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

# VS Code & Debugging

Relevant source files

This document covers the Visual Studio Code workspace configuration and debugging setup for the IronHack Course 2 monorepo. It details how VS Code is configured to support development across the three projects, the debugging configurations available, and the development workflow integration.

For information about dependency management across the monorepo, see Dependency Management. For project-specific development environments, see the individual project sections: Proy1 Development Environment, Proy2 Development & Testing.

## Workspace Configuration

The monorepo provides VS Code workspace configurations at multiple levels to support both project-specific and repository-wide development.

### Project-Specific Workspaces

**Proy3_Pedidos Workspace**

The most complex project includes a dedicated workspace file that provides isolated development environment configuration.


**Proy3_Pedidos Workspace Structure**

| Configuration | Value | Purpose |
|---|---|---|
| `folders[0].path` | `"."` | Sets workspace root to project directory |

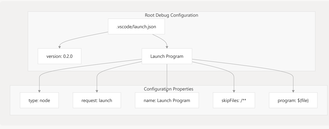
Sources: Proy3_Pedidos/Proy3_Pedidos.code-workspace 1-7

## Debugging Configuration

The repository implements a hierarchical debugging setup with configurations at both repository and project levels.

### Root-Level Debugging

The primary debugging configuration provides a generic launcher that can execute any currently opened file.




**Root Launch Configuration Properties**

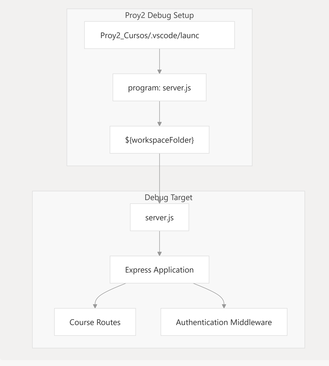
| Property | Value | Function |
|---|---|---|
| `type` | `"node"` | Specifies Node.js debugger |
| `request` | `"launch"` | Launches new debug session |
| `name` | `"Launch Program"` | Display name in debug menu |
| `skipFiles` | `["<node_internals>/**"]` | Excludes Node.js internal files |
| `program` | `"${file}"` | Executes currently active file |

Sources: .vscode/launch.json 1-17

### Project-Specific Debugging

**Proy2_Cursos Debug Configuration**

The course management project provides a specialized debugging setup that targets the main server file directly.



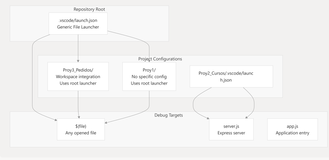

**Proy2_Cursos Launch Configuration**

| Property | Value | Function |
|---|---|---|
| `program` | `"${workspaceFolder}\\server.js"` | Targets main server file |
| Path Resolution | Windows-style backslashes | Platform-specific path handling |

Sources: Proy2_Cursos/.vscode/launch.json 1-17

## Debug Configuration Hierarchy

The debugging setup follows a hierarchical pattern that provides flexibility for different development scenarios.




**Configuration Usage Patterns**

| Project | Configuration Source | Target | Use Case |
|---|---|---|---|
| Proy1 | Root `.vscode/launch.json` | `${file}` | Debug any file individually |
| Proy2_Cursos | Project `.vscode/launch.json` | `server.js` | Debug full application |
| Proy3_Pedidos | Root + Workspace | `${file}` | Flexible debugging with workspace features |

Sources: .vscode/launch.json 1-17 Proy2_Cursos/.vscode/launch.json 1-17

## Development Workflow Integration

The VS Code configuration supports multiple development workflows across the monorepo structure.

### File-Based Debugging Workflow

The root configuration enables rapid testing and debugging of individual files without full application startup.

**Workflow Steps:**

1. Open any JavaScript file in VS Code
2. Set breakpoints in the code
3. Use "Launch Program" debug configuration
4. Debug executes `${file}` directly

### Application-Level Debugging Workflow

Proy2_Cursos demonstrates application-level debugging for full Express.js application testing.

**Workflow Steps:**

1. Navigate to `Proy2_Cursos` directory
2. Open VS Code with project-specific configuration
3. Use "Launch Program" to start `server.js`
4. Debug full application with all middleware and routes

### Workspace Development Workflow

Proy3_Pedidos combines workspace features with debugging for enhanced development experience.

**Workflow Features:**

- Isolated project environment
- Workspace-specific settings and extensions
- Integrated debugging with root configuration
- Project-focused development context

Sources: Proy3_Pedidos/Proy3_Pedidos.code-workspace 1-7 .vscode/launch.json 14 Proy2_Cursos/.vscode/launch.json 14