275      } else {
276        setDueDate("");
277      }
278    } catch (error) {
279      console.error("Erro ao buscar o formulário de perfil de calendário:", error);
280        setDueDate("");
281    }
282  };
283
284  // Função para buscar o formulário de perfil de feedback
285  const fetchFeedbackProfileForm = async () => {
286    try {
287      const response = await fetch(`${baseUrl}/feedback-profile-form`);
288      const data = await response.json();
289      if (response.ok) {
290        setMessage(data.message);
291      } else {
292        setMessage("");
293      }
294    } catch (error) {
295      console.error("Erro ao buscar o formulário de perfil de feedback:", error);
296        setMessage("");
297    }
298  };
299
300  // Função para buscar o formulário de perfil de login
301  const fetchLoginProfileForm = async () => {
302    try {
303      const response = await fetch(`${baseUrl}/login-profile-form`);
304      const data = await response.json();
305      if (response.ok) {
306        setUserId(data.userId);
307      } else {
308        setUserId("");
309      }
310    } catch (error) {
311      console.error("Erro ao buscar o formulário de perfil de login:", error);
312        setUserId("");
313    }
314  };
315
316  // Função para buscar o formulário de perfil de cadastro
317  const fetchSignupProfileForm = async () => {
318    try {
319      const response = await fetch(`${baseUrl}/signup-profile-form`);
320      const data = await response.json();
321      if (response.ok) {
322        setUserId(data.userId);
323      } else {
324        setUserId("");
325      }
326    } catch (error) {
327      console.error("Erro ao buscar o formulário de perfil de cadastro:", error);
328        setUserId("");
329    }
330  };
331
332  // Função para buscar o formulário de perfil de recuperação de senha
333  const fetchResetPasswordProfileForm = async () => {
334    try {
335      const response = await fetch(`${baseUrl}/reset-password-profile-form`);
336      const data = await response.json();
337      if (response.ok) {
338        setUserId(data.userId);
339      } else {
340
```

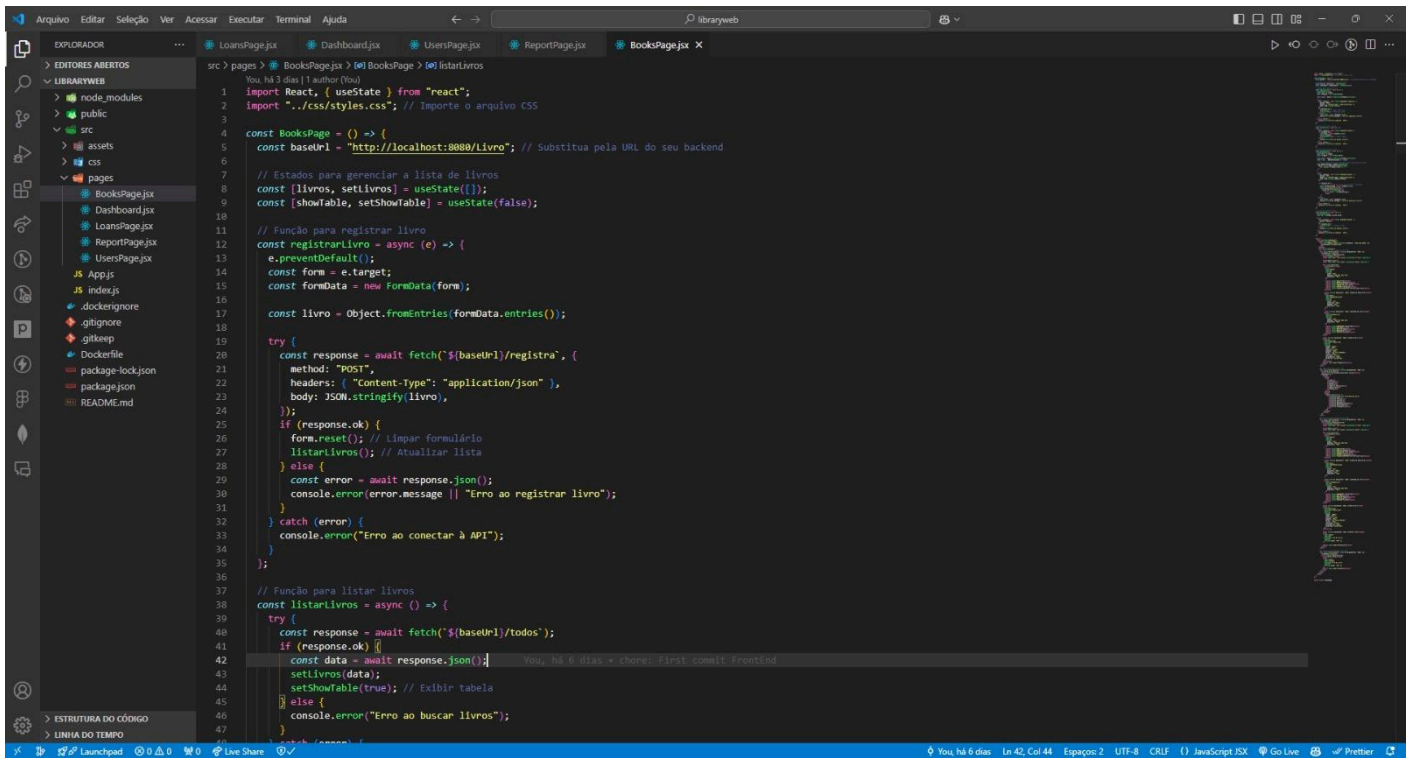
The screenshot shows a VS Code editor with a React project. The Explorer sidebar on the left displays the project structure, including folders like 'src' and 'pages'. The main editor area shows the code for 'Dashboard.jsx'. The code includes imports for React, Link, and Router, and a function 'Dashboard' that returns a JSX element. The code defines a header with a title 'Gerenciamento de Biblioteca' and a main content area with a navigation list and a book image placeholder.

```

src > pages > Dashboard.jsx | Dashboard
You have 1 class (1 author) (you)
1 import React from "react";
2 import { Link } from "react-router-dom";
3 import livrologo from "../assets/livrologo.png"; // Importando a imagem
4
5 const Dashboard = () => {
6   return (
7     <div className="dashboard">
8       <header className="header">
9         <div className="header_title" style={{ fontFamily: "Times New Roman" }}>
10           Gerenciamento de Biblioteca
11         </div>
12       </header>
13
14       <main className="main">
15         <div className="image-container">
16           <img
17             src={livrologo}
18             alt="Gerenciamento de Biblioteca"
19             className="dashboard_image"
20           />
21         </div>
22
23         <nav className="nav">
24           <ul className="nav_list">
25             <li className="nav_item">
26               <Link to="/users" className="nav_link">
27                 Gerenciar Usuários
28               </Link>
