Frontend React da Livraria — Tutorial Completo

Disciplina: Desenvolvimento Web I
Curso: Ciência da Computação - 4ª fase
Professor: Fabricio Bizotto

Sumário

Introdução

Este tutorial apresenta a construção completa de um frontend React para o sistema de livraria, incluindo:

- **Vite** como ferramenta de build
- **React Router** para navegação SPA
- **Axios** para requisições HTTP
- **Context API** para gerenciamento de estado de autenticação
- **CRUD completo** de livros
- **Proteção de rotas** privadas
- **Comunicação com backend** via proxy e cookies

Criando o Projeto

Na raiz do projeto `livraria_node_http`, execute:

```
# Criar projeto React com Vite
npm create vite@latest frontend -- --template react

# Entrar na pasta frontend
cd frontend

# Instalar dependências base
npm install
```

> **Nota:** O Vite cria uma estrutura base com React 18, sendo mais rápido que o Create React App.

Instalando Dependências

Instale as bibliotecas adicionais necessárias:

```
npm install react-router-dom@6 axios
```

Dependências finais do projeto:

- `react` e `react-dom` (18.x) - biblioteca React
- `react-router-dom` (6.x) - roteamento SPA
- `axios` (1.x) - cliente HTTP
- `vite` (5.x) - build tool

Estrutura de Pastas

A estrutura completa do frontend será:

```
frontend/
├── src/
│   ├── components/
│   │   ├── Header.jsx
│   │   ├── Header.css
│   │   ├── LivroCard.jsx
│   │   ├── LivroCard.css
│   │   ├── LivroForm.jsx
│   │   ├── LivroForm.css
│   │   └── PrivateRoute.jsx
│   ├── contexts/
│   │   └── AuthContext.jsx
│   ├── pages/
│   │   ├── Login.jsx
│   │   ├── Register.jsx
│   │   ├── Auth.css
│   │   ├── Home.jsx
│   │   ├── Home.css
│   │   ├── Livros.jsx
│   │   └── Livros.css
│   ├── services/
│   │   ├── api.js
│   │   ├── authService.js
│   │   └── livrosService.js
│   ├── App.jsx
│   ├── App.css
│   ├── main.jsx
│   └── index.css
├── vite.config.js
├── index.html
└── package.json
```

Configurando Vite e Proxy

Edite o arquivo `vite.config.js`:

```
// frontend/vite.config.js
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  server: {
    port: 3000,
    proxy: {
      '/api': {
        target: 'http://localhost:3333',
        changeOrigin: true,
      }
    }
  }
})
```

Por que usar proxy?

- Redireciona requisições `/api/*` para o backend (porta 3333)
- Mantém sessão via cookies
- Evita problemas de CORS
- Sem proxy, seria necessário usar URLs completas como `http://localhost:3333/api`

Configurando Axios

Passo 1: Criar serviço base

Crie a pasta `src/services` e o arquivo `api.js`:

```
// frontend/src/services/api.js
import axios from 'axios';

const api = axios.create({
  baseURL: '/api',
  withCredentials: true,
  headers: { 'Content-Type': 'application/json' }
});

// Interceptor para tratamento de erros
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      // Redireciona para login apenas se não estiver em página pública
      const publicRoutes = ['/login', '/register'];
      const currentPath = window.location.pathname;
      if (!publicRoutes.includes(currentPath)) {
        window.location.href = '/login';
      }
    }
    return Promise.reject(error);
  }
);

export default api;
```

Configurações importantes:

- `baseURL: '/api'` - usa o proxy configurado no Vite
- `withCredentials: true` - envia cookies de sessão automaticamente
- **Interceptor 401** - redireciona para login quando não autenticado

---

Criando Serviços de API

Serviço de Autenticação

```
// frontend/src/services/authService.js
import api from './api';

export const authService = {
  async register(userData) {
    const response = await api.post('/auth/register', userData);
    return response.data;
  },

  async login(credentials) {
    const response = await api.post('/auth/login', credentials);
    return response.data;
  },

  async logout() {
    const response = await api.post('/auth/logout');
    return response.data;
  },

  async getMe() {
    const response = await api.get('/auth/me');
    return response.data;
  }
};
```

