Relatório Completo – Projeto PedeJá – Aplicativo de Delivery

Disciplina: Usabilidade, Desenvolvimento Web, Mobile e Jogos

Etapa: Entrega Final – Segunda Etapa (Frontend + Backend + Integração)

Alunos Integrantes do Projeto

- Marcos Guilherme Silva da Cruz - 12723213780
- Rafaela Vitória Bastos Lima Duarte - 1272227460
- Yago Ângelo da Silva Passos - 12725152333
- Sérgio Santiago Oliveira Lisbôa - 12724114613
- Jean Victor Alves de Jesus - 12725161398
- Paulo Vitor Moreira dos Santos - 12723111472
- Caio Caribé Silva - 1272417970

---

1. Informações Gerais do Projeto

- **Nome do Projeto:** PedeJá
- **Objetivo:** Desenvolver um aplicativo de delivery de comida que permita aos usuários se cadastrarem, realizarem login, visualizarem produtos disponíveis e fazerem pedidos de forma rápida e intuitiva.
- **Público-alvo:** Pessoas que buscam praticidade ao fazer pedidos de comida online, utilizando tanto dispositivos móveis quanto desktop.

---

2. Resumo da Primeira Etapa (Já Entregue em 13/05/2025)

2.1 Tecnologias Utilizadas na Etapa 1

- **Front-end inicial:** HTML5, CSS3 e JavaScript

- **Design de Interfaces:** Figma

2.2 Wireframes e Protótipos de Alta Fidelidade

- **Telas iniciais:** Login, Cadastro e HomePage
- **Imagens do Design Final:** Login, Tela Inicial, Ícone de Menu
- Link do protótipo: Food Delivery – Figma

2.3 Princípios de Usabilidade Aplicados

- Visibilidade do status do sistema
- Correspondência com o mundo real
- Controle e liberdade do usuário
- Consistência e padrões
- Prevenção de erros
- Reconhecimento em vez de memorização
- Flexibilidade e eficiência de uso
- Estética e design minimalista

---

3. Desenvolvimento da Segunda Etapa

3.1 Tecnologias Utilizadas na Etapa 2

- **Front-end:** React.js
- **Back-end:** Node.js + Express
- **Banco de Dados:** SQLite (relacional), com estruturação inicial via Sequelize ORM (somente modelagem, consultas feitas via SQL)
- **Integração:** API RESTful

3.2 Funcionalidades Completas Implementadas

Funcionalidade	Status
Cadastro de Usuário	✅ Completo
Login de Usuário	✅ Completo

CRUD de Restaurantes	✅ Completo
CRUD de Pedidos	✅ Completo
CRUD do Carrinho de Compras	✅ Completo
Categorização de Pedidos	✅ Completo
Avaliação de Restaurantes	✅ Completo
Exibição de Produtos com Fotos	✅ Completo

3.3 Integração Front-end + Back-end

- O Front-end comunica-se com o Back-end por meio de chamadas API REST.
- Endpoints para todas as operações CRUD.
- Respostas JSON padronizadas.
- Scripts de inicialização automática de banco ao startar o servidor.

3.4 Estrutura de Repositório e Entrega

Estrutura das Pastas:

```
/src       → Código-fonte Frontend e Backend
/relatório → Documentação e relatórios
/vídeo     → Apresentação em vídeo com demonstração do sistema
```

Tag GitHub: A3-Usabilidade

3.5 Scripts e Banco de Dados

- Criação automática de tabelas caso ainda não existam.
- Inserção automática de registros mínimos exigidos:
  - **3 Restaurantes**
  - **3 Modos de Pagamento**
  - **10 Produtos com fotos**

Documentação da API de Restaurante

Visão Geral

Esta API gerencia usuários, restaurantes, produtos, pedidos, avaliações, endereços, cidades, estados e carrinho de pedidos. Ela foi construída com Node.js, Express e SQLite.

Endpoints

Usuários (/usuarios)

- POST /cadastro

  Cadastra um novo usuário.

  Body: { nome, email, senha, id_grupo }

- POST /login

  Realiza login de usuário.

  Body: { email, senha }

- GET /

  Lista todos os usuários.

- GET /:id

  Busca usuário por ID.

Restaurantes (/restaurantes)

- GET /

  Lista todos os restaurantes.

- POST /create

  Cria um restaurante.

  Body: { nome, taxaFrete, ativo, aberto, foto, id_endereco }

- PUT /edit/:id

  Edita restaurante pelo ID.

- DELETE /delete/:id

Remove restaurante pelo ID.

- GET /:id

  Busca restaurante por ID.

