Manual Técnico e Guia de Instalação

N-Log-de-Fome — Sistema de Gestão de Pedidos de Delivery

Levi de Pontes e Savas Constantin

27 de novembro de 2025

Sumário

1 Introdução

O **N-Log-de-Fome** é uma solução tecnológica Fullstack desenvolvida para modernizar e gerenciar o fluxo de pedidos em estabelecimentos alimentícios. O sistema integra uma interface web responsiva com um servidor robusto de processamento de dados.

A arquitetura do projeto foi migrada para utilizar **Node.js (TypeScript)** no backend com o driver nativo **mysql2**, implementando o padrão de projeto **Repository (DAO)** para execução de consultas SQL puras e otimizadas. No frontend, utiliza-se **React (Vite)** para uma experiência de usuário fluida.

2 Pré-requisitos do Ambiente

Para garantir a execução correta do projeto, o ambiente de desenvolvimento deve atender aos seguintes requisitos:

- **Sistema Operacional:** Windows 10/11, Linux (Ubuntu/Debian) ou macOS;
- **Runtime:** Node.js (Versão LTS 18.x ou superior);
- **Gerenciador de Pacotes:** NPM (Instalado nativamente com o Node);
- **Banco de Dados:** MySQL Server 8.0+ e MySQL Workbench (para execução de scripts e visualização);
- **Controle de Versão:** Git;
- **IDE:** Visual Studio Code (VSCode) atualizado;

3 Instalação e Configuração Inicial

Acesse o repositório oficial em: https://github.com/savass33/N-log-de-Fome

3.1 Clonagem do Repositório

Ação: Obter o Código Fonte

Passo 1: Abra o terminal ou Git Bash na pasta onde deseja salvar o projeto.

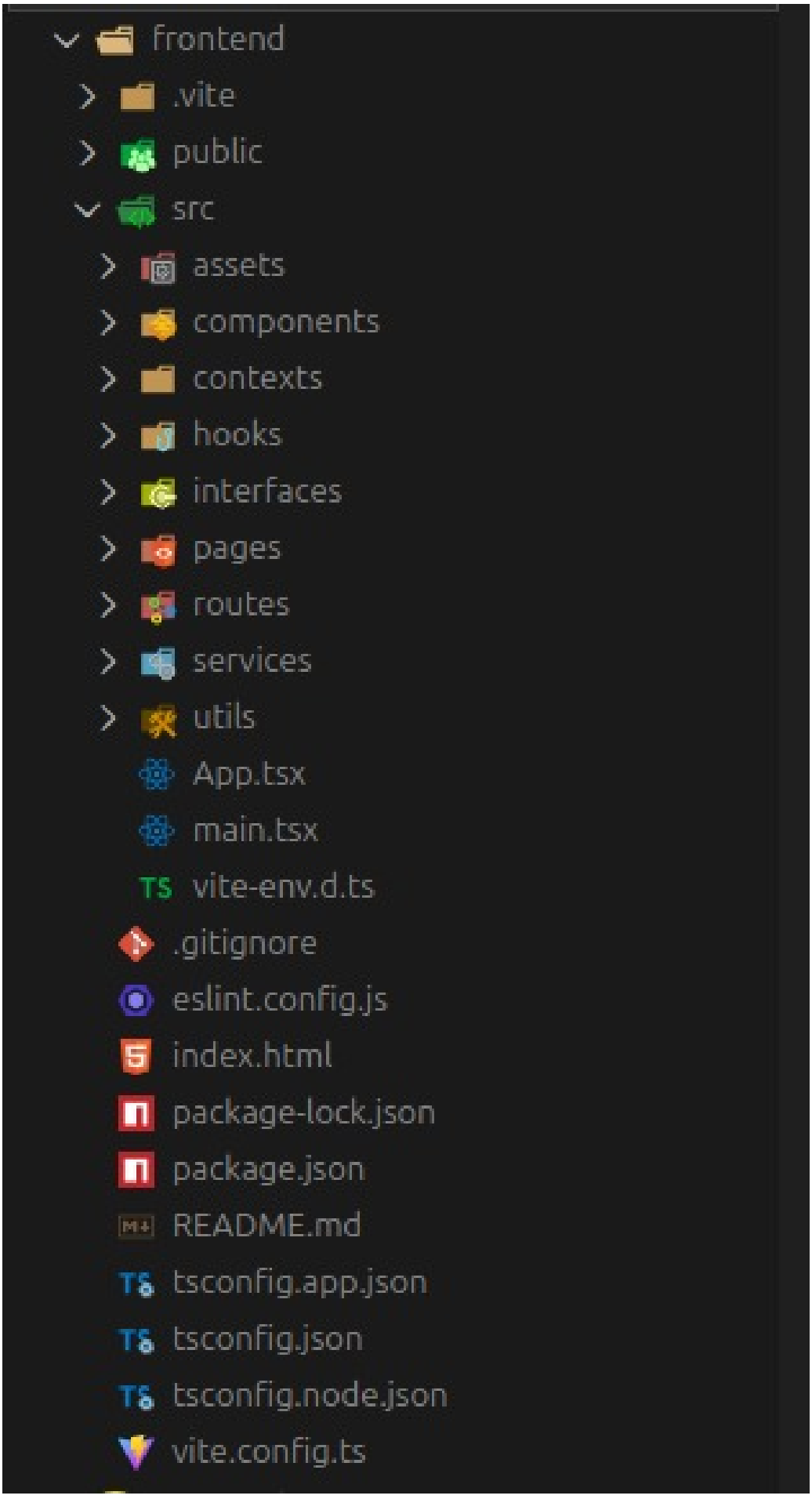
Passo 2: Clone o repositório oficial:

```
git clone https://github.com/savass33/N-log-de-Fome.git
```

