

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

## Development & Testing

> Relevant source files

This document covers the development environment setup, testing framework, and debugging configuration for Project 2 (Proy2\_Cursos). It explains how the Jest testing suite integrates with the authentication system and provides guidance on the development workflow.

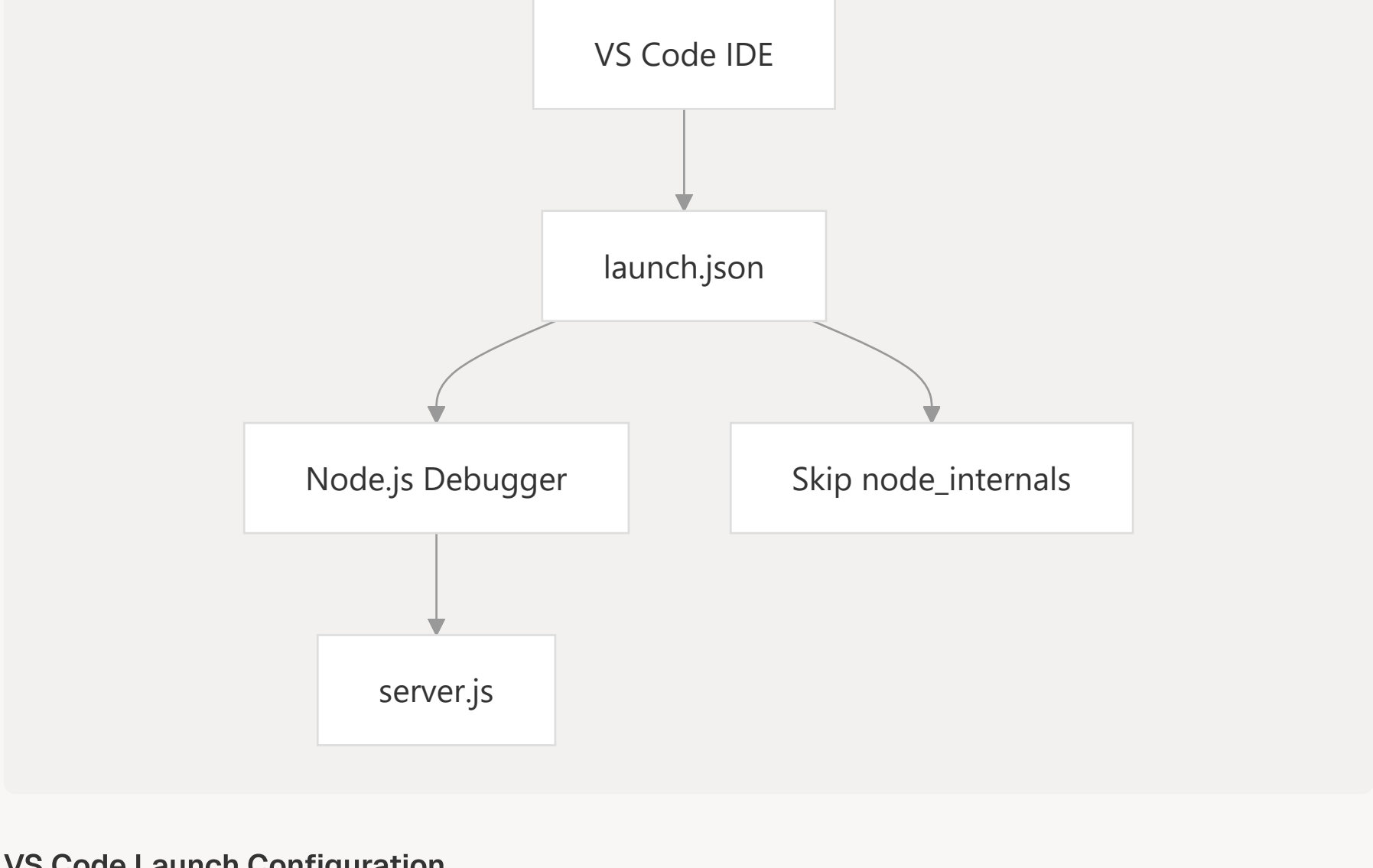
For information about the data models and database configuration, see [Data Models & Database](#). For details about the authentication and middleware implementation, see [Authentication & Middleware](#).

## Development Environment Setup

Project 2 includes a comprehensive development environment with VS Code integration and automated testing capabilities. The development setup supports both interactive debugging and automated testing workflows.