- GET /usuario/:id_usuario

  Lista restaurantes de um usuário.

- GET /check-name/:nome

  Busca restaurante por nome.

Produtos (/produtos)

- GET /

  Lista todos os produtos.

- POST /create

  Cria um produto.

  Body: { nome, preco, descricao, id_restaurante, ativo, foto_produto, categoria }

- GET /search/:id

  Busca produto por ID.

- PUT /edit/:id

  Edita produto pelo ID.

- DELETE /delete/:id

  Remove produto pelo ID.

- GET /restaurante/:id_restaurante

  Lista produtos de um restaurante.

Pedidos (/pedidos)

- GET /

  Lista todos os pedidos.

- POST /create

  Cria um pedido.

Body: { codigo, subtotal, taxaFrete, valorTotal, dataEntrega, dataCancelamento, id_usuario, id_restaurante, id_forma_pagamento, id_status }

- GET /search/restaurante/:id_restaurante

  Lista pedidos de um restaurante.

- GET /search/codigo/:codigo

  Busca pedido por código.

- GET /search/usuario/:id_usuario

  Lista pedidos de um usuário.

- GET /:id

  Busca pedido por ID.

- PUT /edit/:id

  Edita pedido pelo ID.

- DELETE /delete/:id

  Remove pedido pelo ID.

Carrinho de Pedidos (/carrinho)

- GET /

  Lista todos os itens do carrinho.

- POST /create

  Adiciona item ao carrinho.

  Body: { id_pedido, id_produto, quantidade, precoUnitario, precoTotal, obeservacao }

- PUT /edit/:id

  Edita item do carrinho.

- DELETE /delete/:id

  Remove item do carrinho.

Avaliações (/avaliacoes)

- GET /

Lista todas as avaliações.

- GET /:id_restaurante/

  Lista avaliações de um restaurante.

- POST /:id_restaurante/create/

  Cria avaliação para restaurante.

  Body: { nome, avaliacao, comentario }

- PUT /:id_restaurante/edit/:id

  Edita avaliação.

- DELETE /:id_restaurante/delete/:id

  Remove avaliação.

Endereços (/enderecos)

- GET /

  Lista todos os endereços.

- POST /create

  Cria um endereço.

  Body: { rua, numero, bairro, cep, complemento, id_cidade }

- GET /search/:id

  Busca endereço por ID.

- PUT /edit/:id

  Edita endereço.

- DELETE /delete/:id

  Remove endereço.

Cidades (/cidades)

- GET /

  Lista todas as cidades (com sigla do estado).

- POST /create

  Cria uma cidade.

Body: { nome, id_estado }

- PUT /edit/:id

  Edita cidade.

- DELETE /delete/:id

  Remove cidade.

- GET /estado/:id_estado

  Lista cidades de um estado.

- GET /search/:id

  Busca cidade por ID.

Estados (/estados)

- GET /
  Lista todos os estados.

Observações

- Todos os endpoints retornam JSON.
- Os campos obrigatórios são validados e erros são retornados com status apropriado.
- O banco de dados é SQLite, inicializado automaticamente.
- O projeto utiliza CORS liberado para facilitar testes.

Exemplo de Requisição

POST /usuarios/cadastro

Content-Type: application/json

{
"nome": "João",
"email": "
joao@email.com",
"senha": "123456",

Para executar o sistema, siga os passos abaixo:

1. **Backend (API)**
   - Navegue até a pasta `api`.
   - Rode o comando:

```
bash
CopiarEditar
npm install
```

   - Depois, inicie o servidor de desenvolvimento com:

```
bash
CopiarEditar
npm run dev
```

2. **Frontend (React)**
   - Navegue até a pasta `frontend`.
   - Em seguida, execute:

```
bash
CopiarEditar
npm run dev
```

3.6 Responsividade e Usabilidade

- Layout responsivo adaptado a desktop e mobile.
- Continuação da aplicação das **10 heurísticas de Nielsen**.
- Melhorias de interface aplicadas desde o protótipo inicial com base nos testes e feedbacks internos.

3.7 Apresentação em Vídeo

- Duração: **3 minutos**
- Conteúdo: Navegação pelas principais telas, fluxo de pedidos, login/cadastro e integração entre camadas.

---

4. Conclusão

O projeto **PedeJá** atingiu os seguintes marcos:

✅ Desenvolvimento completo do sistema, desde o protótipo até a integração total das camadas.

✅ Entrega de código-fonte organizado e documentado.

✅ Cumprimento de todos os requisitos descritos na documentação oficial da A3.

✅ Produção de vídeo explicativo e entrega completa via repositório GitHub.