

CS 2031 - DBP

# Desarrollo Basado en Plataformas

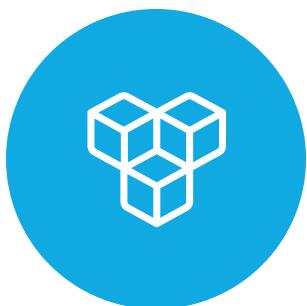
Jesus Bellido

# ¿Qué haremos hoy?

SEMANA 4 - Auditorio

- 1 **Conociendo el Testing**
- 2 **Tipos de Test**
- 3 **TestContainers**
- 4 **GitHub Actions**
- 5 **Anuncios**

# Logros



**Conocer la importancia e impacto de aplicar Testing en el desarrollo de software**



**Comprender los tipos de Testing, entender el rol de los TestContainers**



**Desarrollar estrategias de eventos asincronos**

# Testing

**TRANSFORMATEC**

# ¿Qué es el testing de software?

“El testing es el proceso de evaluar y verificar que un producto de software o una aplicación haga lo que se supone que debe hacer.”

IBM

# ¿Para qué sirve el testing?



## Detección de errores

Permite identificar y corregir bugs o defectos antes del lanzamiento



## Verificación de requisitos

Asegura que el software cumple las especificaciones y requisitos iniciales



## Mejora de la experiencia de usuario

Verifica aspectos como la usabilidad, la accesibilidad y la interfaz de usuario para garantizar que sean intuitivos y eficientes.



## Prevención de problemas futuros

Reduce costos y tiempo de resolver problemas complejos

**El testing de software es fundamental para asegurar la calidad y eficacia de las aplicaciones.**

# Pan para hoy, hambre para mañana: La analogía del testing de software

## La tentación del ahorro inicial

Omitir el testing o reducir su alcance en las etapas iniciales del desarrollo de software puede parecer una manera eficiente de reducir costos y acelerar el tiempo desarrollo.

## Los costos ocultos de omitir el testing

Al igual que el hambre que sigue después de consumir el pan sin prever el futuro, los defectos de software no detectados debido a pruebas insuficientes pueden resultar en problemas graves

## La inversión en pruebas como seguro a largo plazo

Si bien las pruebas representan un gasto inicial, los beneficios a largo plazo son significativos. Detectar y corregir problemas desde el principio evita costos de desarrollo adicionales y reduce el soporte necesario después del lanzamiento

**REGRESIÓN**  
**ESTRÉS**  
**CARGA**  
**ALFA**  
**SMOKE**  
**EXPLORATORIA**

**CÓDIGO**  
**ACEPTACIÓN**  
**INTEGRACIÓN**  
**SEGURIDAD**  
**FUNCIONAL**  
**UNITARIAS**  
**USABILIDAD**

**RENDIMIENTO**  
**BETA**  
**SANIDAD**  
**COMPATIBILIDAD**

# Pruebas de Seguridad (Security Testing)

## Identificación de vulnerabilidades

Buscar exponer las debilidades del sistema que podrían ser explotadas por amenazas externas, como inyecciones SQL o XSS

## Protección de datos

Asegurar la confidencialidad, integridad y disponibilidad de los datos, protegiendo así la información contra accesos no autorizados

## Cumplimiento de estándares

Verificar que el software cumpla con las normativas de seguridad aplicables, lo que es crucial para aplicaciones en sectores regulados como la banca y la salud

## Pruebas rigurosas

Incluir técnicas como pruebas de penetración y evaluación de vulnerabilidades, que simulan ataques para evaluar la robustez del sistema

**Las pruebas de seguridad son esenciales para proteger sistemas y datos ante posibles amenazas**

# Pruebas E2E



## Verificación del flujo completo

Las pruebas E2E verifican el flujo completo de la aplicación de principio a fin para asegurar que todos los componentes funcionen juntos.



## Cobertura de funciones interconectadas

Son cruciales en sistemas complejos donde diferentes módulos y servicios interactúan entre sí.



## Experiencia de usuario final

Aseguran que la experiencia de usuario sea consistente y satisfactoria durante todo el proceso.



## Pruebas en ambiente real

Se ejecutan en un entorno que imita de manera fiel el ambiente real de producción.

**Las pruebas E2E son esenciales para verificar el funcionamiento integral de aplicaciones complejas antes del despliegue a producción.**

# Pruebas de Estrés (Stress Testing)



## Evaluación bajo Carga Extrema

Determinan cómo se comporta el sistema bajo condiciones de estrés inusuales, como cargas de trabajo muy altas o recursos de hardware limitados.



## Identificación de Puntos de Falla

Revelan los límites operativos del sistema y ayudan a identificar qué componentes fallan primero bajo presión.



## Estabilidad y Confiabilidad

Aseguran que el sistema se mantenga estable y confiable incluso durante picos de demanda o fallos de componentes.



## Escenarios Críticos

Son esenciales para sistemas críticos donde el fallo bajo condiciones extremas puede tener consecuencias graves.

