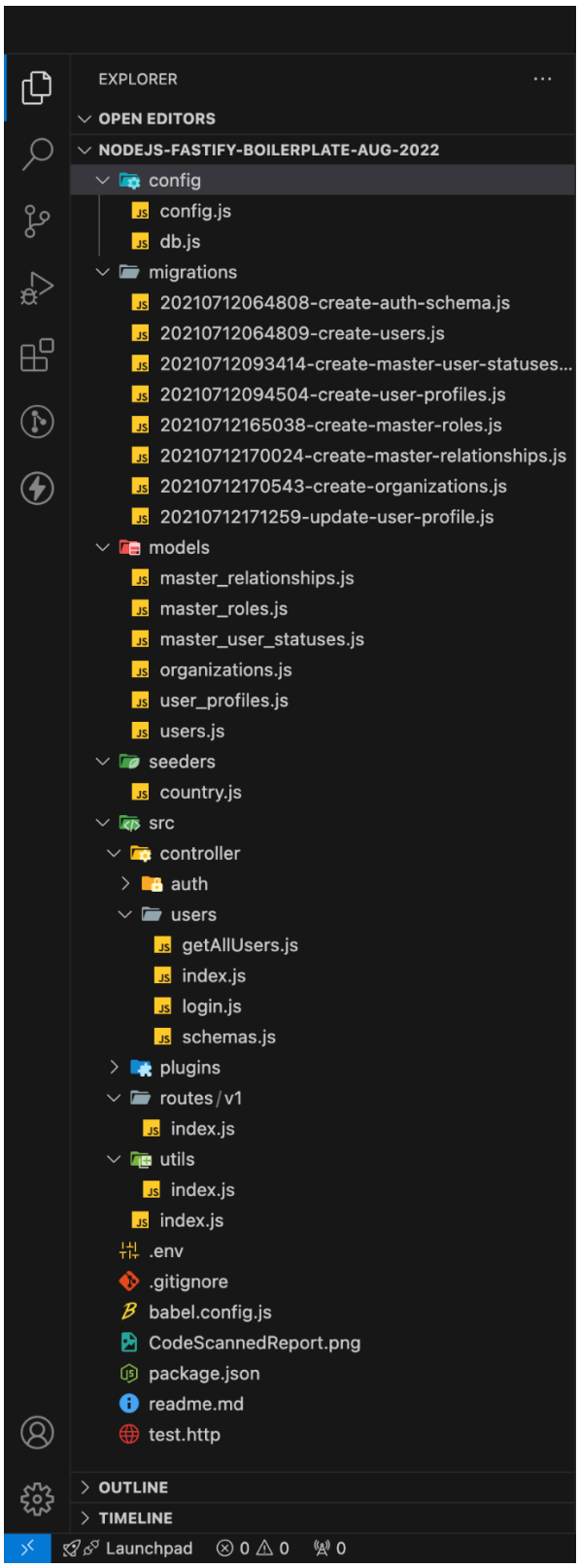
BACKEND BOILERPLATE

A backend boilerplate refers to a ready-to-use, predefined codebase or framework that provides the foundational structure and components needed to kickstart backend development for a project. It includes the essential tools, configurations, and conventions required to build and manage the server-side logic of an application.

Sample Backend Boilerplate :



Explanation of the Boilerplate :

Config Folder

- config.js: Holds application configuration settings
- db.js: Database connection configuration

Migrations Folder (Database structure changes)

- Contains dated files showing database evolution:
- Auth schema creation
- User tables creation
- User profiles setup
- Master roles and relationships setup
- Organizations setup
- Each file is timestamped (like 202107120648...) for tracking order

Models Folder (Database table structures)

- master_relationships.js: Defines relationships between users/roles
- master_roles.js: User role definitions
- master_user_statuses.js: User status types
- organizations.js: Organization structure
- user_profiles.js: User profile information
- users.js: Basic user information

Seeders Folder

- country.js: Initial country data for the database

src Folder (Main application code)