### VS Code Debugging Configuration

The project includes a pre-configured VS Code launch configuration for debugging the Express.js server:



### VS Code Launch Configuration

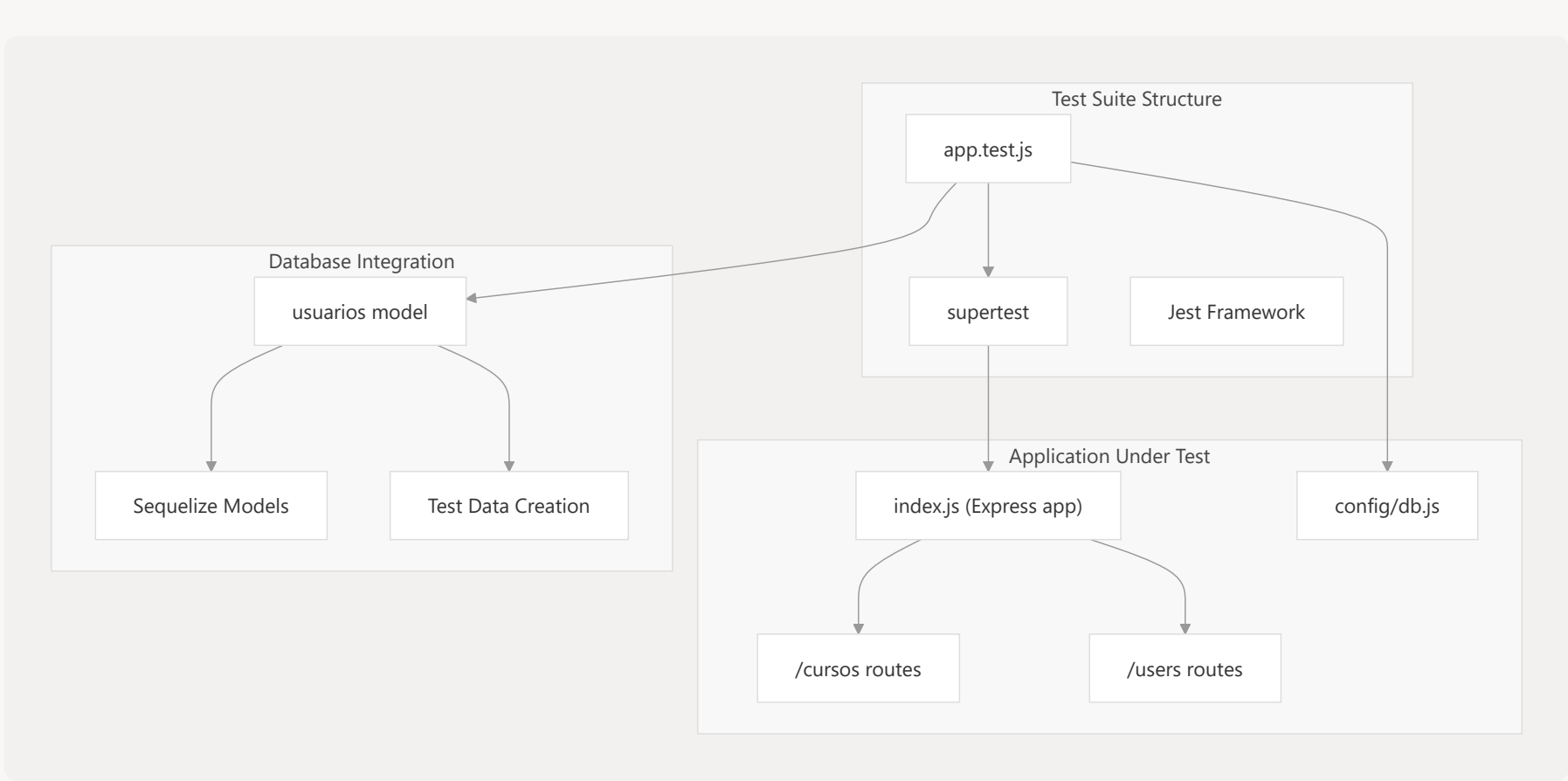
The debugging setup targets the main server entry point with internal Node.js files excluded from debugging sessions.

Sources:  Proy2\_Cursos/.vscode/launch.json 1-17

## Testing Framework


Project 2 implements a comprehensive Jest testing suite with **supertest** for HTTP endpoint testing and Sequelize integration for database operations during tests.

### Test Architecture



### Test Framework Components

The testing architecture integrates HTTP testing with database operations, enabling comprehensive end-to-end testing of authentication flows and protected routes.

