

CompraSync

Integrantes: Gustavo Rafá, Igor Alvarenga, Jean Estanislau, João
Vitor de Souza, Otávio Soares e Pedro Henrique Morais

Sprint 1

Atualizado em: 21/03/2025

Responsável	Tarefa/Requisito	Iniciado em	Prazo	Status	Terminado em
Otávio Soares da Costa e Igor Alvarenga Nunes de Brito	Introdução do Projeto	07/03/2025	23/03/2025	✓	11/03/2025
Gustavo Rafá Pinheiro da Silva e Pedro Henrique Morais de Assis	Especificação do Projeto	11/03/2025	23/03/2025	✓	14/03/2025
João Vítor de Souza França	Arquitetura da Solução	14/03/2025	23/03/2025	✓	21/03/2025
Jean Estanislau de Souza Guimarães	Planejamento do Projeto	21/03/2025	23/03/2025	✓	21/03/2025

Sprint 2

Atualizado em: 12/06/2025

Responsável	Tarefa/Requisito	Iniciado em	Prazo	Status	Terminado em
Igor Alvarenga Nunes de Brito	Páginas de Login e Registro	01/05/2025	23/05/2025	✓	23/05/2025
Gustavo Rafá Pinheiro da Silva	Menu de Listas de Compras e Criação de Casos de Teste	01/05/2025	12/06/2025	✓	12/06/2025
Otávio Soares da Costa	Classes Front-End para consumo das APIs	01/05/2025	23/05/2025	✓	23/05/2025
Pedro Henrique Morais de Assis	Lista de Compras, Adequações e Liderança Técnica	01/05/2025	23/05/2025	✓	23/05/2025
João Vítor de Souza França	Construção da API e Liderança Técnica	01/05/2025	23/05/2025	✓	23/05/2025
Jean Estanislau de Souza Guimarães	Construção do Banco de Dados e Liderança	01/05/2025	23/05/2025	✓	21/05/2025