29             </li>
30             <li className="nav_item">
31               <Link to="/books" className="nav_link">
32                 Gerenciar Livros
33               </Link>
34             </li>
35             <li className="nav_item">
36               <Link to="/loans" className="nav_link">
37                 Gerenciar Empréstimos
38               </Link>
39             </li>
40             <li className="nav_item">
41               <Link to="/report" className="nav_link">
42                 Gerenciar Relatórios
43               </Link>
44             </li>
45           </ul>
46         </nav>
47       </main>
48     </div>
49   );
50 }
51
52 export default Dashboard;
  
```

```
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
Dockerfile
package-lock.json
package.json
README.md
src > pages > UsersPage.jsx > @ UsersPage
You há 3 dias 1 autor (You)
1 import React, { useState, useEffect } from "react";
2 import "../css/UsersPage.css";
3
4 const UsersPage = () => {
5   // Estado para armazenar os dados dos usuários
6   const [usuarios, setUsuarios] = useState([]);
7
8   // Estado para o cadastro de novo usuário
9   const [nome, setNome] = useState("");
10  const [email, setEmail] = useState("");
11  const [telefone, setTelefone] = useState("");
12  const [logradouro, setLogradouro] = useState("");
13  const [cep, setCep] = useState("");
14  const [bairro, setBairro] = useState("");
15  const [cidade, setCidade] = useState("");
16  const [numero, setNumero] = useState("");
17
18  // Estado para editar um usuário
19  const [userId, setUserId] = useState(null);
20  const [novonome, setNovonome] = useState("");
21  const [novonome, setNovonome] = useState("");
22  const [novotelefone, setNovotelefone] = useState("");
23  const [novologradouro, setNovologradouro] = useState("");
24  const [novocep, setNovocep] = useState("");
25  const [novobairro, setNovobairro] = useState("");
26  const [novoestado, setNovoestado] = useState("");
27  const [novonumero, setNovonumero] = useState("");
28  const [showModal, setShowModal] = useState(false);
29
30  // Função para obter usuários da API
31  const fetchUsuarios = async () => {
32    try {
33      const response = await fetch("http://localhost:8080/Usuario/todos");
34      const data = await response.json();
35      setUsuarios(data);
36    } catch (error) {
37      console.error("Erro ao buscar usuários:", error);
38    }
39  };
40
41  // Efeito para carregar usuários na primeira renderização
42  useEffect(() => {
43    fetchUsuarios();
44  }, []);
45
46  // Função para cadastrar usuário
47  const cadastrarUsuario = async (event) => {
48    event.preventDefault();
49    const novoUsuario = {
50      nome: nome,
51      email: email,
52      telefone: telefone,
53      logradouro: logradouro,
54      cep: cep,
55      bairro: bairro,
56      cidade: cidade,
57      numero: numero,
58    };
59    const response = await fetch("http://localhost:8080/Usuario", {
60      method: "POST",
61      headers: {
62        "Content-Type": "application/json",
63      },
64      body: JSON.stringify(novoUsuario),
65    });
66    if (response.ok) {
67      fetchUsuarios();
68    }
69  };
70
71  // Função para editar usuário
72  const editarUsuario = async (userId) => {
73    if (!userId) return;
74    const novoUsuario = {
75      nome: novonome,
76      email: novonome,
77      telefone: novotelefone,
78      logradouro: novologradouro,
79      cep: novocep,
80      bairro: novobairro,
81      cidade: novoestado,
82      numero: novonumero,
83    };
84    const response = await fetch(`http://localhost:8080/Usuario/${userId}`, {