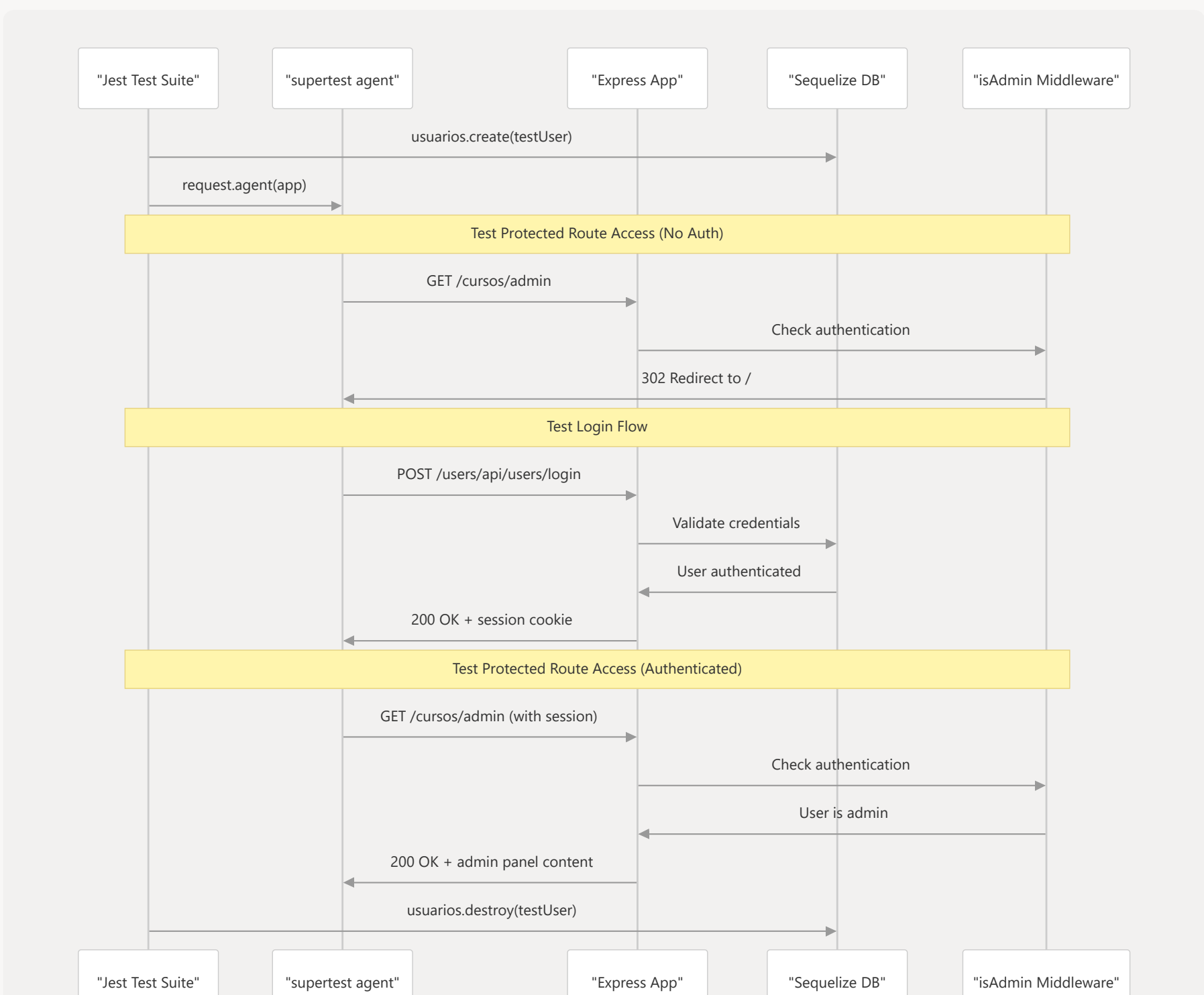
Sources:  Proy2\_Cursos/\_\_tests\_\_/app.test.js 1-80

### Test Patterns and Structure

The test suite implements several key testing patterns:


Test Pattern	Implementation	Purpose
Basic Route Testing	<code>GET /cursos/</code>	Verify basic endpoint availability
Authentication Flow Testing	Login → Protected Route Access	Test complete user authentication workflow
Setup/Teardown	<code>beforeAll</code> / <code>afterAll</code>	Database state management
Session Management	<code>request.agent()</code>	Maintain cookies across requests

