

Last indexed: 11 August 2025 (172577)

- Overview
- Repository Structure
- Getting Started
- Project 1: Product Management (Proy1)
- Database & Configuration
- Product Management System
- Development Environment
- Project 2: Course Platform (Proy2\_Cursos)
- Data Models & Database
- Authentication & Middleware
- Development & Testing
- Project 3: Food Delivery Platform (Proy3\_Pedidos)
- Architecture & Dependencies
- Database Models & Schema
- Environment & Configuration
- Development Environment
- VS Code & Debugging
- Dependency Management

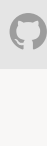
## Project 1: Product Management (Proy1)

> Relevant source files

### Purpose and Scope

This document covers Project 1 (Proy1), the first application in the IronHack Course 2 learning progression. Proy1 implements a basic CRUD (Create, Read, Update, Delete) system for product management with user authentication and role-based access control. The project serves as an introduction to full-stack web development using Node.js, Express.js, and SQLite.

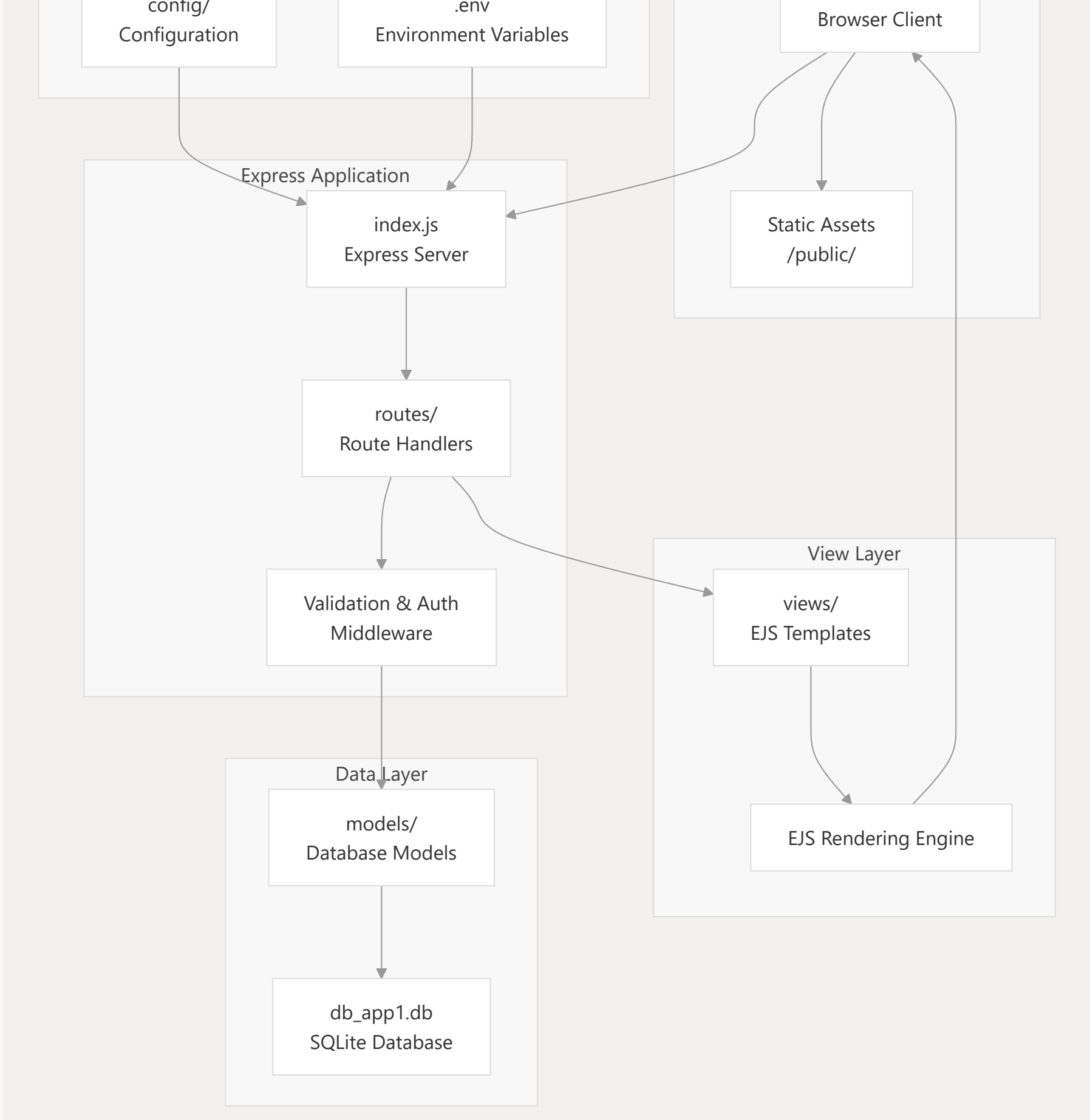
This project demonstrates fundamental web development concepts including server-side routing, database operations, template rendering, and basic authentication. For information about the overall repository structure and setup instructions, see [Overview](#). For more advanced projects with sophisticated authentication and ORM usage, see [Project 2: Course Platform](#) and [Project 3: Food Delivery Platform](#).