85      method: "PUT",
86      headers: {
87        "Content-Type": "application/json",
88      },
89      body: JSON.stringify(novoUsuario),
90    });
91    if (response.ok) {
92      fetchUsuarios();
93    }
94  };
95
96  // Função para deletar usuário
97  const deletarUsuario = async (userId) => {
98    const response = await fetch(`http://localhost:8080/Usuario/${userId}`, {
99      method: "DELETE",
100    });
101    if (response.ok) {
102      fetchUsuarios();
103    }
104  };
105
106  return (
107    <div>
108      <h2>Usuários</h2>
109      <div>
110        <input type="text" value={nome} onChange={setNome} />
111        <input type="text" value={email} onChange={setEmail} />
112        <input type="text" value={telefone} onChange={setTelefone} />
113        <input type="text" value={logradouro} onChange={setLogradouro} />
114        <input type="text" value={cep} onChange={setCep} />
115        <input type="text" value={bairro} onChange={setBairro} />
116        <input type="text" value={cidade} onChange={setCidade} />
117        <input type="text" value={numero} onChange={setNumero} />
118        <button onClick={cadastrarUsuario}>Cadastrar</button>
119      </div>
120      <div>
121        <table>
122          <thead>
123            <tr>
124              <th>ID</th>
125              <th>Nome</th>
126              <th>Email</th>
127              <th>Telefone</th>
128              <th>Logradouro</th>
129              <th>CEP</th>
130              <th>Bairro</th>
131              <th>Cidade</th>
132              <th>Numero</th>
133            </tr>
134          </thead>
135          <tbody>
136            {usuarios.map((user) => (
137              <tr>
138                <td>{user.id}</td>
139                <td>{user.nome}</td>
140                <td>{user.email}</td>
141                <td>{user.telefone}</td>
142                <td>{user.logradouro}</td>
143                <td>{user.cep}</td>
144                <td>{user.bairro}</td>
145                <td>{user.cidade}</td>
146                <td>{user.numero}</td>
147              </tr>
148            ))}
149          </tbody>
150        </table>
151        <div>
152          <input type="text" value={novonome} onChange={setNovonome} />
153          <input type="text" value={novonome} onChange={setNovonome} />
154          <input type="text" value={novotelefone} onChange={setNovotelefone} />
155          <input type="text" value={novologradouro} onChange={setNovologradouro} />
156          <input type="text" value={novocep} onChange={setNovocep} />
157          <input type="text" value={novobairro} onChange={setNovobairro} />
158          <input type="text" value={novoestado} onChange={setNovoestado} />
159          <input type="text" value={novonumero} onChange={setNovonumero} />
160          <button onClick={() => editarUsuario(userId)}>Editar</button>
161          <button onClick={() => deletarUsuario(userId)}>Deletar</button>
162        </div>
163      </div>
164    </div>
165  );
166}
167
168export default UsersPage;
```

```
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
Dockerfile
package-lock.json
package.json
README.md
src > pages > ReportPage.jsx > @ default
You há 14 minutos 1 autor (You)
1 import React, { useState, useEffect } from "react";
2
3 const ReportPage = () => {
4   const [mostBorrowedBooks, setMostBorrowedBooks] = useState([]);
5   const [usersWithLateLoans, setUserWithLateLoans] = useState([]);
6   const [loading, setLoading] = useState(true);
7
8   // Função para buscar os dados do relatório
9   const fetchReports = async () => {
10     try {
11       // Endpoint para livros mais emprestados