## Authentication Flow Testing



### Authentication Test Flow

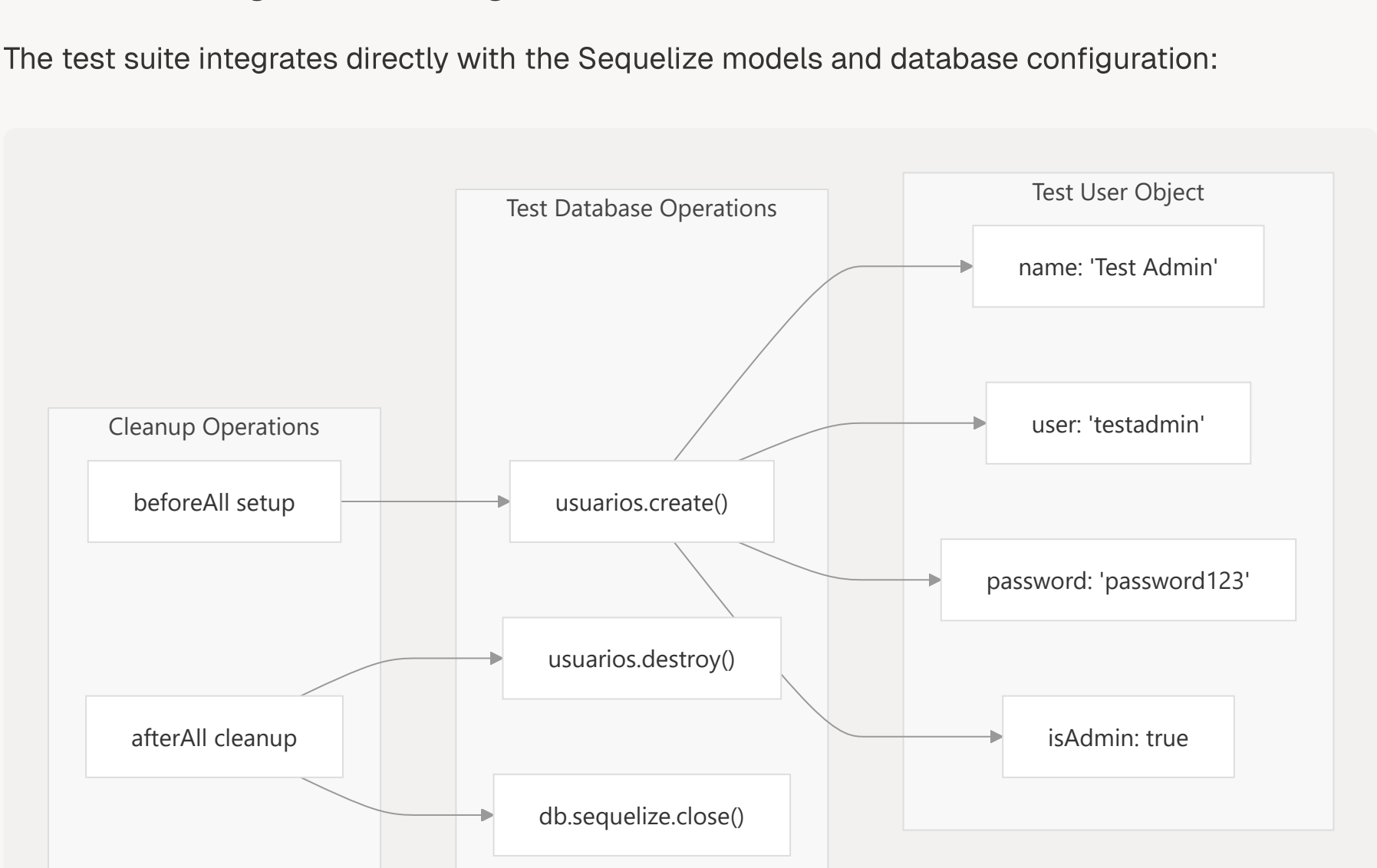
The test suite validates the complete authentication workflow from login to protected route access, ensuring proper session management and authorization.

Sources:  Proy2\_Cursos/\_\_tests\_\_/app.test.js 32-80

## Test Implementation Details

### Database Integration Testing

The test suite integrates directly with the Sequelize models and database configuration:



### Test Database Management


Tests create and clean up test data using the same Sequelize models as the application, ensuring consistency between test and runtime environments.

