UNIVERSIDADE SALVADOR

Eduardo Batista Araujo - 1272327420
Gabriel Santana de Assunção – 12723211354
João Anísio Guimarães Nilo Dantas - 12723213518
Marcelo Silva do Carmo Filho - 1272323017
Tiago Silva Coelho Maciel - 1272326567

A3 – MVP DELIVERY FOOD

SALVADOR
2025

Eduardo Batista Araujo - 1272327420
Gabriel Santana de Assunção – 12723211354
João Anísio Guimarães Nilo Dantas - 12723213518
Marcelo Silva do Carmo Filho - 1272323017
Tiago Silva Coelho Maciel - 1272326567

A3 – MVP DELIVERY FOOD

Relatório final, apresentado a UC de Design e usabilidade para sistemas web, aplicativos e jogos, da Universidade Salvador, como parte das exigências para a obtenção de nota da A3.

Orientador(a) Prof. Lucas Silva dos Santos e Prof. Thiago Dotto Fiuza Neves

SALVADOR
2025

Sumário

Informações Gerais do Projeto

Nome do Projeto

MVP DELIVERY FOOD

Objetivo

Desenvolver uma aplicação web completa e funcional de delivery de comida. O projeto evoluiu de um protótipo inicial para um Mínimo Produto Viável (MVP) que conecta restaurantes, clientes e administradores. O sistema final oferece uma plataforma robusta para o gerenciamento de pedidos, cardápios e usuários, com uma API RESTful no backend e uma Single Page Application (SPA) reativa no frontend, aplicando os princípios de usabilidade e design centrado no usuário.

Descrição do Projeto

Este projeto é um sistema de delivery online que demonstra uma experiência de ponta a ponta para todos os envolvidos. A plataforma é dividida em um backend, responsável pela lógica de negócios e dados, e um frontend, que proporciona uma interface interativa. O sistema simula telas para cadastro, login, visualização de restaurantes e produtos, um carrinho de compras funcional e persistente, e painéis de gerenciamento para donos de restaurantes e administradores do sistema.

Tecnologias Utilizadas

O projeto foi construído com uma arquitetura full-stack, utilizando tecnologias modernas para o backend e o frontend.

Backend

- **Node.js**

Ambiente de execução para o JavaScript no servidor.

- **Express.js**

Framework utilizado para construir a API RESTful e gerenciar as rotas.

- **Sequelize**

ORM (Mapeamento Objeto-Relacional) para interação com o banco de dados de

forma abstraída e segura.

- **MySQL**

Banco de dados relacional para armazenamento dos dados da aplicação.

- **JWT (JSON Web Token)**

Para a criação de tokens de acesso que permitem a autenticação e autorização de rotas.

- **Bcrypt**

Biblioteca para a criptografia (hashing) de senhas antes de serem salvas no banco de dados.

- **Docker**

Para a conteinerização da aplicação e do banco de dados, facilitando a configuração e o deploy.

Frontend

- **React**

Biblioteca para a construção de interfaces de usuário reativas e componentizadas.

- **Vite**

Ferramenta de build e desenvolvimento que oferece um servidor de desenvolvimento rápido.

- **React Router DOM**

Para o gerenciamento de rotas na SPA, permitindo a navegação entre páginas sem recarregar o navegador.

- **Axios**

Cliente HTTP para realizar a comunicação com a API do backend.
Bootstrap & Bootstrap Icons Para estilização e componentes de UI, garantindo um design responsivo e consistente.

- **Context API**

Utilizada para o gerenciamento de estado global, como os dados de autenticação, o conteúdo do carrinho de compras e dados gerais da aplicação.

Arquitetura do Backend

O backend foi desenvolvido seguindo os princípios de uma API RESTful, com uma estrutura organizada e modular.

Estrutura de Projeto

O código-fonte do backend está organizado em diretórios que separam as responsabilidades, como models, controllers, routes e middlewares, seguindo um padrão similar ao MVC.

Modelos de Dados (Sequelize)

A interação com o banco de dados MySQL é gerenciada pelo Sequelize. Os principais modelos definidos são User, Restaurante, Produto, Cozinha, Pedido, Avaliacao, Grupo e Permissao. As associações entre eles (ex um Restaurante pertence a uma Cozinha e a um Usuário "Empresa") estão definidas no arquivo src/models/index.js, criando um esquema relacional robusto.

Rotas e Controladores

As rotas da API são definidas nos arquivos dentro de src/routes. Cada arquivo de rota (userRoutes.js, pedidoRoutes.js, etc.) agrupa os endpoints relacionados a uma entidade. Esses endpoints são direcionados para funções nos controllers, que contêm a lógica de negócio para criar, ler, atualizar ou deletar recursos.

Autenticação e Autorização (RBAC)

- **Autenticação**

O login de um usuário gera um token JWT que contém suas informações e permissões.

- **Autorização**

Foi implementado um sistema de Controle de Acesso Baseado em Papéis (RBAC). Existem Grupos (Admin, Empresa, Cliente) que possuem Permissoes associadas (ex: MANAGE_SYSTEM, MANAGE_PRODUCTS). O middleware authMiddleware valida o token JWT a cada requisição, enquanto o authorizePermission verifica se o usuário

autenticado possui as permissões necessárias para acessar um determinado endpoint.

Integração Frontend e Backend

A comunicação entre o cliente (React) e o servidor (Node.js) é o pilar da aplicação.

Comunicação via API

O frontend utiliza o cliente HTTP Axios para fazer requisições à API do backend. Foi criada uma camada de serviços (src/services), onde cada arquivo (userService.js, restauranteService.js, etc.) é responsável por agrupar as chamadas de API para uma entidade específica, tornando o código mais organizado e reutilizável.