Serviço de Livros

```
// frontend/src/services/livrosService.js
import api from './api';

export const livrosService = {
  async listar() {
    const response = await api.get('/livros');
```

```
    return response.data;
  },

  async buscarPorId(id) {
    const response = await api.get(`/livros/${id}`);
    return response.data;
  },

  async criar(livro) {
    const response = await api.post('/livros', livro);
    return response.data;
  },

  async atualizar(id, livro) {
    const response = await api.put(`/livros/${id}`, livro);
    return response.data;
  },

  async remover(id) {
    const response = await api.delete(`/livros/${id}`);
    return response.data;
  }
};
```

Context API - AuthContext

Passo 1: Criar pasta de contextos

```
mkdir -p src/contexts
```

Passo 2: Criar AuthContext

O AuthContext gerencia o estado de autenticação globalmente, permitindo login, logout, registro e verificação do usuário atual.

```
// frontend/src/contexts/AuthContext.jsx
import React, { createContext, useState, useContext, useEffect } from 'react';
import { authService } from '../services/authService';

const AuthContext = createContext({});

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  // Verifica autenticação ao montar o contexto (executado uma vez)
  useEffect(() => {
    checkAuth();
  }, []);

  const checkAuth = async () => {
    try {
      const userData = await authService.getMe();
      setUser(userData);
    } catch (error) {
      setUser(null);
    } finally {
      setLoading(false);
    }
  };

  const login = async (credentials) => {
    const data = await authService.login(credentials);
    setUser(data.user);
    return data;
  };

  const register = async (userData) => {
    const data = await authService.register(userData);
    return data;
  };

  const logout = async () => {
```

```
    try {
      await authService.logout();
    } finally {
      setUser(null);
    }
  };

  return (
    <AuthContext.Provider value={{ user, loading, login, register, logout, checkAuth }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuth deve ser usado dentro de AuthProvider');
  }
  return context;
};
```

Configurando Rotas e App

App.jsx

```
// frontend/src/App.jsx
import React from 'react'
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom'
import { AuthProvider } from './contexts/AuthContext'
import PrivateRoute from './components/PrivateRoute'
import Header from './components/Header'
import Login from './pages/Login'
import Register from './pages/Register'
import Home from './pages/Home'
import Livros from './pages/Livros'
import './App.css'

function App() {
  return (
    <AuthProvider>
      <Router>
        <div className="app">
          <Header />
          <main className="main-content">
            <Routes>
              <Route path="/login" element={<Login />} />
              <Route path="/register" element={<Register />} />
              <Route path="/" element={<PrivateRoute><Home /></PrivateRoute>} />
              <Route path="/livros" element={<PrivateRoute><Livros /></PrivateRoute>} />
              <Route path="*" element={<Navigate to="/" replace />} />
            </Routes>
          </main>
        </div>
      </Router>
    </AuthProvider>
  )
}

export default App
```

App.css

```
/* frontend/src/App.css */
.app {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

.main-content {
```

```
  flex: 1;
  padding: 20px 0;
}
```

main.jsx

```
// frontend/src/main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

Estilos Globais

```
/* frontend/src/index.css */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', sans-serif;
  background-color: #f5f5f5;
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 20px;
}

.btn {
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 14px;
  transition: all 0.3s ease;
}

.btn-primary {
  background-color: #007bff;
  color: white;
}

.btn-primary:hover {
  background-color: #0056b3;
}

.btn-secondary {
  background-color: #6c757d;
  color: white;
}

.btn-secondary:hover {
  background-color: #5a6268;
}

.btn-success {
  background-color: #28a745;
  color: white;
}

.btn-success:hover {
```

```
  background-color: #218838;
}

.btn-danger {
  background-color: #dc3545;
  color: white;
}

.btn-danger:hover {
  background-color: #c82333;
}

.input-group {
  margin-bottom: 15px;
}

.input-group label {
  display: block;
  margin-bottom: 5px;
  font-weight: 500;
}

