Database Models & Schema

Relevant source files

This document covers the database architecture and Sequelize model implementations for the FoodExpress food delivery platform. It details the entity relationships, model definitions, and schema structure that supports multi-role user management, restaurant operations, and order processing.

For information about the authentication and middleware components that work with these models, see 4.2. For details about the overall architecture and dependencies, see 4.1.

Overview

The FoodExpress platform uses a MySQL database with Sequelize ORM to manage a complex multi-entity schema supporting clients, restaurants, and platform administrators. The schema implements a comprehensive food delivery workflow from user registration through order fulfillment.

Sources: Proy3_Pedidos/def proyecto 1-570

Database Technology Stack

The platform employs the following data persistence technologies:

Component	Technology	Purpose
Database Engine	MySQL	Primary data storage
ORM	Sequelize	Object-relational mapping
Model Configuration	underscored naming	Database field conventions
Timestamps	Automatic	Created/updated tracking
Associations	Sequelize relationships	Foreign key management

Sources: Proy3_Pedidos/models/address.js 1-68 Proy3_Pedidos/def proyecto 146-147

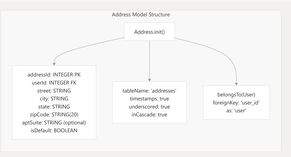
Core Entity Models

User Model Structure

Sources: Proy3_Pedidos/def proyecto 245-249 Proy3_Pedidos/def proyecto 321-324

Address Model Implementation

The `Address` model demonstrates the Sequelize implementation pattern used throughout the platform:



Sources: Proy3_Pedidos/models/address.js 19-67

Entity Relationship Schema

Sources: Proy3_Pedidos/def proyecto 280-280

Model Association Patterns

Address-User Relationship Implementation

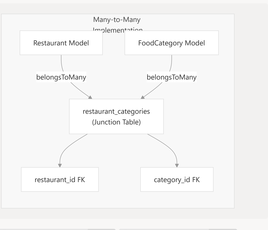
The `Address` model demonstrates the standard association pattern:

Relationship Type	Implementation	Configuration
Foreign Key	`userId: DataTypes.INTEGER`	References `users.user_id`
Association Method	`belongsTo(models.User)`	Alias `'user'`
Cascade Behavior	`inCascade: true`	Automatic cleanup

Sources: Proy3_Pedidos/models/address.js 10-16 Proy3_Pedidos/models/address.js 25-32

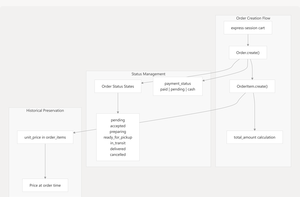
Many-to-Many Resolution

The restaurant-category relationship uses a junction table:



Sources: Proy3_Pedidos/def proyecto 222-234 Proy3_Pedidos/def proyecto 259-265

Order Processing Data Flow



Sources: Proy3_Pedidos/def proyecto 40-61 Proy3_Pedidos/def proyecto 270-280

Sequelize Configuration Standards

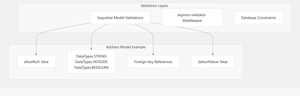
All models follow consistent configuration patterns:

Configuration	Value	Purpose
`timestamps`	`true`	Automatic `created_at` / `updated_at`
`underscored`	`true`	Snake_case field names
`tableName`	Explicit string	Override pluralization
`inCascade`	`true`	Enable cascade operations

Sources: Proy3_Pedidos/models/address.js 58-65

Model Validation Patterns

The schema implements validation at multiple levels:



Sources: Proy3_Pedidos/models/address.js 25-57 Proy3_Pedidos/def proyecto 399-400