Gerenciamento de Estado (Context API)

- **AuthContext**

Gerencia o estado de autenticação do usuário. Após o login, o token e os dados do usuário são armazenados no localStorage e no contexto, tornando-os disponíveis para toda a aplicação.

- **CartContext**

Controla o carrinho de compras. Para usuários deslogados, o carrinho é salvo no localStorage. Ao fazer login, o carrinho local é sincronizado com o carrinho do backend, garantindo persistência entre sessões e dispositivos.

- **AppDataContext**

Carrega e armazena dados globais que são usados em várias partes da aplicação, como a lista de tipos de cozinha e formas de pagamento.

- **Rotas Protegidas**

O frontend utiliza o React Router para definir as páginas da aplicação. Um componente PrivateRoute foi criado para proteger rotas que exigem autenticação e/ou permissões específicas. Ele utiliza o useHasPermission, um hook customizado, que verifica as permissões do usuário (obtidas do backend durante o login) antes de permitir o acesso à página, redirecionando o usuário caso ele não tenha autorização.

Principais Funcionalidades

A plataforma oferece um conjunto distinto de funcionalidades para cada perfil de usuário.

Para Clientes

- **Cadastro e Login**

Sistema de autenticação completo.

- **Navegação e Busca**

Filtros por tipo de cozinha, entrega grátis, restaurantes abertos e busca por nome.

- **Cardápios e Carrinho**

Visualização de produtos, adição de itens a um carrinho de compras persistente e gerenciamento de quantidades.

- **Checkout**

Gerenciamento de endereços de entrega e finalização de pedidos com simulação de diferentes formas de pagamento.

- **Histórico e Avaliação**

Acesso ao histórico de pedidos e possibilidade de avaliar pedidos já entregues.

Para Empresas (Donos de Restaurantes)

- **Dashboard**

Painel para gerenciamento do restaurante e produtos.
Gerenciamento de Restaurante Cadastro e edição de dados como nome, CNPJ, endereço e taxa de frete.

- **Gerenciamento de Cardápio**

Criação, edição e ativação/desativação de produtos.

- **Gerenciamento de Pedidos**

Recebimento e atualização do status dos pedidos (pendente, em preparo, entregue, etc.).

Para Administradores

- **Dashboard Administrativo**

Visão geral com estatísticas de usuários, restaurantes e pedidos.

- **Gerenciamento Total**

Permissão para gerenciar todos os usuários (incluindo atribuição de papéis), restaurantes, tipos de cozinha, formas de pagamento e avaliações do sistema.

Perfis de Usuário e Acesso

O sistema é populado com dados iniciais para facilitar os testes, com perfis pré-configurados. A senha padrão para todos é 123.

- **Administrador**

Email: admin@delivery.com

Permissões Acesso total a todas as funcionalidades de gerenciamento do sistema.

- **Cliente**

Email: cliente@delivery.com

Permissões Realizar e visualizar pedidos, avaliar, gerenciar carrinho e endereços.

- **Empresas (Donos de Restaurantes)**

Email: dono.pizza@example.com (Dono da Pizzaria da Praça)

Email: chef.sushi@example.com (Dono do Sushi Master)

Email: gerente.burger@example.com (Dono do Burger Place)

Permissões Gerenciar o próprio restaurante, cardápio e pedidos recebidos.

Princípios de Usabilidade Aplicados

Os princípios de usabilidade de Nielsen foram aplicados desde a fase de prototipagem e mantidos na versão final da aplicação.

Visibilidade do status do sistema

O sistema fornece feedback constante. O carrinho atualiza em tempo real, e o usuário recebe confirmações visuais ao finalizar um pedido ou ao interagir com o sistema.

Correspondência com o mundo real

A linguagem utilizada é simples e direta, e os formulários possuem rótulos claros.

Controle e liberdade do usuário

O usuário pode facilmente remover itens do carrinho ou cancelar ações antes da finalização de um pedido.

Consistência e padrões

Uma identidade visual consistente foi mantida em todo o sistema, com a mesma paleta de cores, estrutura de botões e títulos.

Prevenção de erros

Campos obrigatórios são validados, e ações críticas (como remover um item) solicitam confirmação.

Reconhecimento em vez de memorização

Menus de navegação são consistentes e as informações importantes estão sempre agrupadas e acessíveis, como no menu de perfil.

Flexibilidade e eficiência

O carrinho pode ser acessado por um ícone flutuante, e a interface se adapta a diferentes tamanhos de tela.

Estética e design minimalista

O design é limpo e focado nas funcionalidades essenciais, sem sobrecarga visual.

Recuperação de erros

O sistema fornece mensagens de erro claras que ajudam o usuário a diagnosticar e corrigir problemas.

Ajuda e documentação

Todas as ações possuem feedback direto e as opções do menu são claramente identificadas.

Responsividade

A interface foi construída com foco em dispositivos móveis, com layout adaptável e botões otimizados para toque.

Considerações Finais

O projeto evoluiu com sucesso de um protótipo de alta fidelidade para um Mínimo Produto Viável (MVP) completo. A integração de um backend robusto com Node.js, Express e MySQL ao frontend reativo com React cumpriu todos os objetivos propostos inicialmente, como autenticação de usuários, painéis de administração e o salvamento de dados reais em um banco de dados

O resultado é uma aplicação de delivery funcional, segura e escalável, que demonstra uma experiência de usuário fluida e acessível, respeitando as boas práticas de usabilidade e design responsivo. O sistema serve como uma base sólida e completa para futuras expansões e implementações de novas funcionalidades.