Temas

Tópicos

1 Introdução

2 Projeto do Sistema

3 Desenvolvimento

4 Conclusão

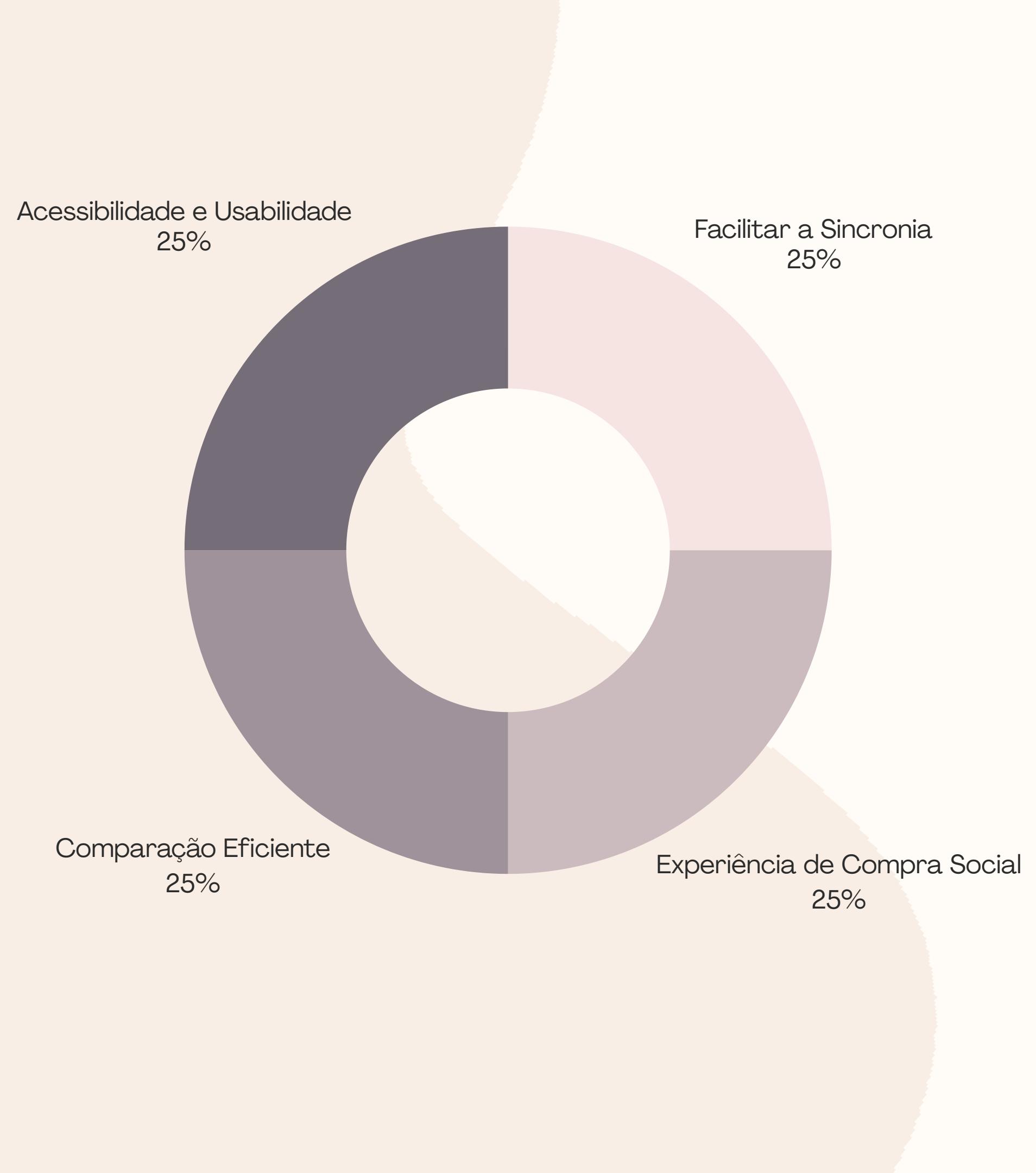
Introdução

Introdução do Projeto

Contextualização

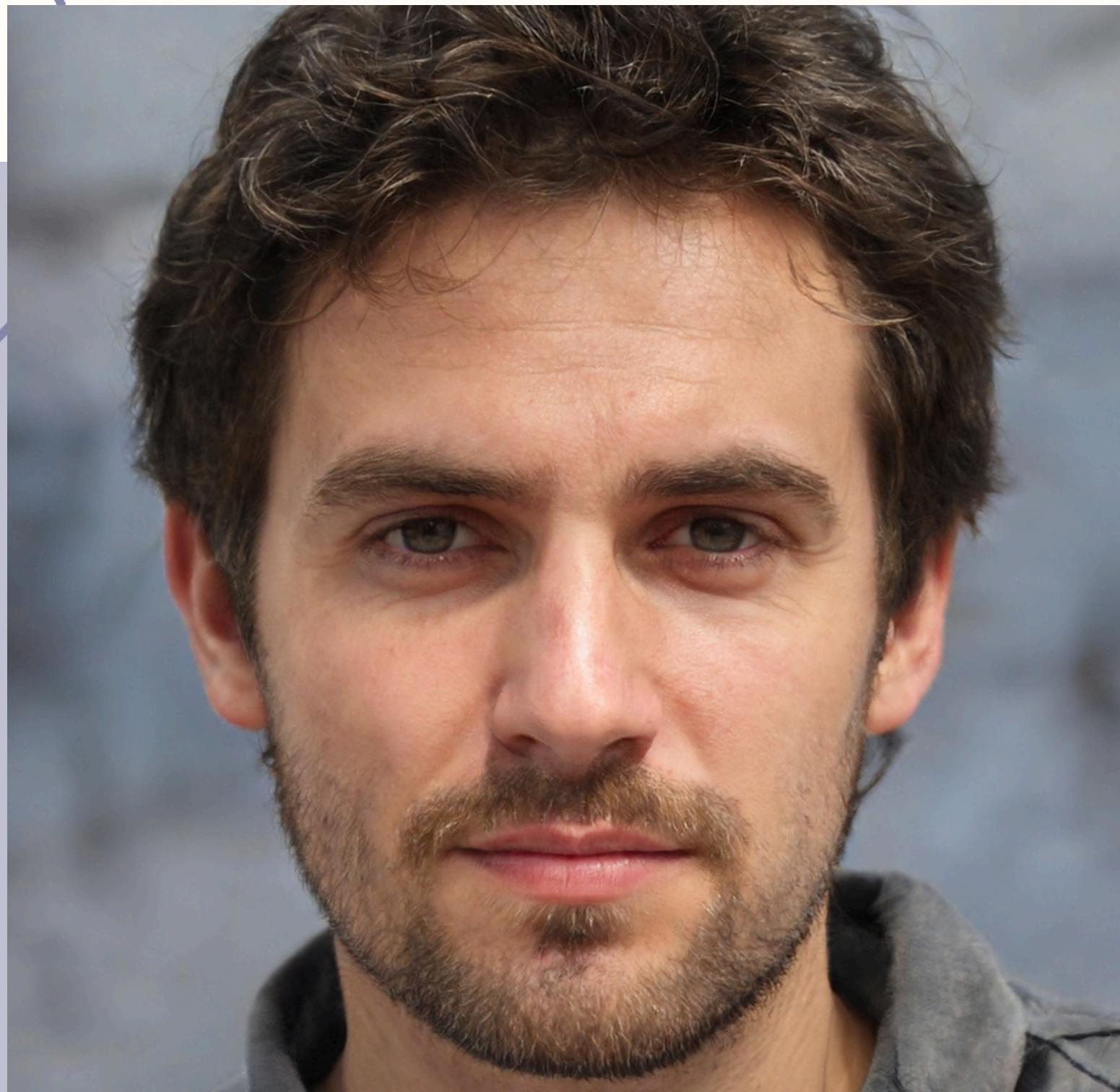
Problemas

- Falta de Sincronização
- Desorganização nas Listas
- Dificuldade de Comparação
- Experiência Fragmentada



Objetivos

Projeto do Sistema



Jovens Adultos (18 a 35 anos)

- Conhecimento tecnológico
- Relação com a tecnologia
- Comportamento
- Motivação



Famílias (35 a 55 anos)

- Conhecimento tecnológico
- Relação com a tecnologia
- Comportamento
- Motivação



Grupos de Trabalho e Amigos (20 a 45 anos)

- Conhecimento tecnológico
- Relação com a tecnologia
- Comportamento
- Motivação

Requisitos Funcionais

ID	Descrição do Requisito	Prioridade
RF-001	A aplicação deve permitir que um usuário visualize uma lista de compras mesmo sem autenticar	ALTA
RF-002	A aplicação deve permitir o cadastro e login pelo próprio usuário, por meio de um email	ALTA
RF-003	A aplicação deve permitir adição, exclusão e edição dos itens da lista se autenticado	ALTA
RF-004	A aplicação deve permitir que o usuário marque se o item foi comprado, mesmo que não esteja autenticado	ALTA
RF-005	A aplicação deve permitir que o usuário filtre os itens da lista por categoria	MÉDIO
RF-006	A aplicação deve notificar o dono da lista quando ocorrer alguma edição	BAIXA
RF-007	A aplicação deve permitir copiar a URL da lista em exibição	BAIXA

