

# Indicaciones para DevOps - Sistema de Gestión de Paquetes

## 1. Configuración del Repositorio y Control de Versiones

### 1.1 Orden:

- ◆ Configurar un repositorio en GitHub con una estructura clara de ramas.

### 1.2 Explicación:

El repositorio debe seguir un flujo de ramas estructurado para organizar el código de manera eficiente:

- ◆ `main`: Contiene el código estable listo para producción.
- ◆ `develop`: Contiene el código en desarrollo antes de ser fusionado con `main`.
- ◆ `feature/{nombre}`: Ramas para nuevas funcionalidades (ejemplo: `feature/gestion-pagos`).
- ◆ `hotfix/{nombre}`: Ramas para corregir errores críticos en producción.

### 1.3 Acción requerida:

- ◆ Crear el repositorio en GitHub.
- ◆ Configurar las ramas mencionadas y proteger `main` y `develop` contra cambios directos.
- ◆ Implementar un flujo de trabajo con pull requests y revisiones obligatorias antes de fusionar código.

## 2. Implementación de Pruebas Automáticas

### 2.1 Orden:

- ◆ Implementar pruebas unitarias, de integración y de seguridad para cada funcionalidad clave.

## 2.2 Explicación:

Las pruebas aseguran que el software funcione correctamente antes de ser desplegado. Se deben incluir:

- ◆ Pruebas unitarias: Verifican funciones individuales del código.
- ◆ Pruebas de integración: Aseguran que los módulos interactúan correctamente.
- ◆ Pruebas de seguridad: Detectan vulnerabilidades en el código y dependencias.

## 2.3 Acción requerida:

- ◆ Implementar backend
- ◆ Escribir pruebas para:
  - a. Gestión de paquetes: Creación, actualización, eliminación y clasificación.
  - b. Gestión de envíos: Creación de envíos y rastreo.
  - c. Gestión de usuarios: Registro, autenticación y permisos.
  - d. Gestión de pagos y facturación: Generación de facturas y procesamiento de pagos.

# 3. Contenerización y Orquestación

## 3.1 Orden:

- ◆ Crear imágenes Docker del backend y un `docker-compose.yml` para el entorno de desarrollo.

### 3.2 Explicación:

- ♦ Docker permite empaquetar el software en contenedores, facilitando su ejecución en cualquier entorno. `docker-compose.yml` ayuda a gestionar múltiples contenedores en desarrollo.

### 3.3 Acción requerida:

- ♦ Escribir un `Dockerfile` para el backend con:
  - a. Instalación de dependencias.
  - b. Exposición del puerto adecuado.
  - c. Configuración de variables de entorno.
- ♦ Crear `docker-compose.yml` para levantar base de datos y backend en desarrollo.

## 4. Integración Continua (CI)

### 4.1 Orden:

- ♦ Configurar un pipeline en GitHub Actions CI/CD para ejecutar pruebas y análisis de código automáticamente en cada cambio.

### 4.2 Explicación:

- ♦ CI garantiza que cada modificación en el código es probada antes de fusionarse, evitando errores en producción.

### 4.3 Acción requerida:

- ♦ Crear un workflow CI/CD en `.github/workflows/ci.yml` o `.gitlab-ci.yml` que haga lo siguiente:
  - a. Ejecute pruebas unitarias e integración en cada push.
  - b. Analice el código con SonarQube para detectar problemas.
  - c. Construya las imágenes Docker después de las pruebas.
  - d. Publique los artefactos en un registro de contenedores.

## 5. Entrega y Despliegue Continuo (CD)

### 5.1 Orden:

- ♦ Configurar despliegues automáticos en staging y controlados en producción.

### 5.2 Explicación:

- ♦ La Entrega Continua (CD) permite que el software esté siempre listo para desplegarse. Se hacen pruebas en un entorno de staging antes de pasar a producción.

### 5.3 Acción requerida:

- ♦ Staging:
  - a. Implementar un despliegue automático en un entorno de pruebas después de pasar CI.
- ♦ Producción:
  - b. Desplegar manualmente desde GitHub con aprobación del equipo DevOps.
  - c. Configurar rollback automático si falla el despliegue.

## **6. Seguridad y Cumplimiento**

### **6.1 Orden:**

- ◆ Implementar autenticación segura y escaneo de vulnerabilidades.

### **6.2 Explicación:**

- ◆ La seguridad es crucial para proteger la información del sistema y los usuarios.

### **6.3 Acción requerida:**

- ◆ Autenticación segura: Aplicar cifrado en Hash.
- ◆ Escaneo de vulnerabilidades:
  - a. Configurar Alertas de seguridad en GitHub.

#	Tarea	Responsable	Estado (Completado / Pendiente)
1	Configuración del Repositorio y Control de Versiones	Ingeniero DevOps	Hecho
2	Implementación de Pruebas Automáticas	Analista / Ingeniero DevOps	Hecho
3	Contenerización y Orquestación	Ingeniero DevOps	En progreso
4	Integración Continua (CI)	Analista / Desarrolladores	En progreso
5	Entrega y Despliegue Continuo (CD)	Ingeniero DevOps	Pendiente
6	Seguridad y Cumplimiento	Analista / Desarrolladores / Ingeniero DevOps	Hecho