**GUIA PORTAFOLIO**

# 📂 Cómo estructurar tu portafolio

Tu portafolio debería estar en **GitHub/GitLab/Bitbucket** y organizado de manera clara. Incluye:

## **1.README profesional** en cada proyecto:

* + Breve descripción.
  + Tecnologías usadas.
  + Cómo instalar/ejecutar.
  + Screenshots o diagramas.
  + Explicación de arquitectura (si aplica).

## 2.Currículum en tu repositorio principal (o link a tu CV en LinkedIn).

## **3.Demos desplegados** (en Railway, Render, Vercel, AWS, etc.) para que reclutadores puedan probarlos sin necesidad de clonar el repo.

# 📑 Cómo presentar tu portafolio

* Un repositorio llamado **java-backend-portfolio** que tenga un **README principal** con:
  + ✨ Breve bio profesional.
  + 🚀 Lista de proyectos (con links).
  + 🛠️ Tecnologías que dominas.
  + 🌎 Links de despliegues (si los tienes).
  + 📧 Contacto (LinkedIn, correo, etc.).

Ejemplo de estructura:

📦 java-backend-portfolio

┣ 📂 task-manager-api

┣ 📂 user-auth-service

┣ 📂 ecommerce-backend

┣ 📂 booking-microservices

┣ 📂 integrations-api

┣ 📂 stock-market

┣ README.md

# 🚀 Proyectos que puedes incluir

Aquí te propongo una ruta de proyectos, cada uno agregando más complejidad:

## 1. CRUD API simple (task-manager-api)

* **Tecnologías:** Spring Boot, Spring Data JPA, H2/PostgreSQL.
* **Ejemplo:** API de gestión de tareas (to-do list).
* Objetivo: mostrar que dominas CRUD + REST + conexión a DB.

## 2. API con autenticación y roles(user-auth-service)

* **Tecnologías:** Spring Boot, Spring Security, JWT.
* **Ejemplo:** Sistema de usuarios con login/registro y roles (admin/user).
* Objetivo: demostrar autenticación, seguridad y autorización.

## 3. E-commerce básico (mini backend)

* **Tecnologías:** Spring Boot, JPA, PostgreSQL/MySQL, Swagger.
* **Funcionalidades:**
  + Productos
  + Carrito
  + Órdenes
  + Usuarios con roles (admin puede crear productos)
* Objetivo: mostrar modelado de datos, relaciones, endpoints RESTful.

## 4. Microservicios + comunicación(booking-microservices)

* **Tecnologías:** Spring Cloud, Eureka, API Gateway, Docker.
* **Ejemplo:** Sistema de reservas (usuarios, reservas, pagos).
* Objetivo: mostrar que entiendes microservicios, escalabilidad y comunicación entre servicios.

## 5. Proyecto con DevOps / despliegue(integrations-api)

* **Tecnologías:** Docker, Jenkins/GitHub Actions, Railway/Render/AWS.
* **Ejemplo:** Desplegar tu e-commerce en la nube con CI/CD.
* Objetivo: mostrar que sabes **llevar tu app a producción**.

## 6. Proyecto con integraciones externas(stock-market)

* **Ejemplo:** API de clima, conversor de divisas o stock market.
* **Objetivo:** demostrar que puedes consumir APIs externas.

# 🎯 Consejos extra

* Documenta bien tu código (JavaDocs, README).
* Usa **Swagger/OpenAPI** en tus APIs para documentación automática.
* Crea **tests unitarios** (JUnit, Mockito).
* Mantén commits claros en Git.
* Opcional: hacer un **blog técnico** en GitHub Pages o Medium para mostrar lo que aprendes.