Requisitos não Funcionais

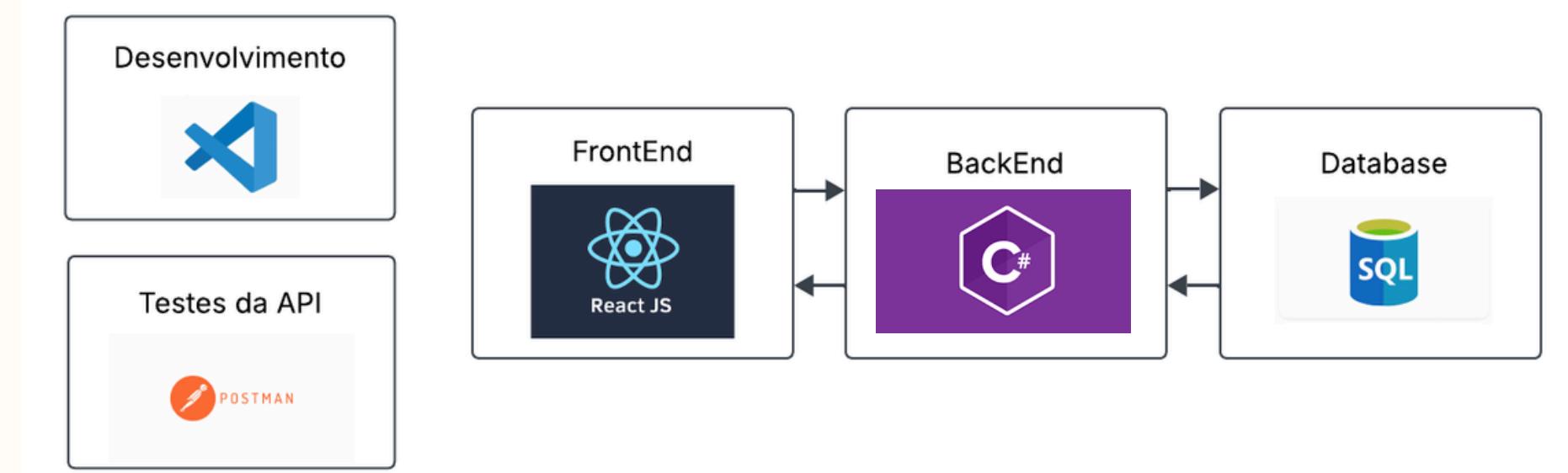
ID	Descrição do Requisito	Prioridade
RNF-001	O sistema deve suportar mais de um usuário logado simultaneamente	ALTA
RNF-002	A aplicação deve estar disponível 98% do tempo	ALTA
RNF-003	A interface deve ser intuitiva e acessível para usuários de diferentes níveis de familiaridade com tecnologia	MÉDIO

Restrições

ID	Restrição
01	O projeto deverá ser entregue até o final do semestre
02	O sistema será desenvolvido com ferramentas gratuitas ou disponibilizadas pela PUC
03	O sistema web deverá ser acessível via navegador, independentemente da plataforma