## Route Testing Specifications

The test suite covers critical application routes:

Route	Test Type	Expected Behavior
<code>GET /cursos/</code>	Basic functionality	Returns 200 status
<code>GET /cursos/admin</code> (no auth)	Authorization check	302 redirect to <code>/</code>
<code>POST /users/api/users/login</code>	Authentication	200 on valid credentials
<code>GET /cursos/admin</code> (authenticated)	Protected route access	200 + admin panel content

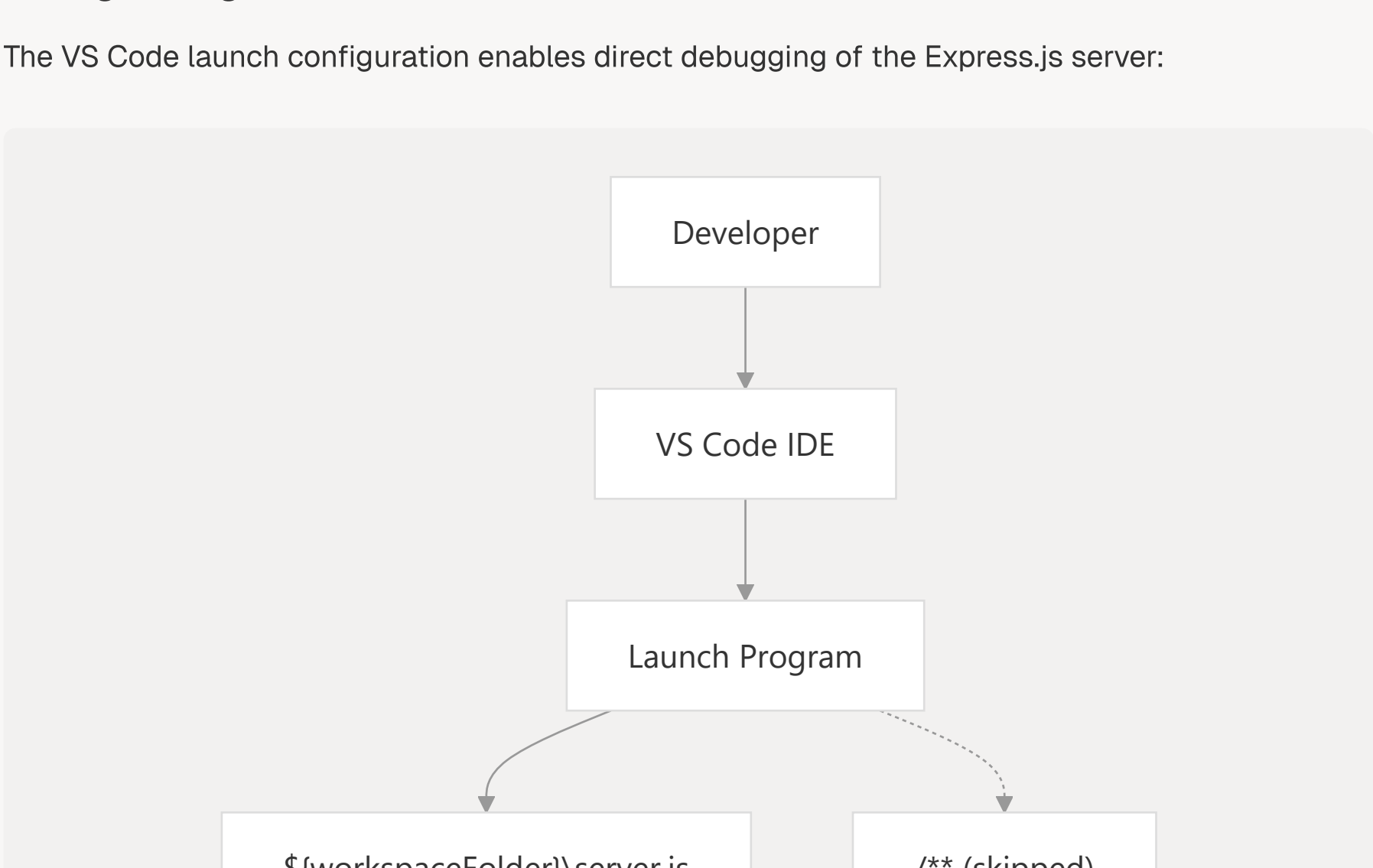
Sources:  Proy2\_Cursos/\_\_tests\_\_/app.test.js 8-24

