Universidade Salvador Unifacs

Alunos

Alice Martins Bahiense Bezerra Bauler
RA: 12724132022

Catarina dos Santos Romeiro
RA: 12724131744

Eduardo Copque da Silva
RA: 12725149734

Roan Nascimento Lisboa
RA: 12725138629

Sistemas Distribuídos e Mobile

Projeto de criação de uma API funcional para reservas de um restaurante.

Professor Orientador: Eduardo Sidney Da Silva Xavier

Salvador - Bahia
2025.1

Sumário

1. APRESENTAÇÃO DO PROJETO

Criar um sistema de reserva de restaurante com a finalidade de atender as necessidades de três perfis de usuários, sendo eles;

- **Atendentes**: Responsáveis por criar e cancelar reservas
- **Garçons**: Responsáveis por confirmar reservas e gerenciar o status das mesas
- **Gerentes**: Responsáveis por monitorar o desempenho através de relatórios analiticos

1.1 OBJETIVOS DO PROJETO

O projeto teve como objetivo criar um sistema que automatize os processos das reservas, fornecer uma visão em tempo real da ocupação do restaurante, gerar relatórios gerenciais para tomadas de decisões e reduzir conflitos da alocação de mesas.

2. APRESENTAÇÃO DE SOLUÇÃO

Um sistema distribuído com três camadas principais:

- **Frontend:**

Camada com a responsabilidade de trazer um interface responsiva para os três perfis de usuários, sendo eles: atendentes, garçons e gerentes.

Desenvolvido em HTML, CSS e JavaScript.

- **Backend:**

Camada responsável pela integração com o banco de dados e inicialização do servidor do projeto.

Desenvolvido com a API REST em Node.js, programado em TypeScript.

- **Banco de Dados:**

Camada responsável pela criação do banco de dados da aplicação, tendo como modelo relacional com tabelas referentes nesse projeto as tabelas: mesas, garçons e reservas.

Banco de dados escolhido PostgreSQL

As funcionalidades desta aplicação garantem criação e cancelamento de reservas, confirmação de ocupação de mesas, relatórios por período, mesa e garçons e liberação automática de mesas após uso.

3. JUSTIFICATIVA DAS ESCOLHAS TÉCNICAS

Front-End com JavaScript:

Optou-se por não utilizar frameworks no desenvolvimento do frontend devido ao escopo **restrito** e bem **delimitado** da aplicação. Essa abordagem permite um carregamento mais rápido, eliminando **sobrecargas** desnecessárias introduzidas por bibliotecas **robustas**. Além disso, o código permanece **enxuto** e de fácil **manutenção**, o que facilita intervenções rápidas e ajustes pontuais sem a complexidade adicional de dependências externas.

Back-End com Node.js e TypeScript:

O **Node.js** foi escolhido por sua alta performance em operações de **entrada e saída (I/O)**, característica essencial para aplicações que demandam escalabilidade e baixa latência. O **TypeScript** complementa essa escolha ao oferecer tipagem **estática**, o que contribui significativamente para a segurança do código e facilita a manutenção a **longo prazo**. Já o **Express** foi adotado por ser um framework **minimalista** e **leve**, ideal para a construção de **APIs RESTful** de forma **simples, eficiente** e com boa organização **estrutural**.

Banco de Dados PostgreSQL

Optou-se por um banco de dados relacional devido à necessidade de garantir integridade e consistência em operações críticas, asseguradas pelo suporte total às propriedades **ACID**. O modelo relacional se mostrou adequado para representar os **relacionamentos complexos** do **domínio** da aplicação. Além disso, a escolha oferece **flexibilidade** para futuras **extensões**, com suporte nativo ao **armazenamento** e manipulação de dados **semi estruturados** por meio do tipo **JSONB**.

Docker:

O **Docker** foi adotado para garantir o **isolamento** de cada serviço em **containers independentes**, promovendo **maior segurança** e **controle** sobre o **ambiente de execução**. Essa abordagem facilita a **portabilidade** da aplicação entre diferentes ambientes, simplificando a **implantação** e reduzindo problemas de **configuração**. Além disso, o uso do mesmo ambiente tanto em **desenvolvimento** quanto em **produção** assegura **consistência** no comportamento da aplicação, minimizando **divergências** e facilitando a **detecção de erros**.