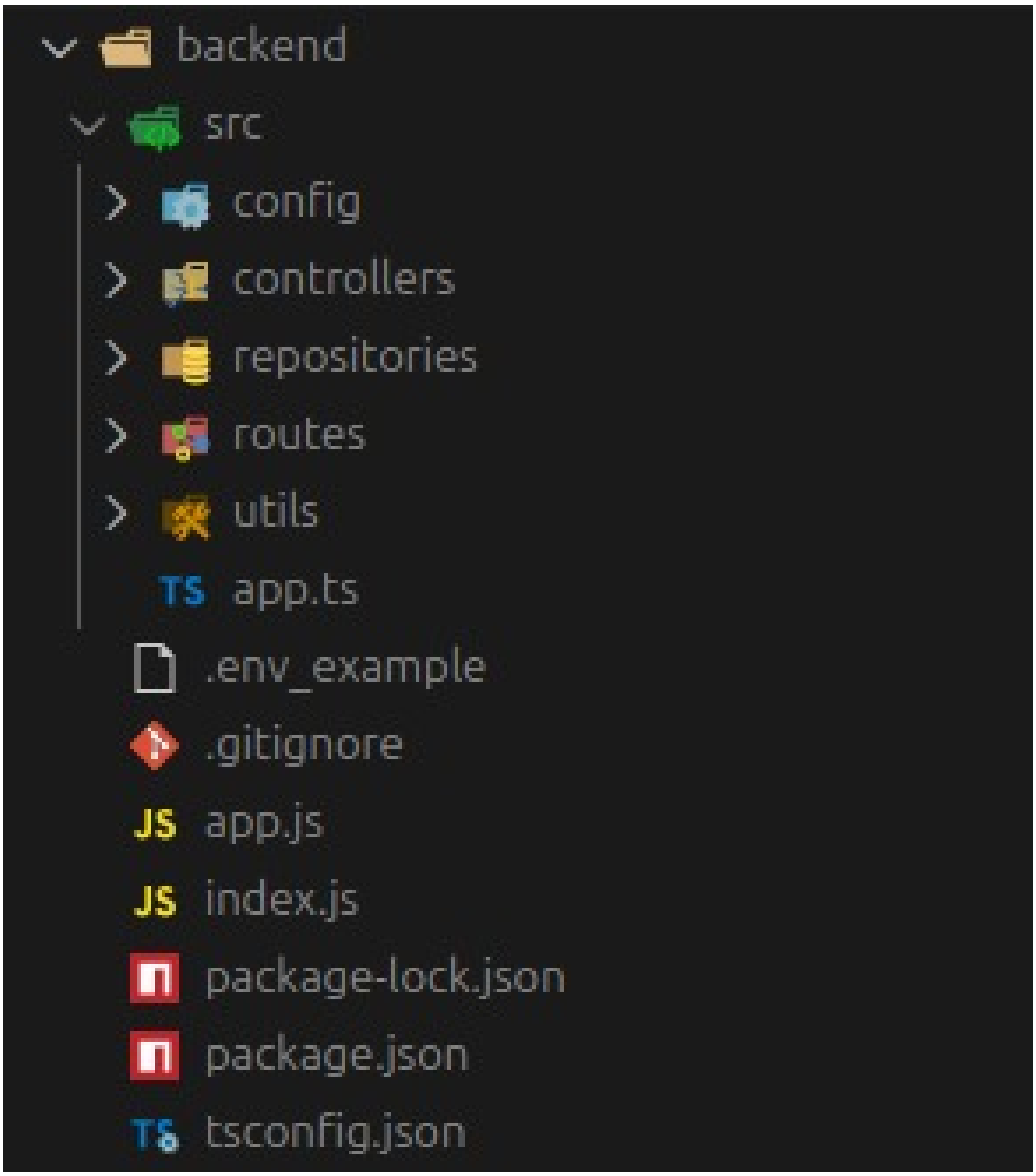
Passo 3: Acesse o diretório raiz do projeto:

```
cd N-log-de-Fome
```