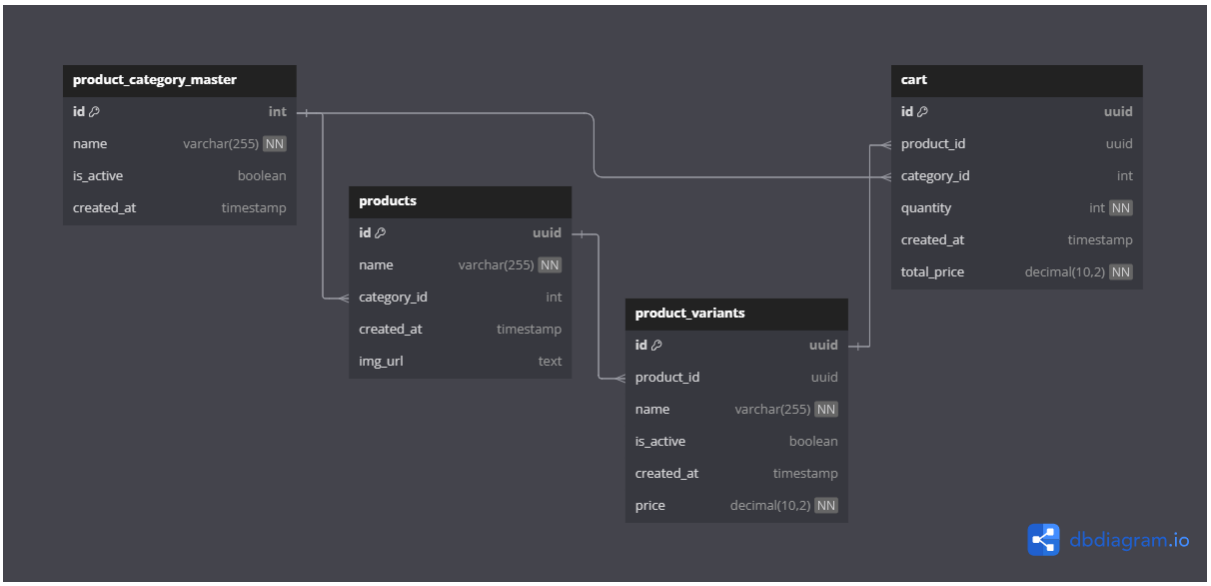
- **controller/**
    - auth/: Authentication logic
    - users/: User management functions
        - getAllUsers.js: Get users list
        - login.js: Login handling
        - schemas.js: Data validation rules
- **plugins/**: Fastify plugin configurations
- **routes/v1**: API endpoints (version 1)

    - index.js: Main route definitions
- **utils/**: Helper functions

Root Files

- .env: Environment variables
- .gitignore: Git ignore rules
- babel.config.js: JavaScript compiler settings
- package.json: Project dependencies
- readme.md: Project documentation
- test.http: API test endpoints

DB DIAGRAM



DB Diagram Structure & Format:

1. **Table Representation**
    - Dark rectangular boxes represent tables
    - Table name is shown in the header (e.g., "product_category_master")
    - Each row below represents a column
2. **Column Format**
    - Column name on the left
    - Data type on the right
    - Special notations:
        - NN means "NOT NULL"

        - Primary keys marked with a key symbol (🔑)
        - UUIDs vs integers for IDs
3. **Data Types Shown**
    - varchar(255): Text with max length
    - boolean: True/False values
    - timestamp: Date/time values
    - int: Whole numbers
    - decimal(10,2): Numbers with decimals
    - uuid: Unique identifiers
    - text: Long text content
4. **Relationships**
    - Lines connecting tables show relationships
    - Direction of arrows indicates relationship type
    - Shows foreign key connections between tables

Common Conventions in the Diagram:

1. **Naming**
    - Tables: snake_case (lowercase with underscores)
    - Clear, descriptive names
    - Suffix '_master' for lookup tables
2. **Standard Fields**
    - id: For primary keys
    - created_at: Timestamp for record creation
    - is_active: For soft delete functionality
3. **Common Patterns**
    - UUID vs Integer IDs
    - Timestamp tracking
    - Status flags
    - Foreign key relationships