.input-group input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.alert {
  padding: 12px 20px;
  border-radius: 4px;
  margin-bottom: 20px;
}

.alert-error {
  background-color: #f8d7da;
  color: #721c24;
}

.alert-success {
  background-color: #d4edda;
  color: #155724;
}

.loading {
  text-align: center;
  padding: 40px;
}
```

Componentes

1. PrivateRoute

Componente que protege rotas privadas, permitindo acesso apenas para usuários autenticados.

```
mkdir -p src/components
```

```
// frontend/src/components/PrivateRoute.jsx
import React from 'react';
import { Navigate } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';

const PrivateRoute = ({ children }) => {
  const { user, loading } = useAuth();

  if (loading) {
    return (
      <div className="loading">
        <p>Carregando...</p>
      </div>
    );
```

```
  }

  return user ? children : <Navigate to="/login" replace />;
};

export default PrivateRoute;
```

2. Header

```
// frontend/src/components/Header.jsx
import React from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import './Header.css';

const Header = () => {
  const { user, logout } = useAuth();
  const navigate = useNavigate();

  const handleLogout = async () => {
    await logout();
    navigate('/login');
  };

  return (
    <header className="header">
      <div className="container header-content">
        <Link to="/" className="logo">
          <h1>📚 Livraria</h1>
        </Link>

        <nav className="nav">
          {user ? (
            <>
              <Link to="/" className="nav-link">Início</Link>
              <Link to="/livros" className="nav-link">Livros</Link>
              <div className="user-info">
                <span>Olá, {user.username || user.email}!</span>
                <button onClick={handleLogout} className="btn btn-secondary">
                  Sair
                </button>
              </div>
            </>
          ) : (
            <>
              <Link to="/login" className="nav-link">Login</Link>
              <Link to="/register" className="nav-link">Registrar</Link>
            </>
          )}
        </nav>
      </div>
    </header>
  );
};

export default Header;
```

```
/* frontend/src/components/Header.css */
.header {
  background-color: #fff;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  padding: 1rem 0;
}

.header-content {
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

```css
.logo {
  text-decoration: none;
  color: #333;
}

.logo h1 {
  font-size: 24px;
  margin: 0;
}

.nav {
  display: flex;
  align-items: center;
  gap: 20px;
}

.nav-link {
  text-decoration: none;
  color: #333;
  font-weight: 500;
  transition: color 0.3s ease;
}

.nav-link:hover {
  color: #007bff;
}

.user-info {
  display: flex;
  align-items: center;
  gap: 15px;
}

.user-info span {
  color: #666;
  font-size: 14px;
}
```

3. LivroCard

```jsx
// frontend/src/components/LivroCard.jsx
import React from 'react';
import './LivroCard.css';

const LivroCard = ({ livro, onEdit, onDelete }) => {
  return (
    <div className="livro-card">
      <h3>{livro.titulo}</h3>
      <p><strong>Autor:</strong> {livro.autor}</p>
      <p><strong>Ano:</strong> {livro.ano}</p>
      {livro.editora && <p><strong>Editora:</strong> {livro.editora}</p>}

      <div className="card-actions">
        <button onClick={() => onEdit(livro)} className="btn btn-primary">
          ✏ Editar
        </button>
        <button onClick={() => onDelete(livro.id)} className="btn btn-danger">
          🗑 Remover
        </button>
      </div>
    </div>
  );
};

export default LivroCard;
```

```css
/* frontend/src/components/LivroCard.css */
.livro-card {
  background: white;
  border-radius: 8px;
  padding: 20px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  transition: transform 0.2s ease, box-shadow 0.2s ease;
```

```
}

.livro-card:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(0,0,0,0.15);
}

.livro-card h3 {
  margin: 0 0 15px 0;
  font-size: 20px;
  color: #333;
}

.livro-card p {
  margin: 8px 0;
  color: #666;
  font-size: 14px;
}

.card-actions {
  margin-top: 20px;
  display: flex;
  gap: 10px;
}

.card-actions .btn {
  flex: 1;
  font-size: 13px;
  padding: 8px 12px;
}
```