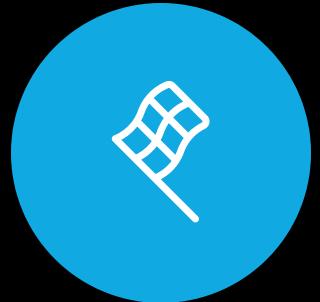
**Las pruebas de estrés son esenciales para evaluar el rendimiento y la confiabilidad de un sistema bajo condiciones extremas.**

# Software para Stress Testing

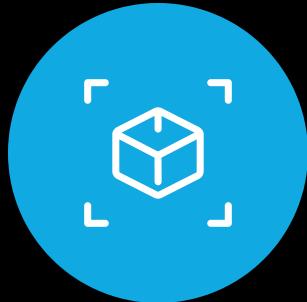


- **Herramienta para prueba de carga y estrés**
- **Ayuda a medir el rendimiento de aplicaciones web y otros servicios**

# Pruebas Unitarias (Unit Testing)



**Validar funcionamiento**  
Las pruebas unitarias validan que cada unidad funcional del software funcione correctamente.



**Aislamiento de componentes**  
Se prueban los componentes de forma aislada para evitar interferencias.



**Detección temprana de errores**  
Permiten identificar y corregir errores temprano en el desarrollo.



**Automatización**  
Se pueden automatizar con frameworks como JUnit para ejecutarse rápida y repetidamente.

**Las pruebas unitarias son esenciales en el desarrollo de software para validar cada unidad funcional y detectar errores temprano.**

# Reducir acoplamiento en las pruebas unitarias

- ✓ **Inyección de dependencias**  
Permite sustituir dependencias reales por mocks durante las pruebas
- ✓ **Interfaces**  
Permiten enfocarse en la lógica del código sin preocuparse por otras partes del sistema
- ✓ **Mocking y stubbing**  
Para simular comportamientos y controlar interacciones externas
- ✓ **Refactorización**  
Para reducir acoplamiento cuando las pruebas se vuelven difíciles de mantener

# Ejemplo: Test Unit

Testeo de un componente que divide dos enteros a y b



```
@GetMapping("/div/{a}/{b}")
public ResponseEntity<Integer> add(
    @PathVariable Integer a,
    @PathVariable Integer b
) {
    if (b == 0) {
        return ResponseEntity
            .badRequest().build();
    }
    return ResponseEntity.ok(a / b);
}
```



```
@Test
public void testDiv() throws Exception {
    mockMvc.perform(get("/div/{a}/{b}", 4, 2))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.value").value(2));
}

@Test
public void testDivByZero() throws Exception {
    mockMvc.perform(get("/div/{a}/{b}", 4, 0))
        .andExpect(status().isBadRequest());
}
```

# Kahoot!

Testing



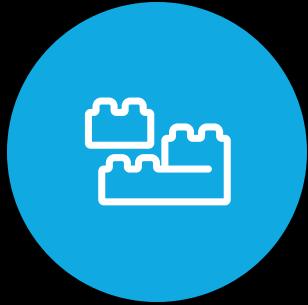
# TestContainers

# TestContainers



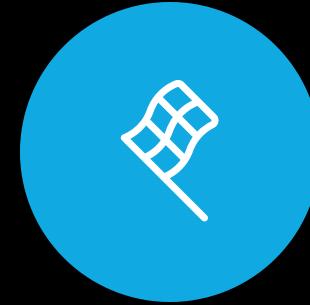
## ¿Qué es TestContainers?

TestContainers es una biblioteca de Java que facilita la creación y gestión de contenedores Docker en pruebas de integración.



## Funcionalidad

Permite a los desarrolladores escribir pruebas de integración que involucren componentes que requieren contenedores Docker, como bases de datos, servidores web, servicios de terceros, etc.



## ¿Cómo funciona?

TestContainers maneja la creación, inicialización, y destrucción de los contenedores Docker automáticamente durante la ejecución de las pruebas.

**TestContainers es una herramienta útil para gestionar contenedores Docker en pruebas de integración, simplificando la configuración de entornos de prueba y mejorando la confiabilidad.**

# TestContainers para Persistencia

## Entorno aislado

Testcontainers inicia una instancia de PostgreSQL en un contenedor Docker antes de ejecutar las pruebas. Esto garantiza un entorno limpio y aislado.

## Configuración consistente

Se puede configurar el contenedor con una versión específica de PostgreSQL y parámetros predefinidos. Esto asegura un entorno de pruebas consistente.