Passo 4: Verifique a existência das pastas `backend` e `frontend`.


frontend
.vite
public
src
assets
components
contexts
hooks
interfaces
pages
routes
services
utils
App.tsx
main.tsx
vite-env.d.ts
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts

Estrutura inicial do repositório clonado (Frontend)



Estrutura inicial do repositório clonado (Backend)

4 Estrutura Arquitetural do Projeto

O projeto adota uma arquitetura separada (Client-Server) em Monorepo, utilizando o padrão MVC no backend com Repositórios para acesso a dados.

4.1 Frontend (Camada de Apresentação)

Localizado na pasta `frontend/`, construído com React e Vite.

- **src/pages/** → Telas principais (Login, Dashboard, Cardápio).
- **src/components/** → Componentes reutilizáveis (Botões, Inputs, Cards).
- **src/services/** → Camada de abstração HTTP (Axios) para comunicação com a API.

- **src/hooks/** → Hooks personalizados para autenticação e estado global.

4.2 Backend (Camada de Dados e Regras)

Localizado na pasta `backend/`, utilizando Node.js, Express e SQL Puro.

- **src/app.ts** → Ponto de entrada da aplicação, configuração de middlewares e rotas.
- **src/config/db.ts** → Configuração do Pool de Conexões MySQL.
- **src/controllers/** → Controladores que gerenciam a lógica de requisição/resposta.
- **src/repositories/** → Camada de Acesso a Dados (DAO). Contém as classes que executam as queries SQL brutas (`SELECT`, `INSERT`, etc.).
- **src/routes/** → Definição dos endpoints da API.
- **.env** → Variáveis de ambiente para conexão segura com o banco.

5 Guia de Execução Passo a Passo

A execução do sistema requer uma ordem específica para garantir que as dependências do banco de dados estejam satisfeitas antes do backend iniciar.

5.1 1. Inicialização do Frontend

Ação: Configurar Interface do Usuário

Passo 1: Abra um terminal na raiz do projeto.

Passo 2: Navegue até o diretório do frontend:

```
cd frontend
```

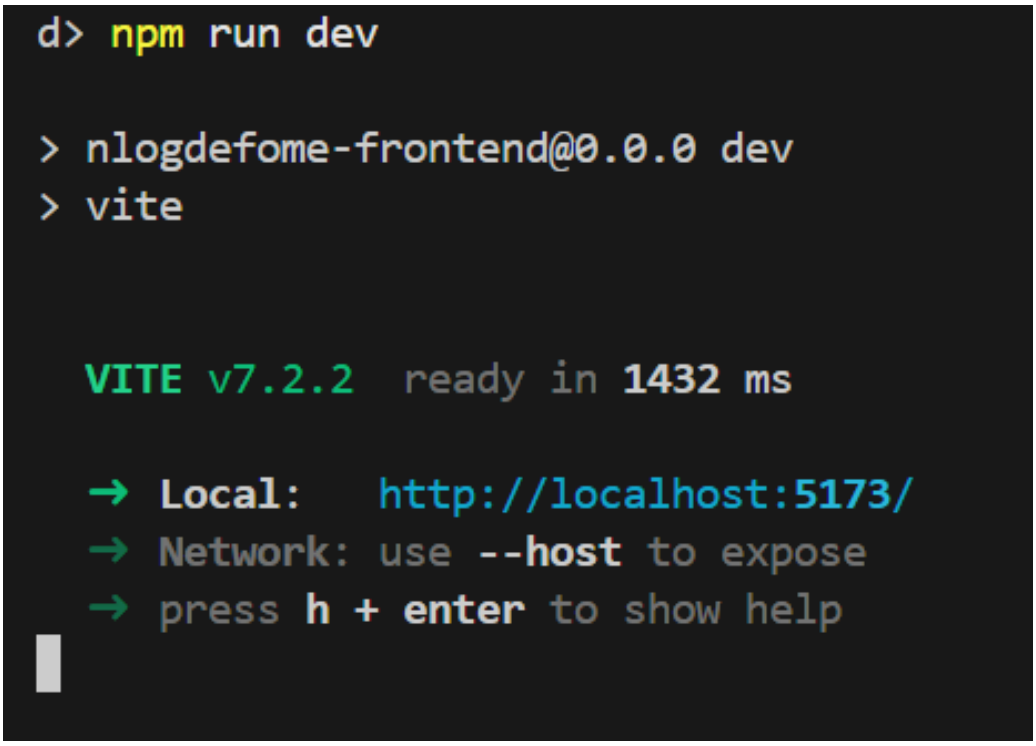
Passo 3: Instale as dependências do projeto:

```
npm install
```