4. LivroForm

```
// frontend/src/components/LivroForm.jsx
import React, { useState, useEffect } from 'react';
import './LivroForm.css';

const LivroForm = ({ livro, onSubmit, onCancel }) => {
  const [formData, setFormData] = useState({
    titulo: '',
    autor: '',
    ano: '',
    editora: ''
  });

  useEffect(() => {
    if (livro) {
      setFormData({
        titulo: livro.titulo || '',
        autor: livro.autor || '',
        ano: livro.ano || '',
        editora: livro.editora || ''
      });
    }
  }, [livro]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    onSubmit(formData);
  };

  return (
    <div className="livro-form-overlay">
      <div className="livro-form-container">
        <h2>{livro ? 'Editar Livro' : 'Novo Livro'}</h2>
        <form onSubmit={handleSubmit}>
          <div className="input-group">
            <label htmlFor="titulo">Título *</label>
            <input
              type="text"
```

```
              id="titulo"
              name="titulo"
              value={formData.titulo}
              onChange={handleChange}
              required
            />
          </div>

          <div className="input-group">
            <label htmlFor="autor">Autor *</label>
            <input
              type="text"
              id="autor"
              name="autor"
              value={formData.autor}
              onChange={handleChange}
              required
            />
          </div>

          <div className="input-group">
            <label htmlFor="ano">Ano *</label>
            <input
              type="number"
              id="ano"
              name="ano"
              value={formData.ano}
              onChange={handleChange}
              required
              min="1000"
              max="9999"
            />
          </div>

          <div className="input-group">
            <label htmlFor="editora">Editora</label>
            <input
              type="text"
              id="editora"
              name="editora"
              value={formData.editora}
              onChange={handleChange}
            />
          </div>

          <div className="form-actions">
            <button type="button" onClick={onCancel} className="btn btn-secondary">
              Cancelar
            </button>
            <button type="submit" className="btn btn-success">
              {livro ? 'Atualizar' : 'Criar'}
            </button>
          </div>
        </form>
      </div>
    </div>
  );
};

export default LivroForm;
```

```
/* frontend/src/components/LivroForm.css */
.livro-form-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0,0,0,0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
}
```

```css
.livro-form-container {
  background: white;
  padding: 30px;
  border-radius: 8px;
  width: 90%;
  max-width: 500px;
  max-height: 90vh;
  overflow-y: auto;
}

.livro-form-container h2 {
  margin: 0 0 20px 0;
  text-align: center;
}

.form-actions {
  display: flex;
  gap: 10px;
  margin-top: 20px;
}

.form-actions .btn {
  flex: 1;
}
```

Páginas

1. Login

```
mkdir -p src/pages
```

```jsx
// frontend/src/pages/Login.jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import './Auth.css';

const Login = () => {
  const [formData, setFormData] = useState({
    email: '',
    password: ''
  });
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const { login, user } = useAuth();
  const navigate = useNavigate();

  // Redireciona se já estiver autenticado
  useEffect(() => {
    if (user) {
      navigate('/');
    }
  }, [user, navigate]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');
    setLoading(true);

    try {
      await login(formData);
      navigate('/');
    } catch (err) {
      setError(err.response?.data?.erro || 'Erro ao fazer login.');
    } finally {
      setLoading(false);
    }
```

```
  };

  return (
    <div className="auth-container">
      <div className="auth-card">
        <h2>Login</h2>
        {error && <div className="alert alert-error">{error}</div>}

        <form onSubmit={handleSubmit}>
          <div className="input-group">
            <label htmlFor="email">Email</label>
            <input
              type="email"
              id="email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              required
              disabled={loading}
            />
          </div>

          <div className="input-group">
            <label htmlFor="password">Senha</label>
            <input
              type="password"
              id="password"
              name="password"
              value={formData.password}
              onChange={handleChange}
              required
              disabled={loading}
            />
          </div>

          <button type="submit" className="btn btn-primary btn-block" disabled={loading}>
            {loading ? 'Entrando...' : 'Entrar'}
          </button>
        </form>

        <p className="auth-link">
          Não tem uma conta? <Link to="/register">Registre-se</Link>
        </p>
      </div>
    </div>
  );
};

export default Login;
```

