Getting Started

> Relevant source files

This document provides step-by-step instructions for setting up and running the IronHack Course 2 monorepo. It covers the initial installation process, dependency management, and launching the three web applications either individually or concurrently.

For detailed information about the repository structure and npm workspaces configuration, see Repository Structure. For project-specific setup and configuration details, refer to the individual project sections: Project 1, Project 2, and Project 3.

Prerequisites

Before setting up the monorepo, ensure your development environment meets the following requirements:

Requirement	Description	Version
Node.js	JavaScript runtime environment	Required
npm	Node package manager	Required
Git	Version control system	Required for cloning

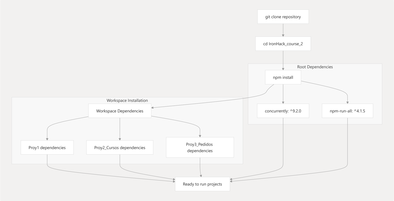
The repository uses npm workspaces to manage dependencies across multiple projects, which requires a recent version of npm that supports workspace functionality.

Sources: README.md 6-7

Installation Process

The monorepo uses a centralized dependency management approach through npm workspaces. The installation process involves cloning the repository and installing dependencies for all three projects simultaneously.

Installation Flow



Step-by-Step Installation

1. **Clone the repository:**

```
git clone https://github.com/emanavas/IronHack_course_2
cd IronHack_course_2
```

2. **Install all dependencies:**

```
npm install
```

This single command installs dependencies for all three workspace projects as defined in the `workspaces` array in package.json 13-17 plus the root-level dependencies `concurrently` and `npm-run-all` used for project orchestration.

Sources: README.md 10-16 package.json 13-25

Running the Projects

The monorepo provides flexible execution options through npm scripts defined in the root `package.json`. You can run projects individually or launch all three simultaneously.

Project Execution Matrix

Project	npm Script	Workspace Target	Port	URL
Project 1	`start:proy1`	`Proy1`	3000	http://localhost:3000
Project 2	`start:proy2`	`Proy2_Cursos`	3002	http://localhost:3002
Project 3	`start:proy3`	`Proy3_Pedidos`	3003	http://localhost:3003

Individual Project Execution

Each project can be started independently using workspace-specific npm scripts:

```
# Start Project 1 (Product Management)
npm run start:proy1

# Start Project 2 (Course Platform)
npm run start:proy2

# Start Project 3 (Food Delivery Platform)
npm run start:proy3
```

The npm scripts use the `--workspace` flag to target specific project directories as defined in package.json 7-9

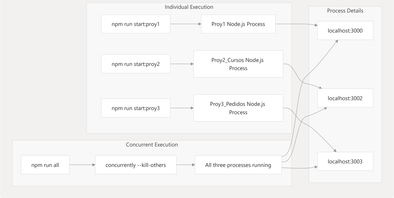
Concurrent Execution

To run all three projects simultaneously, use the `all` script which leverages the `concurrently` package:

```
npm run all
```

This command executes the concurrent startup flow defined in package.json 11 using the `--kill-others` flag to ensure all processes terminate together when one is stopped.

Execution Flow Diagram



Sources: README.md 18-56 package.json 6-11

Project Verification

After starting the projects, verify they are running correctly by accessing their respective URLs:

Verification Checklist

Project	URL	Expected Response
Proy1	http://localhost:3000	Product management interface
Proy2_Cursos	http://localhost:3002	Course platform login/dashboard
Proy3_Pedidos	http://localhost:3003	Food delivery platform interface

Each project runs independently with its own Express.js server instance and database connection. The port assignments are configured within each project's individual startup scripts.

Troubleshooting Common Issues

Port Conflicts

If any project fails to start due to port conflicts, check for existing processes using the assigned ports (3000, 3002, 3003) and terminate them before restarting.

Dependency Issues

If dependencies fail to install correctly, try clearing npm cache and reinstalling:

```
npm cache clean --force
rm -rf node_modules package-lock.json
npm install
```

Workspace Resolution

If npm workspace commands fail, ensure you're running the commands from the root directory where the main `package.json` with workspace configuration is located.

Sources: package.json 1-26 README.md 1-57