 Proy2\_Cursos/\_\_tests\_\_/app.test.js 61-79

## Development Workflow

### Debug Configuration

The VS Code launch configuration enables direct debugging of the Express.js server:

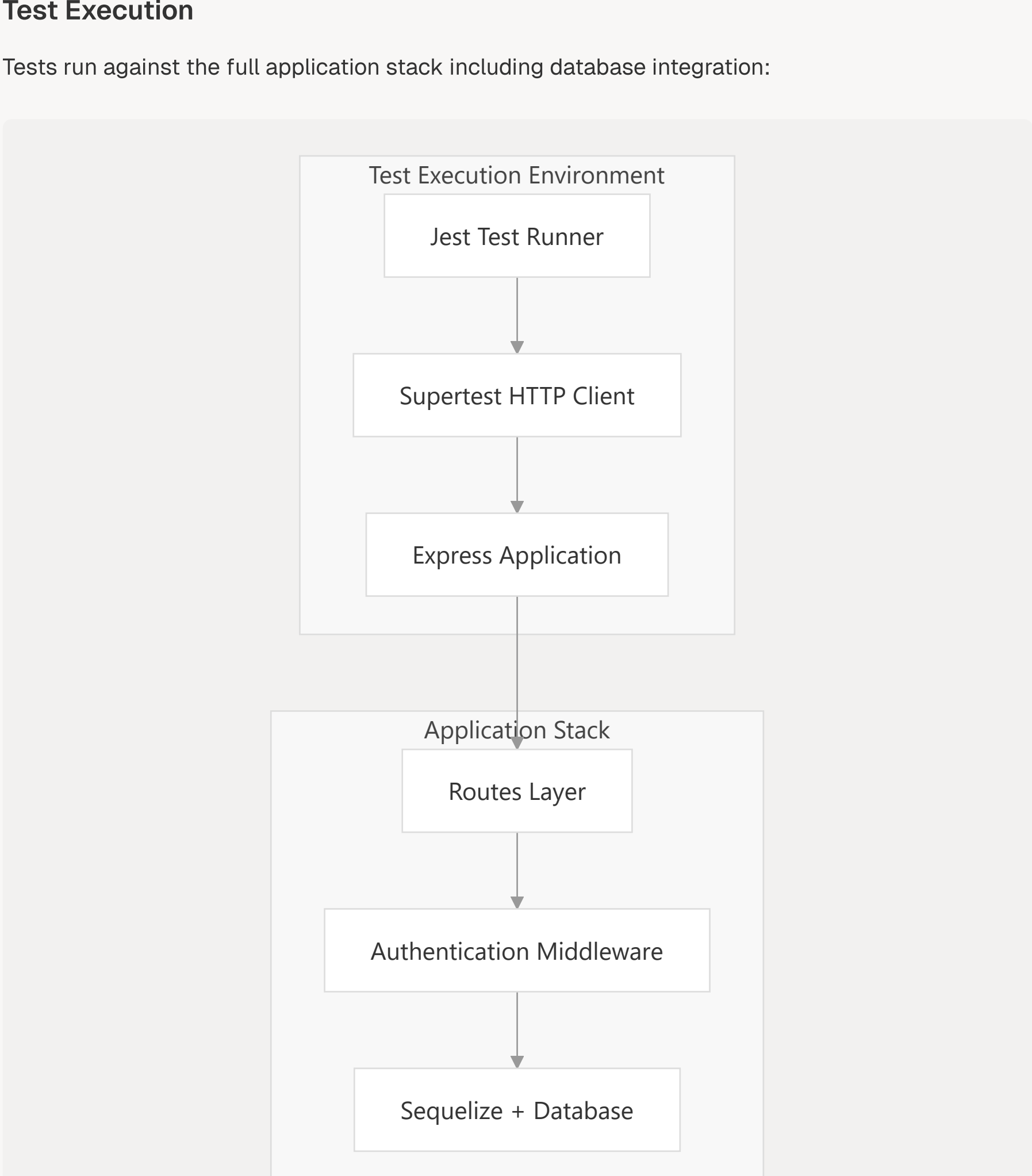


### Debug Process Flow

The debugging setup directly targets `server.js` while excluding Node.js internal files from the debugging session for cleaner debugging experience.

## Test Execution

Tests run against the full application stack including database integration:



### Test Stack Integration

The testing framework exercises the complete application stack, from HTTP requests through middleware to database operations.

Sources:  Proy2\_Cursos/\_\_tests\_\_/app.test.js 3-6  Proy2\_Cursos/.vscode/launch.json 7-16