12       const booksResponse = await fetch(
13         "http://localhost:8080/Livro/maisEmprestado"
14       );
15
16       if (!booksResponse.ok) {
17         throw new Error("Erro ao buscar os livros mais emprestados");
18       }
19
20       const booksData = await booksResponse.json();
21       setMostBorrowedBooks(booksData);
22
23       // Endpoint para usuários com empréstimos pendentes
24       const usersResponse = await fetch(
25         "http://localhost:8080/emprestimo/todos"
26       );
27
28       if (!usersResponse.ok) {
29         throw new Error("Erro ao buscar os usuários com empréstimos pendentes");
30       }
31
32       const usersData = await usersResponse.json();
33
34       // Filtra os usuários para exibir apenas os com status "EM_ATRASO"
35       const filteredUsers = usersData.filter(
36         (user) => user.usuario && user.usuario.status === "EM_ATRASO"
37       );
38
39       setUserWithLateLoans(filteredUsers);
40
41       setLoading(false);
42     } catch (error) {
43       console.error("Erro ao buscar os dados do relatório:", error);
44       setLoading(false);
45     }
46   };
47
48   return (
49     <div>
50       <h2>Relatório</h2>
51       <div>
52         <h3>Livros mais emprestados</h3>
53         <table>
54           <thead>
55             <tr>
56               <th>ID</th>
57               <th>Título</th>
58               <th>Quantidade</th>
59             </tr>
60           </thead>
61           <tbody>
62             {mostBorrowedBooks.map((book) => (
63               <tr>
64                 <td>{book.id}</td>
65                 <td>{book.titulo}</td>
66                 <td>{book.quantidade}</td>
67               </tr>
68             ))}
69           </tbody>
70         </table>
71       </div>
72       <div>
73         <h3>Usuários com empréstimos pendentes</h3>
74         <table>
75           <thead>
76             <tr>
77               <th>ID</th>
78               <th>Nome</th>
79               <th>Email</th>
80               <th>Telefone</th>
81               <th>Logradouro</th>
82               <th>CEP</th>
83               <th>Bairro</th>
84               <th>Cidade</th>
85               <th>Numero</th>
86             </tr>
87           </thead>
88           <tbody>
89             {usersWithLateLoans.map((user) => (
90               <tr>
91                 <td>{user.id}</td>
92                 <td>{user.nome}</td>
93                 <td>{user.email}</td>
94                 <td>{user.telefone}</td>
95                 <td>{user.logradouro}</td>
96                 <td>{user.cep}</td>
97                 <td>{user.bairro}</td>
98                 <td>{user.cidade}</td>
99                 <td>{user.numero}</td>
100               </tr>
101             ))}
102           </tbody>
103         </table>
104       </div>
105     </div>
106   );
107 }
108
109 export default ReportPage;
```



Demonstração de Uso

1. Tela Inicial - Gerenciamento de Biblioteca

Gerenciamento de Biblioteca



[Gerenciar Usuários](#) [Gerenciar Livros](#) [Gerenciar Empréstimos](#) [Gerenciar Relatórios](#)

2. Registro/Lista/Atualização/Deletar de Livro

Gerenciamento de Biblioteca

Registrar Livro

Título:

Autor:

Gênero: Data de Publicação: Estado do Livro:

Descrição:

Lista de Livros

Atualizar Livro

Título:

Autor:

Gênero: Data de Publicação: Estado do Livro:

Descrição:

ID do Livro:

Deletar Livro

ID do Livro:

3. Registro/Lista de Empréstimo e Devolução de Livro

Gerenciamento de Biblioteca

Registrar Empréstimo

Usuário ID:

Livro ID:

Data Limite:

Registrar Empréstimo

Lista de Empréstimos

Nenhum empréstimo ativo encontrado.

