⚛️ Frontend React da Livraria — Parte 7

Vite + React + React Router + Axios

👨‍🏫 **Professor:** Fabricio Bizotto

📘 **Disciplina:** Desenvolvimento Web I

🎓 **Curso:** Ciência da Computação

📅 **Fase:** 4ª fase

Roteiro

- Criando o projeto do zero com Vite
- Instalando dependências
- Estrutura de pastas
- Configurando Vite + Proxy
- Axios (instância, interceptor de 401)
- Contexto de Autenticação (AuthContext)
- Protegendo rotas (PrivateRoute)
- Páginas: Login, Register, Home, Livros (CRUD)
- Componentes reutilizáveis
- Serviços de API (authService, livrosService)
- Executando o frontend

Criando o projeto React com Vite

```
# Na raiz do projeto livraria_node_http
npm create vite@latest frontend -- --template react

# Entrar na pasta frontend
cd frontend

# Instalar dependências base
npm install
```

> Vite cria estrutura base com React 18, mais rápido que Create React App.

Instalando dependências adicionais

```
# Instalar React Router e Axios
npm install react-router-dom@6 axios
```

Dependências finais:

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

Estrutura de pastas

```
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 + Proxy

```
// 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,
      }
    }
  }
})
```

> Proxy redireciona `/api/*` para backend (3333), mantém sessão via cookies.
>
> Sem proxy, precisaria usar URL completa `http://localhost:3333/api` no Axios. Com proxy, basta usar `/api`.

Limpando arquivos padrão do Vite

```
# Remover arquivos de exemplo
rm src/App.css src/index.css

# Manter apenas: main.jsx, App.jsx, index.html
```

Estrutura inicial:

```
frontend/
├── src/
│   ├── main.jsx    # Entry point
│   └── App.jsx     # Componente raiz
├── index.html
└── vite.config.js
```

Criando serviço Axios base

Crie a pasta `src/services` e o arquivo `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;
```

Serviços 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;
  }
};
```

Context API: AuthContext

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

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

Context API: AuthContext (parte 1/2)

```jsx
// 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. Isso é feito uma vez.
  useEffect(() => { checkAuth(); }, []);

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

Context API: AuthContext (parte 2/2)

```
  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;
};
```

Criando Componentes

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

Componente PrivateRoute

```
// 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;
```

> Apenas usuários autenticados podem acessar rotas protegidas.

Criando estilos globais (parte 1/2)

```
/* 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;
}

.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;
}
```

Criando estilos globais (parte 2/2)

```
.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;
}

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

Configurando App.jsx e rotas

```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
```

Componente Header

```jsx
// 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;
```

Página de Login

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

```
// 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">
```

Página de Registro

```jsx
// 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;
```

Página Home

```jsx
// frontend/src/pages/Home.jsx
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;
```

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;
  }
};
```

Página de Livros (CRUD) - Parte 1

```jsx
// 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.');
    } 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.');
    }
  };

  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.');
    }
  };

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

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

  if (loading) return <div className="loading">Carregando livros...</div>;
  // continua no próximo slide...
```

Página de Livros (CRUD) - Parte 2

```
  // ...continuação de Livros.jsx
  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;
```

Componente 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;
```

Componente LivroForm

```jsx
// 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;
```

Atualizando 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>,
)
```

Executando o projeto completo

Terminal 1 - Backend:

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

> Backend rodando em http://localhost:3333

Terminal 2 - Frontend:

```
cd frontend
npm run dev
```

> Frontend rodando em http://localhost:3000

Testando a aplicação

1. Acesse: http://localhost:3000
2. Clique em "Registrar"
3. Preencha: username, email, senha
4. Faça login com email e senha
5. Navegue para "Livros"
6. Adicione, edite e remova livros

> Sessão mantida via cookies httpOnly do backend.

Estrutura final do projeto

```
livraria_node_http/
├── backend/
│   ├── src/
│   │   ├── controllers/
│   │   ├── models/
│   │   ├── repositories/
│   │   ├── routes/
│   │   └── ...
│   └── server.js
│
├── frontend/
│   ├── src/
│   │   ├── components/    # Header, LivroCard, LivroForm, PrivateRoute
│   │   ├── contexts/      # AuthContext
│   │   ├── pages/         # Login, Register, Home, Livros
│   │   ├── services/      # api, authService, livrosService
│   │   ├── App.jsx
│   │   └── main.jsx
│   ├── vite.config.js
│   └── package.json
└── package.json
```

Resumo

- ✅ Projeto criado do zero com Vite
- ✅ Proxy configurado para backend (porta 3333)
- ✅ Axios com interceptor 401
- ✅ Context API para autenticação
- ✅ Rotas públicas e privadas
- ✅ CRUD completo de livros
- ✅ Componentes reutilizáveis
- ✅ Sessão via cookies httpOnly

> Próximos passos: paginação, busca/filtro, testes, deploy.