2. Register

```
// frontend/src/pages/Register.jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import './Auth.css';

const Register = () => {
  const [formData, setFormData] = useState({
    username: '',
    email: '',
    password: '',
    confirmPassword: ''
  });
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const { register, user } = useAuth();
  const navigate = useNavigate();

  useEffect(() => {
    if (user) {
      navigate('/');
    }
  }, [user, navigate]);
```

```
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');

    if (formData.password !== formData.confirmPassword) {
      setError('As senhas não coincidem');
      return;
    }

    setLoading(true);
    try {
      const { confirmPassword, ...registerData } = formData;
      await register(registerData);
      navigate('/login');
    } catch (err) {
      setError(err.response?.data?.erro || 'Erro ao criar conta.');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="auth-container">
      <div className="auth-card">
        <h2>Registrar</h2>
        {error && <div className="alert alert-error">{error}</div>}

        <form onSubmit={handleSubmit}>
          <div className="input-group">
            <label htmlFor="username">Nome de usuário</label>
            <input
              type="text"
              id="username"
              name="username"
              value={formData.username}
              onChange={handleChange}
              required
              disabled={loading}
            />
          </div>

          <div className="input-group">
            <label htmlFor="email">Email</label>
            <input
              type="email"
              id="email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              required
              disabled={loading}
            />
          </div>

          <div className="input-group">
            <label htmlFor="password">Senha</label>
            <input
              type="password"
              id="password"
              name="password"
              value={formData.password}
              onChange={handleChange}
              required
              minLength="6"
              disabled={loading}
            />
          </div>

          <div className="input-group">
            <label htmlFor="confirmPassword">Confirmar Senha</label>
            <input
```

```
              type="password"
              id="confirmPassword"
              name="confirmPassword"
              value={formData.confirmPassword}
              onChange={handleChange}
              required
              minLength="6"
              disabled={loading}
            />
          </div>

          <button type="submit" className="btn btn-primary btn-block" disabled={loading}>
            {loading ? 'Criando conta...' : 'Registrar'}
          </button>
        </form>

        <p className="auth-link">
          Já tem uma conta? <Link to="/login">Faça login</Link>
        </p>
      </div>
    </div>
  );
};

export default Register;
```

Estilos de Autenticação

```
/* frontend/src/pages/Auth.css */
.auth-container {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: calc(100vh - 80px);
  padding: 20px;
}

.auth-card {
  background: white;
  padding: 40px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
  width: 100%;
  max-width: 400px;
}

.auth-card h2 {
  margin: 0 0 20px 0;
  text-align: center;
}

.btn-block {
  width: 100%;
  margin-top: 10px;
}

.auth-link {
  text-align: center;
  margin-top: 20px;
  font-size: 14px;
}

.auth-link a {
  color: #007bff;
  text-decoration: none;
}

.auth-link a:hover {
  text-decoration: underline;
}
```