4. ASPECTOS TÉCNICOS DETALHADOS

Stack Tecnológica Completa

Back-End:

- **Linguagem:** TypeScript 5.8
- **Runtime:** Node.js 20
- **Framework:** Express 5
- **Bibliotecas:**
    - **pg:** Cliente PostgreSQL
    - **dotenv:** Gerenciamento de variáveis de ambiente
    - **cors:** Middleware para Cross-Origin Resource Sharing

Front-End:

- **Linguagens:** HTML, CSS, JavaScript
- **Bibliotecas:**
    - CSS puro
    - JavaScript modular sem frameworks

Banco de Dados:

- **SGBD**: PostgreSQL 15
- **Extensões**: Utilização de triggers para histórico
- **Índices**: Otimizados para consultas frequentes

Infraestrutura:

- **Conteinerização**: Docker com Docker Compose
- **Orquestração**: Compose para multi-container
- **Servidor Web**: Nginx para frontend

Ambiente de Desenvolvimento

- **Docker Desktop**: Para containers locais
- **VS Code**: IDE com extensões para TypeScript/Docker
- **Postman**: Testes de API durante desenvolvimento
- **pgAdmin**: Interface gráfica para o PostgreSQL

Fluxo de trabalho:

- **Desenvolvimento:**

Durante o desenvolvimento utilizamos contêineres isolados para cada serviço.

- **Testes:**

Testes manuais em cada perfil de usuário e validação de cenários de erro.

- **Implementação:**

Na implementação foram utilizadas Builds de imagens docker otimizadas, configuradas via variáveis de ambiente.

5. EXPLICAÇÃO DOS ARQUIVOS DA PASTA RAIZ

pasta raiz: A pasta **principal** da aplicação armazena os arquivos essenciais para seu funcionamento. Entre eles estão o **.env**, que guarda as **variáveis de ambiente** necessárias para a configuração do sistema, o **.gitignore**, responsável por indicar ao **Git** quais arquivos devem ser ignorados no controle de versão, e o **docker-compose.yml**, que define a **configuração** e **orquestração** dos containers Docker utilizados na aplicação.

backend: A pasta principal do **backend** contém os arquivos essenciais do **servidor**, como **rotas**, **controllers** e **configurações**. O **tsconfig.json** define as regras do TypeScript, enquanto o **Dockerfile** permite a criação de uma imagem Docker da aplicação. Os arquivos **package.json** e **package-lock.json** gerenciam e versionam as dependências. Dentro da pasta **src**, o **index.ts** inicializa o **servidor**. A subpasta **controllers** armazena a lógica das **requisições** da **API** e interações com o **banco**. Em **database**, o **db.ts** realiza a conexão com o PostgreSQL e emite erros em caso de falha. Já a pasta **routes** organiza as rotas da aplicação, sendo utilizadas nas **requisições** do frontend. Essa estrutura garante organização, manutenibilidade e escalabilidade ao projeto.

database: pasta responsável por definir o banco de dados que será inicializado pelo **Docker Compose** durante a construção do ambiente. Seu destaque é a subpasta **docker-entrypoint-initdb.d**, que contém o arquivo **01-init.sql**. Esse script é executado automaticamente na criação do container e contém toda a estrutura inicial do banco de dados da aplicação, incluindo tabelas, relações e dados essenciais.

frontend: A pasta principal do **frontend** reúne os arquivos essenciais da **interface** da aplicação. O **Dockerfile** é utilizado para gerar a imagem **Docker**, enquanto o **nginx.conf** configura o servidor **NGINX** que será responsável por servir a aplicação após a **compilação**. O **index.html** funciona como a **tela principal,** oferecendo acesso às demais **seções**. Os arquivos **style.css** e **home-style.css** definem a **estilização global** e específica do menu inicial, respectivamente. A sub pasta **assets** armazena **imagens** e **recursos visuais** da aplicação. Além disso, há três pastas dedicadas aos diferentes perfis de usuário (**garçom**, **caixa** e **gerente**), cada uma contendo seu próprio **index.html, arquivos de estilo** e um **index.js** responsável por realizar **requisições** ao **backend** e **tratar** os **dados** recebidos.

6. LINKS ÚTEIS

Github: https://github.com/RoanNL/Projeto-A3-UC-Sistemas-Distribuidos

Link do vídeo de apresentação: https://youtu.be/gZ9wGOkPEmg