Tecnologias Utilizadas

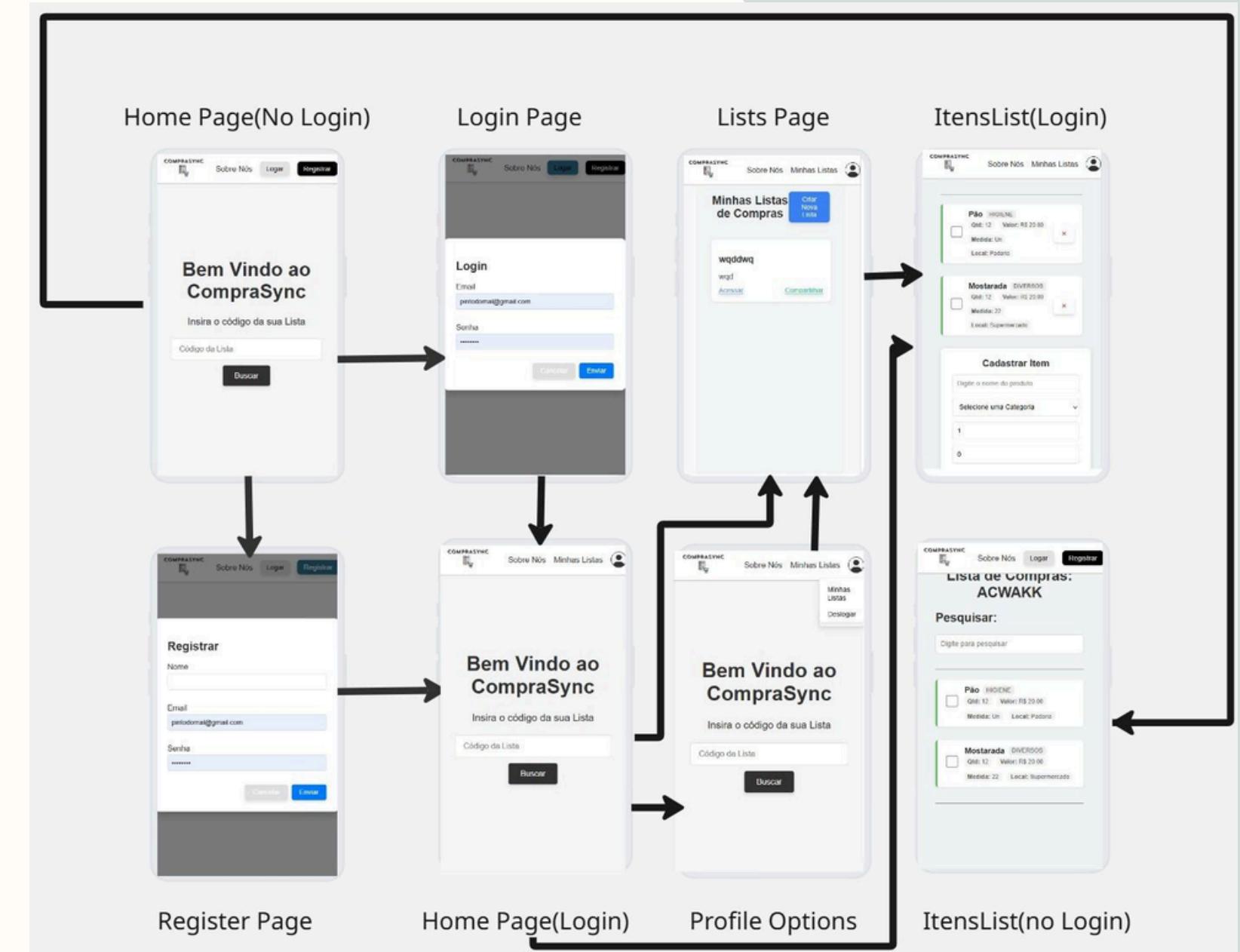
- React.js (Frontend)
- C# (Backend)
- PostgreSQL (Banco de Dados)
- Visual Studio (IDE)
- Postman
- Render (Servidor Back-End)
- Vercel (Servidor Front-End)
- SupaBase (Host do BD)





Desenvolvimento

User Flow



Casos de Teste

The screenshot shows a REST API testing interface. On the left, a sidebar lists various API endpoints under 'API Lista de Compras': POST Login, POST Criar Lista de Compras, GET Listar Listas de Compras, POST Adicionar Produto à Lista, GET Listar Produtos de uma Lista, DEL Deletar Listas de Compra, and DEL Deletar Produto de Lista. The main panel displays a POST request for 'Registrar Usuário' to the URL <https://testdeployapi-ovvd.onrender.com/auth/registro>. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2   "nome": "João Teste",  
3   "email": "bbb@bbb.com",  
4   "senha": "bbbbbb"  
5 }
```

Below the body, the 'Test Results' section shows three passed tests:

- PASSED Status 200 - Registro bem-sucedido
- PASSED Resposta é JSON
- PASSED Resposta contém token de autenticação

Casos de Teste

The screenshot shows the Postman application interface. On the left, the sidebar lists the "API Lista de Compras" collection, with the "POST Registrar Usuário" request selected. The main panel displays the "POST" request details for "https://testdeployapi-ovvd.onrender.com/auth/registro". The "Headers" tab is active, showing 10 headers. The "Test Results" tab is selected, displaying three test cases:

- FAILED** Status 200 - Registro bem-sucedido | Assertion: expected response to have status code 200 but got 400
- PASSED** Resposta é JSON
- FAILED** Resposta contém token de autenticação | Assertion: expected { message: 'Email já cadastrado' } to have property 'token'

Casos de Teste

Conclusão