3. Home

```jsx
// frontend/src/pages/Home.jsx (continuação)
import React from 'react';
import { Link } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import './Home.css';

const Home = () => {
  const { user } = useAuth();

  return (
    <div className="container">
      <div className="home-container">
        <div className="welcome-card">
          <h1>Bem-vindo ao Sistema de Gerenciamento de Livraria! 📚</h1>
          <p className="subtitle">
            Olá, <strong>{user?.username || user?.email}</strong>!
          </p>
          <p>Sistema completo para gerenciar sua coleção de livros.</p>

          <div className="cta">
            <Link to="/livros" className="btn btn-primary btn-large">
              Ver Meus Livros
            </Link>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Home;
```

```css
/* frontend/src/pages/Home.css */
.home-container {
  max-width: 1000px;
  margin: 0 auto;
}

.welcome-card {
  background: white;
  padding: 40px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
  text-align: center;
}

.welcome-card h1 {
  font-size: 32px;
  margin-bottom: 15px;
  color: #333;
}

.subtitle {
  font-size: 18px;
  color: #666;
  margin-bottom: 20px;
}

.welcome-card p {
  font-size: 16px;
  color: #666;
  margin-bottom: 15px;
}

.cta {
  margin-top: 30px;
}

.btn-large {
  font-size: 18px;
  padding: 15px 40px;
  text-decoration: none;
  display: inline-block;
}
```

4. Livros (CRUD Completo)

```
// frontend/src/pages/Livros.jsx
import React, { useState, useEffect } from 'react';
import { livrosService } from '../services/livrosService';
import LivroCard from '../components/LivroCard';
import LivroForm from '../components/LivroForm';
import './Livros.css';

const Livros = () => {
  const [livros, setLivros] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState('');
  const [showForm, setShowForm] = useState(false);
  const [editingLivro, setEditingLivro] = useState(null);
  const [successMessage, setSuccessMessage] = useState('');

  useEffect(() => {
    carregarLivros();
  }, []);

  const carregarLivros = async () => {
    try {
      setLoading(true);
      setError('');
      const data = await livrosService.listar();
      setLivros(data);
    } catch (err) {
      setError('Erro ao carregar livros.');
      console.error(err);
    } finally {
      setLoading(false);
    }
  };

  const handleCreate = () => {
    setEditingLivro(null);
    setShowForm(true);
  };

  const handleEdit = (livro) => {
    setEditingLivro(livro);
    setShowForm(true);
  };

  const handleDelete = async (id) => {
    if (!window.confirm('Tem certeza que deseja remover este livro?')) {
      return;
    }

    try {
      await livrosService.remover(id);
      showSuccess('Livro removido com sucesso!');
      carregarLivros();
    } catch (err) {
      setError('Erro ao remover livro.');
      console.error(err);
    }
  };

  const handleSubmit = async (formData) => {
    try {
      if (editingLivro) {
        await livrosService.atualizar(editingLivro.id, formData);
        showSuccess('Livro atualizado com sucesso!');
      } else {
        await livrosService.criar(formData);
        showSuccess('Livro criado com sucesso!');
      }
      setShowForm(false);
      setEditingLivro(null);
      carregarLivros();
    } catch (err) {
      setError(err.response?.data?.erro || 'Erro ao salvar livro.');
      console.error(err);
```

```
    }
  };

  const handleCancel = () => {
    setShowForm(false);
    setEditingLivro(null);
  };

  const showSuccess = (message) => {
    setSuccessMessage(message);
    setTimeout(() => setSuccessMessage(''), 3000);
  };

  if (loading) {
    return <div className="loading">Carregando livros...</div>;
  }

  return (
    <div className="container">
      <div className="livros-header">
        <h1>Meus Livros</h1>
        <button onClick={handleCreate} className="btn btn-primary">
          + Adicionar Livro
        </button>
      </div>

      {successMessage && (
        <div className="alert alert-success">{successMessage}</div>
      )}

      {error && (
        <div className="alert alert-error">{error}</div>
      )}

      {livros.length === 0 ? (
        <div className="empty-state">
          <p>Nenhum livro cadastrado ainda.</p>
          <button onClick={handleCreate} className="btn btn-primary">
            Adicionar seu primeiro livro
          </button>
        </div>
      ) : (
        <div className="livros-grid">
          {livros.map((livro) => (
            <LivroCard
              key={livro.id}
              livro={livro}
              onEdit={handleEdit}
              onDelete={handleDelete}
            />
          ))}
        </div>
      )}

      {showForm && (
        <LivroForm
          livro={editingLivro}
          onSubmit={handleSubmit}
          onCancel={handleCancel}
        />
      )}
    </div>
  );
};

export default Livros;
```

```
/* frontend/src/pages/Livros.css */
.livros-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 30px;
}
```

```css
.livros-header h1 {
  margin: 0;
  font-size: 28px;
}

.livros-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
  margin-top: 20px;
}

.empty-state {
  text-align: center;
  padding: 60px 20px;
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.empty-state p {
  font-size: 18px;
  color: #666;
  margin-bottom: 20px;
}
```