Devolução de Livro

ID Empréstimo:

Data de Devolução

Devolver Livro

4. Relatórios de Empréstimo e Usuários com Empréstimo Pendentes

Gerenciamento de Biblioteca

Relatórios

Livros Mais Empréstados

ID Livro	Título	Autor	Quantidade de Empréstimos
Nenhum dado disponível.			

Usuários com Empréstimos Pendentes

ID Usuário	Nome	Email	Status
Nenhum dado disponível.			

5. Cadastro/Lista de Usuário(s)

Gerenciamento de Biblioteca

Cadastro de Usuário

Nome:

Email:

Telefone:

Endereço

Logradouro: CEP: Bairro:

Cidade: Número:

Cadastrar

Lista de Usuários

ID Usuário	Nome	Email	Telefone	Endereço	Ações
------------	------	-------	----------	----------	-------

6. Cadastro/Lista de Usuário(s)

Gerenciamento de Biblioteca

Cadastro de Usuário

Nome:

Email:

Telefone:

Endereço

Logradouro: CEP: Bairro:

Cidade: Número:

Cadastrar

Lista de Usuários

ID Usuário	Nome	Email	Telefone	Endereço	Ações
1	teste	teste@gmail.com	813443434	aaaa, aaaa, aaaa	<p>Editar</p> <p>Deletar</p>

7. Registro/Lista de Empréstimo e Devolução de Livro

Gerenciamento de Biblioteca

Registrar Empréstimo

Usuário ID:

Livro ID:

Data Limite:

Registrar Empréstimo

Empréstimo registrado com sucesso!

Lista de Empréstimos

ID Empréstimo	Nome do Livro	Nome do Usuário	D.devolução	Status
1	teste	teste	2024-12-02	COM_LIVRO

Devolução de Livro

ID Empréstimo:

Data de Devolução:

Devolver Livro

8. Registro/Lista/Atualização/Deletar de Livro

Gerenciamento de Biblioteca

Relatórios

Livros Mais Empréstados

ID Livro	Título	Autor	Quantidade de Empréstimos
1	teste	test	1

Usuários com Empréstimos Pendentes

ID Usuário	Nome	Email	Status
Nenhum dado disponível.			

9. Registro/Lista/Atualização/Deletar de Livro

Gerenciamento de Biblioteca

Registrar Livro

Título:

Autor:

Gênero: Data de Publicação: Estado do Livro:

Descrição:

Lista de Livros

ID	Título	Autor	Gênero	Data de Publicação	Descrição	Estado
1	teste	tet	ACAO	2024-12-17	rereer	CONSERVADO

Atualizar Livro

Título:

Autor:

Gênero: Data de Publicação: Estado do Livro:

Descrição:

ID do Livro:

Deletar Livro

ID do 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, lembrar 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.