Documentação do Projeto — Sistema de Agendamento

Escola: EEEP Deputado Roberto Mesquita

Disciplina: Qualidade e Teste de Software

Alunas(os): Glória Maria e Ana Luiza Bezerra

Docente: Everson Sousa

18.06.2025

SUMÁRIO

Introdução

A qualidade de software é fundamental para garantir que os sistemas atendam às necessidades dos usuários, sejam confiáveis, eficientes e de fácil utilização. O uso de testes automatizados auxilia na detecção de erros, melhora a manutenção do código e assegura maior robustez na entrega do software.

Este projeto foi escolhido por refletir uma necessidade cotidiana: organizar compromissos de forma prática. Assim, além de treinar desenvolvimento com React, as alunas puderam aplicar conceitos de testes e boas práticas no desenvolvimento de software.

O objetivo da aplicação é desenvolver um sistema de agendamento de compromissos, onde o usuário pode registrar um compromisso com título, data e horário, visualizando uma lista organizada de atividades.

Palavras-chave: Agendamento, Qualidade de Software, Testes Automatizados, React, Interface.

Introduction

Software quality is essential to ensure that systems meet users' needs, are reliable, efficient, and easy to use. Automated testing plays a crucial role in identifying errors, improving code maintenance, and ensuring robust software delivery.

This project was chosen because it addresses a common daily need: organizing appointments in a practical way. Therefore, this project not only reinforces knowledge in React development but also applies concepts of testing and software quality best practices.

The objective of this application is to develop an appointment scheduling system, where users can register appointments with a title, date, and time, and visualize an organized list of tasks.

Keywords: Scheduling, Software Quality, Automated Testing, React, Interface.

Descrição da Aplicação

A aplicação consiste em um Sistema de Agendamento de Compromissos, que permite ao usuário cadastrar compromissos informando título, data e hora, além de visualizar todos os compromissos cadastrados.

Público-alvo: estudantes, profissionais ou qualquer pessoa que deseje se organizar com uma agenda simples e funcional.

Funcionalidades principais:

- **Inserção de compromissos com título, data e horário;**
- **Visualização dos compromissos cadastrados;**
- **Interface simples e objetiva.**

Requisitos

Requisitos Funcionais (RF)

- **RF01 — Permitir que o usuário insira o título do compromisso.**
- **RF02 — Permitir que o usuário selecione a data.**
- **RF03 — Permitir que o usuário selecione o horário.**
- **RF04 — Adicionar o compromisso à lista após clicar em "Adicionar".**
- **RF05 — Exibir a lista de compromissos cadastrados.**

Requisitos Não Funcionais (RNF)

- **RNF01 — Interface responsiva e acessível.**
- **RNF02 — Tempo de resposta imediato após adicionar um compromisso.**
- **RNF03 — Interface limpa, objetiva e de fácil utilização.**
- **RNF04 — Código organizado e com boas práticas de desenvolvimento com React.**

Arquitetura da Aplicação

Tecnologias Utilizadas

- **React — Biblioteca JavaScript para desenvolvimento de interfaces.**
- **Vite — Ferramenta para desenvolvimento rápido.**
- **Jest + Testing Library — Ferramentas para testes automatizados.**
- **CSS — Estilização da interface.**
- **Node.js + npm (node_modules) — Gerenciamento de pacotes e dependências.**

Estrutura de Pastas

```
/src
 ├── /tests
 │    └── App.test.jsx
 ├── App.jsx
 ├── App.css
 ├── main.jsx
```

Componentização

Por se tratar de um projeto simples, foi utilizado apenas um componente principal (App.jsx), que concentra toda a lógica e a interface da aplicação. Em projetos maiores, poderia ser realizada a separação dos seguintes componentes:

- **InputForm.jsx — formulário de cadastro de compromissos;**
- **ScheduleList.jsx — lista de compromissos cadastrados;**
- **ScheduleItem.jsx — exibição de cada compromisso individualmente.**

Processo de Qualidade e Boas Práticas

- **O código foi estruturado utilizando React Hooks (useState) para gerenciamento dos estados da aplicação.**
- **Foram utilizadas boas práticas de nomenclatura, com nomes claros e descritivos, como `title`, `date`, `time` e `appointments`.**
- **A interface apresenta layout limpo, com campos de fácil interação e botões bem posicionados.**
- **O resultado da operação, que é a lista de compromissos, é atualizado automaticamente após cada adição.**
- **Foram aplicados princípios de acessibilidade, como labels, placeholders descritivos e contraste adequado.**

Testes Automatizados

Ferramentas Utilizadas

- **React**
- **Vite**
- **Jest + Testing Library**
- **CSS**
- **Node.js + npm (node_modules)**

Tipos de Testes Aplicados

- **Testes de unidade — verificação de funções isoladas, se aplicável;**
- **Testes de componentes — renderização de inputs, botão e lista de compromissos.**

Descrição dos Principais Testes

Teste	Descrição
Verificar se os inputs estão na tela	Testa se os campos de título, data e horário aparecem corretamente.
Verificar se o botão "Adicionar" existe	Garante que o botão foi renderizado corretamente.
Permitir digitar no input de título	Testa se o input aceita entrada de texto.
Permitir selecionar data e hora	Verifica se os campos de data e hora funcionam corretamente.
Efetuar cadastro de compromisso	Insere dados e verifica se o compromisso aparece na lista.
Verificar múltiplos cadastros	Garante que o sistema permite cadastrar mais de um compromisso.
Impedir cadastro com campo vazio	Testa se o sistema impede cadastro quando algum campo está vazio (se implementado).
Conferir se o compromisso aparece na lista correta	Valida que o compromisso cadastrado aparece corretamente na lista exibida.
Testar a atualização da lista após novo cadastro	Garante que a lista se atualiza dinamicamente após cada novo compromisso adicionado.
Testar usabilidade dos inputs	Verifica foco, digitação e interação dos campos da interface.

Conclusão

O desenvolvimento deste projeto proporcionou às alunas aprendizado prático sobre:

- **Criação de interfaces utilizando a biblioteca React;**
- **Aplicação de testes automatizados com Jest e Testing Library;**
- **Organização de código, adoção de boas práticas e compromisso com a qualidade no desenvolvimento de software.**

A aplicação atendeu aos objetivos propostos, sendo funcional, prática e devidamente testada. Como possíveis melhorias futuras, sugere-se a implementação de funcionalidades como edição e remoção de compromissos, filtros por data e armazenamento local dos dados (localStorage).

Referências

REACT. Documentação oficial. Disponível em: https://react.dev. Acesso em: 10 jun. 2025.

TESTING LIBRARY. Documentação oficial. Disponível em: https://testing-library.com. Acesso em: 10 jun. 2025.

JEST. Documentação oficial. Disponível em: https://jestjs.io. Acesso em: 10 jun. 2025.

MDN WEB DOCS. Documentação de HTML, CSS e JavaScript. Disponível em: https://developer.mozilla.org. Acesso em: 10 jun. 2025.

W3SCHOOLS. CSS Tutorial. Disponível em: https://www.w3schools.com/css/. Acesso em: 10 jun. 2025.

Sistema de Agendamento. Disponível em: https://sistema-de-agendamento-projeto-de-q.vercel.app. Acesso em: 10 jun. 2025.