Executando o Projeto

Passo 1: Iniciar o Backend

Abra um terminal na raiz do projeto e execute:

```
# Na raiz do projeto livraria_node_http
npm run dev
```

O backend estará rodando em: **http://localhost:3333**

Passo 2: Iniciar o Frontend

Abra um **segundo terminal** e execute:

```
# Entrar na pasta frontend
cd frontend

# Iniciar o servidor de desenvolvimento
npm run dev
```

O frontend estará rodando em: **http://localhost:3000**

> **Nota:** Certifique-se de que o backend está rodando antes de iniciar o frontend, pois a aplicação depende da API.

Testando a Aplicação

1. Registro de Novo Usuário

1. Acesse: **http://localhost:3000**
2. Clique em **"Registrar"** no menu superior
3. Preencha o formulário:
   - Nome de usuário: `joao`
   - Email: `joao@example.com`
   - Senha: `senha123`
   - Confirmar Senha: `senha123`
4. Clique em **"Registrar"**
5. Você será redirecionado para a página de login

2. Login

1. Na página de login, insira:

- Email: `joao@example.com`
- Senha: `senha123`

2. Clique em **"Entrar"**
3. Você será redirecionado para a página inicial

3. Navegação

- Veja a mensagem de boas-vindas com seu nome de usuário
- Note o menu superior com as opções **"Início"** e **"Livros"**
- Seu nome de usuário aparecerá no canto superior direito

4. Gerenciar Livros

Adicionar um Livro

1. Clique em **"Livros"** no menu
2. Clique em **"➕ Adicionar Livro"**
3. Preencha o formulário:
   - Título: `Clean Code`
   - Autor: `Robert C. Martin`
   - Ano: `2008`
   - Editora: `Prentice Hall`
4. Clique em **"Criar"**
5. Veja a mensagem de sucesso e o livro aparecerá na lista

Editar um Livro

1. Clique no botão **"✏ Editar"** em um livro
2. Modifique os campos desejados
3. Clique em **"Atualizar"**
4. As alterações serão salvas

Remover um Livro

1. Clique no botão **"🗑 Remover"** em um livro
2. Confirme a exclusão no diálogo
3. O livro será removido da lista

5. Logout

1. Clique no botão **"Sair"** no canto superior direito
2. Você será deslogado e redirecionado para a página de login
3. Tente acessar rotas protegidas (como `/livros`) - você será redirecionado para login

---

Estrutura Final do Projeto

```
livraria_node_http/
├── backend/
│   ├── src/
│   │   ├── controllers/
│   │   │   ├── authController.js
│   │   │   └── livrosController.js
│   │   ├── models/
│   │   │   ├── User.js
│   │   │   └── Livro.js
│   │   ├── repositories/
│   │   │   ├── userRepository.js
│   │   │   └── livrosRepository.js
│   │   ├── routes/
│   │   │   ├── authRoutes.js
│   │   │   └── livrosRoutes.js
│   │   └── middlewares/
│   │       └── authMiddleware.js
│   ├── server.js
│   └── package.json
│
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   ├── Header.jsx / Header.css
│   │   │   ├── LivroCard.jsx / LivroCard.css
```

```
│   │   │   ├── LivroForm.jsx / LivroForm.css
│   │   │   └── PrivateRoute.jsx
│   │   ├── contexts/
│   │   │   └── AuthContext.jsx
│   │   ├── pages/
│   │   │   ├── Login.jsx
│   │   │   ├── Register.jsx
│   │   │   ├── Auth.css
│   │   │   ├── Home.jsx / Home.css
│   │   │   ├── Livros.jsx
│   │   │   └── Livros.css
│   │   ├── services/
│   │   │   ├── api.js
│   │   │   ├── authService.js
│   │   │   └── livrosService.js
│   │   ├── App.jsx / App.css
│   │   ├── main.jsx
│   │   └── index.css
│   ├── vite.config.js
│   ├── index.html
│   └── package.json
│
└── package.json (root)
```