Sources:  [Proy1/README.md](#) 1-61

### Architecture Overview

Proy1 follows a traditional MVC (Model-View-Controller) architecture pattern built on Express.js. The application handles product management operations with user authentication and administrative controls.

### Application Architecture



Sources:  [Proy1/README.md](#) 44-57

### Technology Stack


Proy1 implements a focused technology stack suitable for learning fundamental web development concepts:

Component	Technology	Purpose
Runtime	Node.js	Server-side JavaScript execution
Framework	Express.js	Web application framework
Database	SQLite	Lightweight relational database
Templating	EJS	Dynamic HTML rendering
Validation	express-validator	Input data validation
Environment	dotenv	Environment variable management

### Key Dependencies

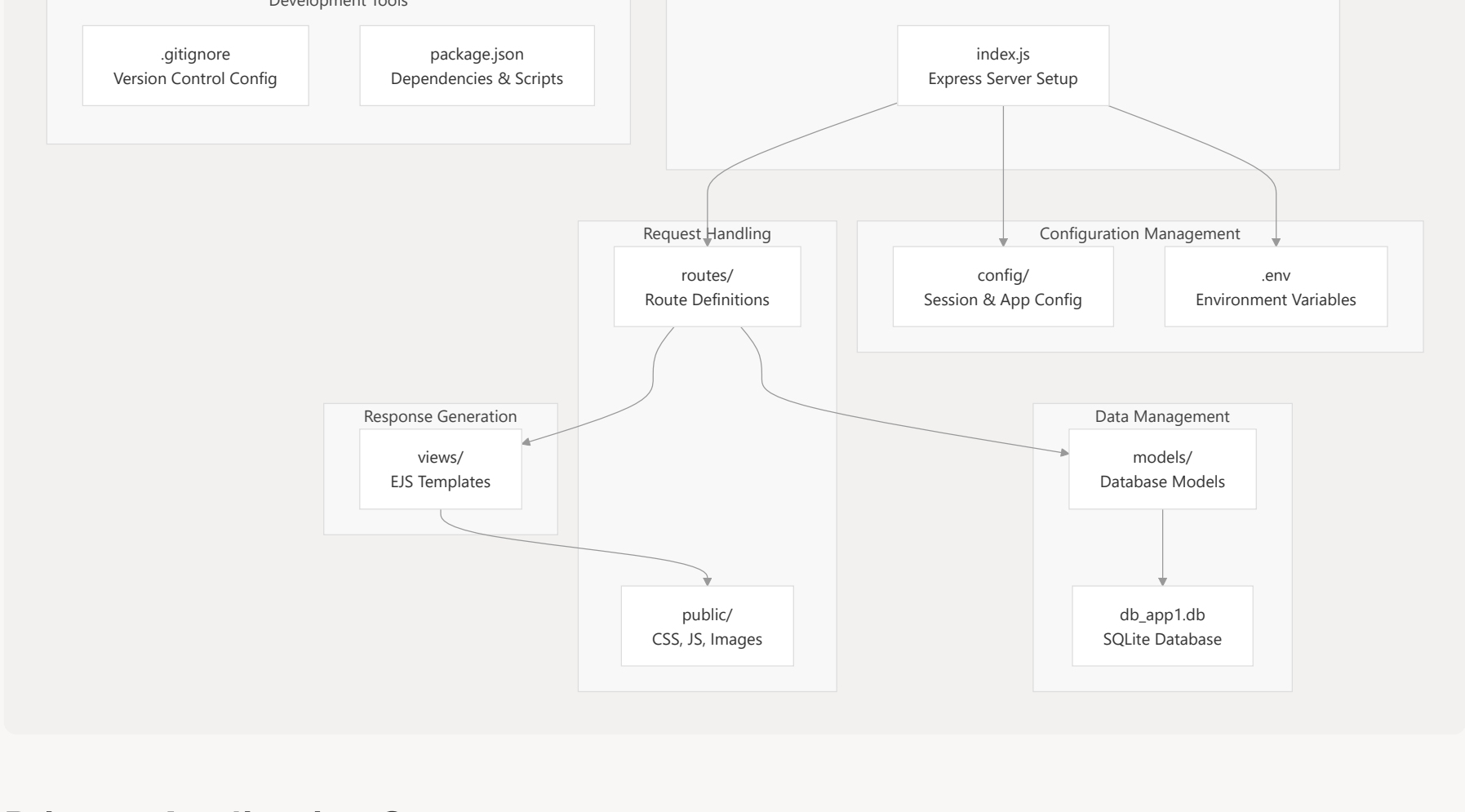
The project uses a minimal set of dependencies to maintain simplicity while demonstrating core concepts:

- Express.js:** Primary web framework for routing and middleware
- SQLite:** File-based database requiring no separate server process
- EJS:** Template engine for server-side rendering
- express-validator:** Middleware for validating user input
- dotenv:** Environment variable management for configuration

Sources:  [Proy1/README.md](#) 8-16


### Core Application Components

#### File Structure and Responsibilities



#### Primary Application Components

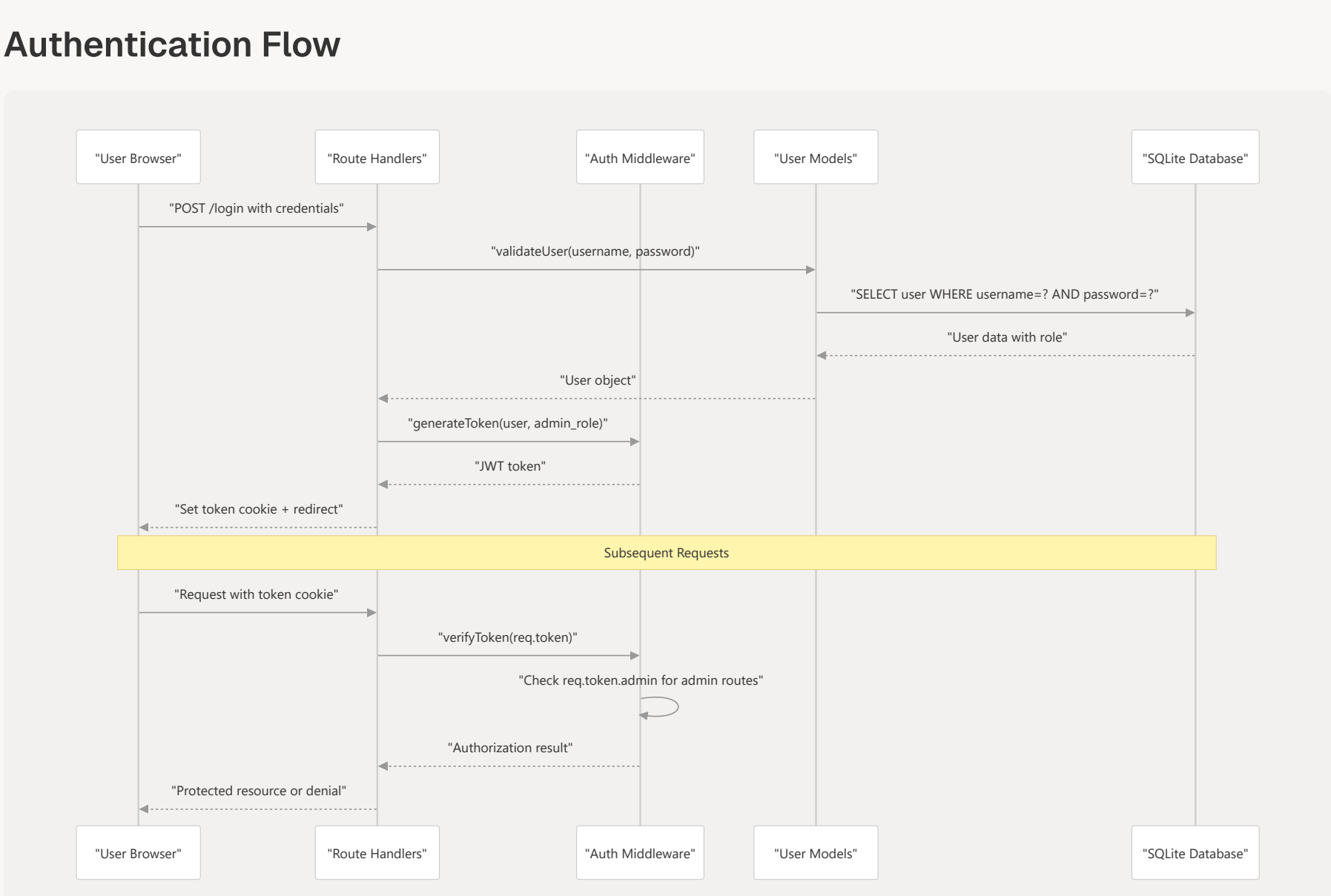
Directory	Purpose	Key Files
<code>config/</code>	Application configuration and session management	Session config, database config
<code>models/</code>	Database models and data access logic	Product model, User model
<code>routes/</code>	HTTP route handlers and business logic	Product routes, Auth routes
<code>views/</code>	EJS templates for HTML generation	Product views, Admin views
<code>public/</code>	Static assets served directly	CSS, JavaScript, images

Sources:  [Proy1/README.md](#) 44-57

### Authentication System

Proy1 implements a token-based authentication system with role-based access control. The system supports user login/logout functionality and administrative privilege management.

#### Authentication Flow



#### Token Structure and Validation

The authentication system uses a simple token-based approach with the following characteristics:

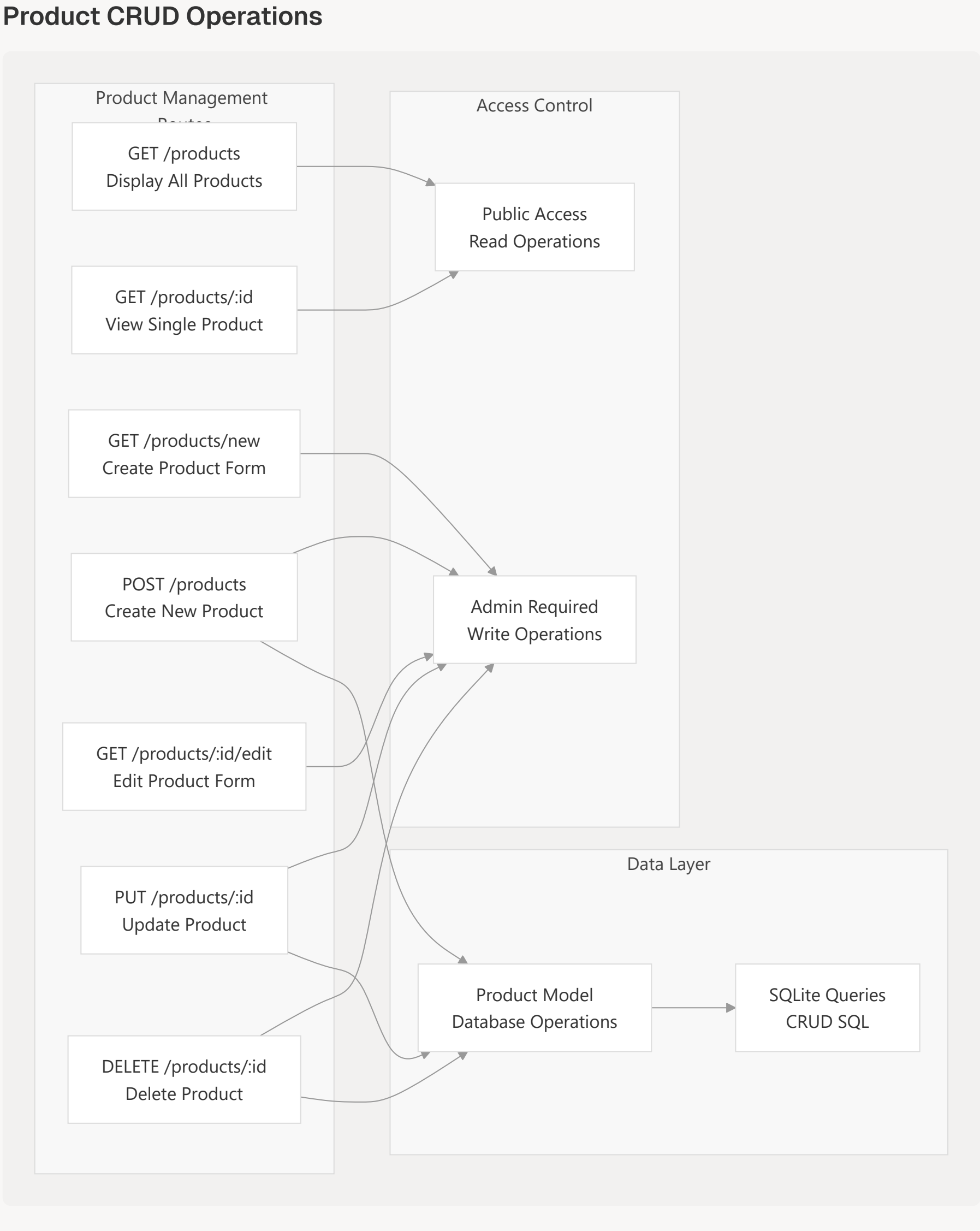
- Token Secret:** Configured via `TOKEN_SECRET` environment variable with default value `supersecrettoken`
- Admin Role Checking:** Uses `req?.token?.admin` property for administrative access control
- Session Management:** Basic session implementation for maintaining user state

Sources: Referenced from high-level architecture diagrams

### Product Management System

The core functionality of Proy1 centers around CRUD operations for product management. The system provides different access levels based on user roles.


#### Product CRUD Operations



#### Data Validation and Processing

The application uses `express-validator` middleware to ensure data integrity:

- Input Validation:** Server-side validation for all product form inputs
- Sanitization:** Data cleaning and normalization before database operations
- Error Handling:** Validation error messages displayed to users via EJS templates

Sources:  [Proy1/README.md](#) 15 referenced from technology stack description

### Development and Deployment

#### Local Development Setup

The project provides straightforward development setup with the following commands:

Command	Purpose
<code>npm install</code>	Install project dependencies
<code>npm start</code>	Start the Express server

#### Environment Configuration



The application uses environment variables for configuration management:

- PORT:** Server port (default: 3000)
- TOKEN\_SECRET:** Authentication token secret
- Database Path:** SQLite database file location

#### Version Control

The project includes appropriate `.gitignore` configuration to exclude:

- `node_modules/` directory
- `DB.Browser.for.SQLite-v3.13.1-win32/` database browser tools

Sources:  [Proy1/README.md](#) 17-40  [Proy1/.gitignore](#) 1-2