Passo 4: Execute o servidor de desenvolvimento Vite:

```
npm run dev
```

Passo 5: O terminal indicará o endereço local (ex: `http://localhost:5173`). Mantenha este terminal aberto.



Frontend rodando no terminal

5.2 2. Configuração do Banco de Dados (MySQL)

Ação: Database First - Criação e População

Atenção

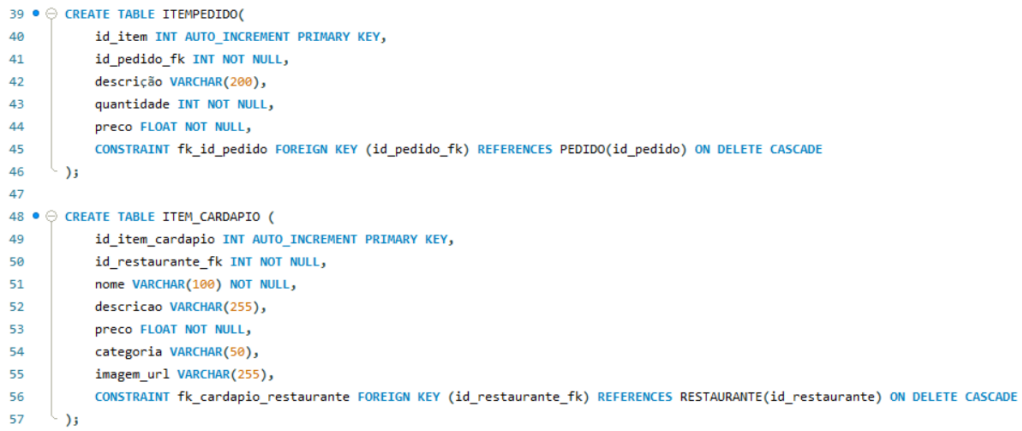
Como o sistema utiliza SQL puro, é essencial criar o banco e as tabelas manualmente antes de iniciar a API.

Passo 1: Abra o **MySQL Workbench**.

Passo 2: Abra o arquivo de script SQL localizado na raiz do repositório: `script.sql`.

Passo 3: Execute o script completo para criar o banco `n_log_de_fome` e suas tabelas.

Passo 4: (Opcional) Insira dados iniciais para teste.



```sql
CREATE TABLE ITEMPEDIDO(
    id_item INT AUTO_INCREMENT PRIMARY KEY,
    id_pedido_fk INT NOT NULL,
    descrição VARCHAR(200),
    quantidade INT NOT NULL,
    preco FLOAT NOT NULL,
    CONSTRAINT fk_id_pedido FOREIGN KEY (id_pedido_fk) REFERENCES PEDIDO(id_pedido) ON DELETE CASCADE
);

CREATE TABLE ITEM_CARDAPIO (
    id_item_cardapio INT AUTO_INCREMENT PRIMARY KEY,
    id_restaurante_fk INT NOT NULL,
    nome VARCHAR(100) NOT NULL,
    descricao VARCHAR(255),
    preco FLOAT NOT NULL,
    categoria VARCHAR(50),
    imagem_url VARCHAR(255),
    CONSTRAINT fk_cardapio_restaurante FOREIGN KEY (id_restaurante_fk) REFERENCES RESTAURANTE(id_restaurante) ON DELETE CASCADE
);
```

Script SQL executado no Workbench

5.3 3. Inicialização do Backend

Ação: Configuração de Ambiente e Start da API

Passo 1: Abra um **segundo terminal** (novo processo).