Recursos Implementados

✅ Funcionalidades

- **Autenticação completa**
  - Registro de novos usuários
  - Login com email e senha
  - Logout
  - Sessão mantida via cookies httpOnly
- **Gerenciamento de Livros (CRUD)**
  - Listar todos os livros
  - Criar novo livro
  - Editar livro existente
  - Remover livro
- **Proteção de Rotas**
  - Rotas privadas acessíveis apenas para usuários autenticados
  - Redirecionamento automático para login quando não autenticado
- **Interface Responsiva**
  - Design moderno e limpo
  - Grid responsivo para listagem de livros
  - Formulários modais
  - Mensagens de feedback (sucesso/erro)

✅ Tecnologias Utilizadas

- **Vite** - Build tool rápida e moderna
- **React 18** - Biblioteca JavaScript para UI
- **React Router 6** - Roteamento SPA
- **Axios** - Cliente HTTP
- **Context API** - Gerenciamento de estado global
- **CSS Modules** - Estilização componentizada

Conceitos Importantes

1. Proxy do Vite

O proxy configurado no `vite.config.js` permite que requisições para `/api` sejam automaticamente redirecionadas para `http://localhost:3333`. Isso evita problemas de CORS e mantém a sessão via cookies.

Sem proxy:

```
await axios.get('http://localhost:3333/api/livros')
```

Com proxy:

```
await axios.get('/api/livros')
```

2. Interceptor Axios

O interceptor no `api.js` captura todas as respostas HTTP com status 401 (não autorizado) e redireciona automaticamente para a página de login, exceto quando já está em uma rota pública.

3. Context API

O `AuthContext` fornece um estado global de autenticação que pode ser acessado por qualquer componente da aplicação usando o hook `useAuth()`.

4. PrivateRoute

Componente que envolve rotas protegidas, verificando se o usuário está autenticado antes de renderizar o conteúdo. Se não estiver autenticado, redireciona para login.

5. withCredentials

A configuração `withCredentials: true` no Axios garante que os cookies de sessão sejam enviados automaticamente em todas as requisições, mantendo a autenticação persistente.

Próximos Passos

Melhorias que podem ser implementadas:

1. **Paginação** - Adicionar paginação na listagem de livros
2. **Busca e Filtros** - Implementar busca por título/autor
3. **Validações** - Adicionar validações mais robustas nos formulários
4. **Loading States** - Melhorar estados de carregamento
5. **Tratamento de Erros** - Mensagens de erro mais específicas
6. **Testes** - Adicionar testes unitários e de integração
7. **Deploy** - Preparar aplicação para produção

Resumo

Neste tutorial, você aprendeu a:

✅ Criar um projeto React moderno com Vite
✅ Configurar proxy para comunicação com backend
✅ Implementar autenticação com Context API
✅ Proteger rotas privadas
✅ Criar um CRUD completo de livros
✅ Utilizar Axios para requisições HTTP
✅ Gerenciar estado global com Context API
✅ Criar componentes reutilizáveis
✅ Estilizar aplicação com CSS moderno
✅ Manter sessão via cookies httpOnly

O resultado é uma aplicação full-stack funcional com autenticação e gerenciamento de dados!