Passo 2: Navegue até o diretório do backend:

```
cd backend
```

Passo 3: Instale as dependências:

```
npm install
```

Passo 4: Crie um arquivo `.env` na pasta `backend/` baseado no `.env_example`:

```
PORT=3001
DB_HOST=localhost
DB_USER=root
DB_PASS=sua_senha_aqui
DB_NAME=n_log_de_fome
```

Passo 5: Inicie o servidor:

```
npm run dev
```

Passo 6: Mensagem esperada: `Servidor Backend rodando na porta 3001` e `Conectado ao MySQL com Sucesso!`.

Backend rodando e conectado ao MySQL

6 Documentação da API e Endpoints

O backend expõe uma API RESTful organizada por recursos.

6.1 Autenticação

- **POST** `/api/auth/login` → Login unificado (Cliente, Restaurante ou Admin).

6.2 Recursos Principais

Endpoints de CRUD

- **Clientes:** `GET/POST/PUT/DELETE /api/clientes`
- **Restaurantes:** `GET/POST/PUT/DELETE /api/restaurantes`
- **Cardápio:** `GET/POST/PUT/DELETE /api/cardapio`
- **Pedidos:**
  - **GET** `/api/pedidos` → Lista geral (Admin).
  - **POST** `/api/pedidos` → Criação de pedido com itens (Transação atômica).
  - **PUT** `/api/pedidos/:id` → Atualização de status.

7 Modelagem de Dados (Schema)

O banco de dados relacional é composto pelas seguintes entidades principais:

Cliente: Armazena dados pessoais e histórico de endereços.

Restaurante: Dados do estabelecimento, tipo de cozinha e detalhes de contato.

Admin: Usuários com privilégios de gestão do sistema.

Pedido: Entidade central que conecta Cliente e Restaurante. Possui status (Pendente, Preparando, etc.).

ItemPedido: Tabela pivô que armazena os produtos e quantidades de cada pedido.

ItemCardapio: Produtos oferecidos por cada restaurante.

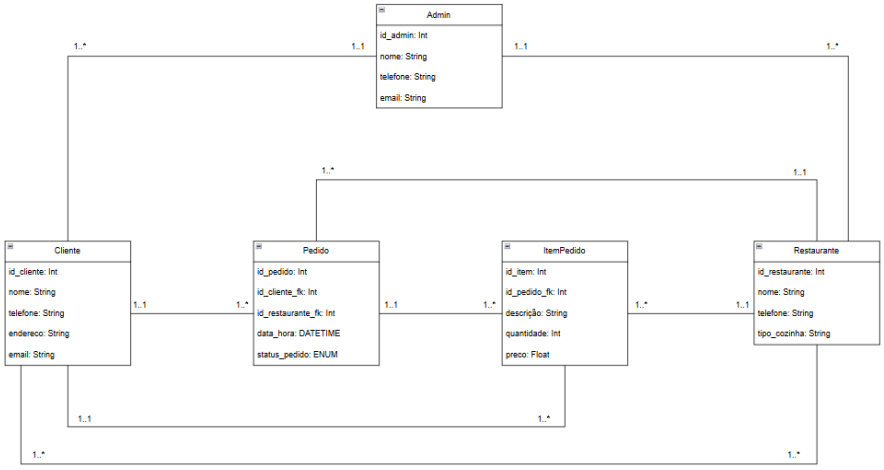


Diagrama Relacional

8 Resolução de Problemas Comuns

- **Erro "Access denied for user":**
  - Verifique se o arquivo `.env` na pasta `backend` contém a senha correta do seu usuário root do MySQL.
- **Erro "Unknown database":**
  - Certifique-se de que rodou o `script.sql` antes de iniciar o backend. O banco `n_log_de_fome` deve existir.
- **Erro de Conexão (ECONNREFUSED):**
  - Verifique se o serviço MySQL está rodando e se a porta 3306 está liberada.
- **Erro de CORS no Frontend:**
  - Verifique se o backend está rodando na porta 3001.
  - O frontend está configurado para apontar para `http://localhost:3001/api`.

9 Conclusão

Este documento detalhou o processo de instalação e arquitetura do projeto **N-Log-de-Fome**. A migração para SQL Puro com arquitetura de Repositórios permitiu um controle granular sobre as consultas ao banco de dados, resultando em um sistema educativo robusto e performático para a gestão de delivery.