## Pruebas de integración realistas

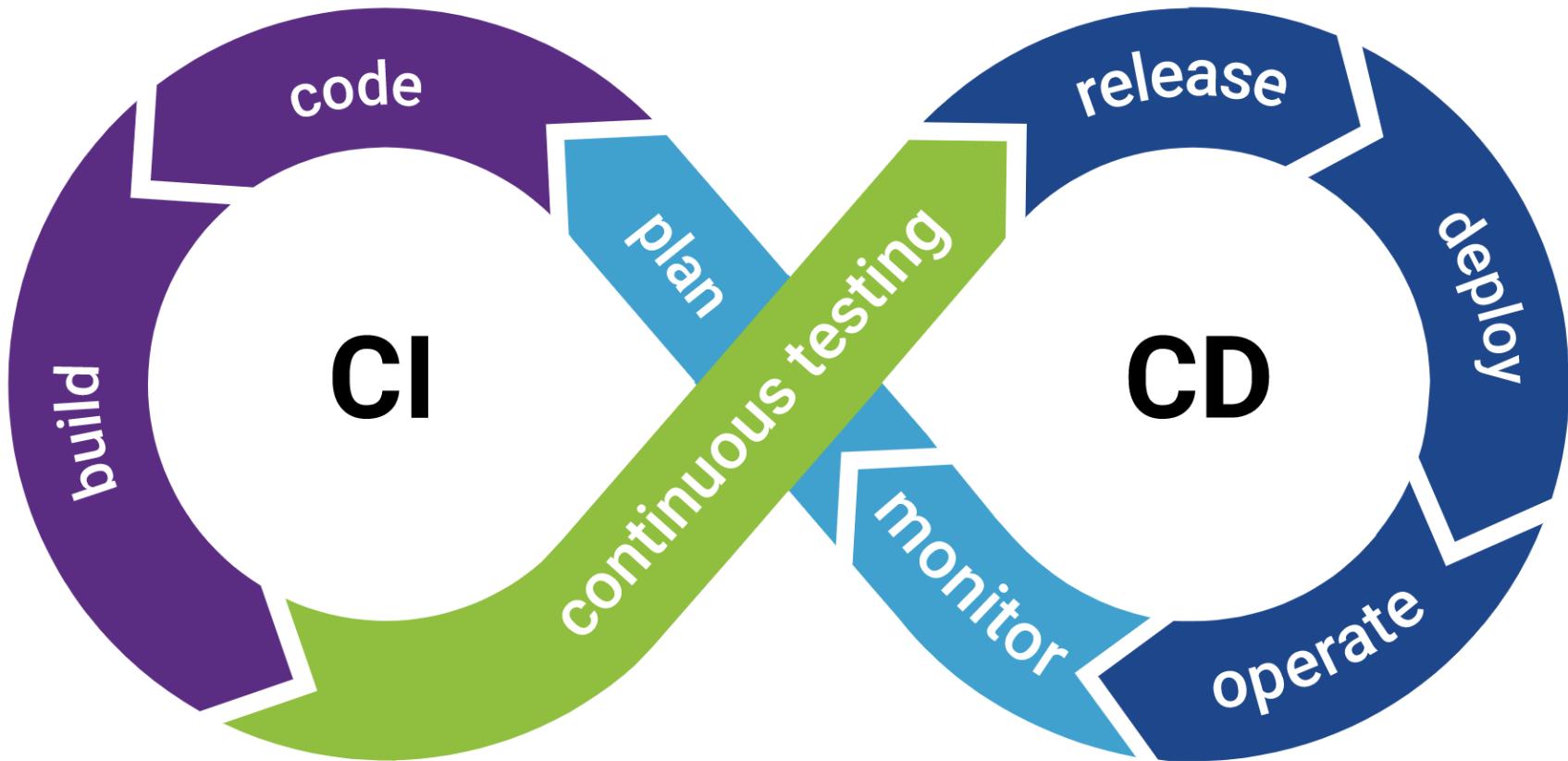
Al usar un contenedor real de PostgreSQL, las pruebas unitarias se convierten en pruebas de integración que interactúan con una base de datos real.

## Automatización y CI/CD

Los Testcontainers se integran fácilmente en CI/CD para ejecutar pruebas de persistencia automáticamente.

# Github Actions

**TRANSFORMATEC**



# CI/CD

## 1 Plan

Los equipos definen los requisitos y el alcance del software, incluidas las nuevas características y las correcciones a realizar.

## 2 Code

Los desarrolladores escriben código para nuevas funciones, mejoras o correcciones de errores. El código debe estar comentado y seguir guías de estilo.

## 3 Build

El código fuente se compila en código ejecutable. Se gestionan las dependencias y bibliotecas necesarias.

## 4

## Continuous Testing

Ejecutar pruebas automatizadas sobre el código para asegurar que los cambios no rompan funcionalidades existentes.

## 5

## Monitor

Observar y verificar el rendimiento del software para asegurar que funciona como se espera y recolectar datos.

## 6

## Release

Preparación y disponibilidad del software para ser entregado o lanzado a los usuarios.

## 7

## Deploy

Despliegue del software en el entorno de producción donde es accesible para los usuarios.

# GitHub Actions



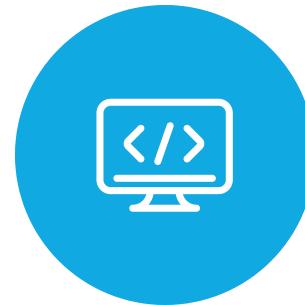
**GitHub Actions facilita la automatización de flujos de trabajo de desarrollo de software**

Permite automatizar tareas como compilación, pruebas, implementación, etc.



**Proporciona infraestructura para ejecutar flujos de trabajo**

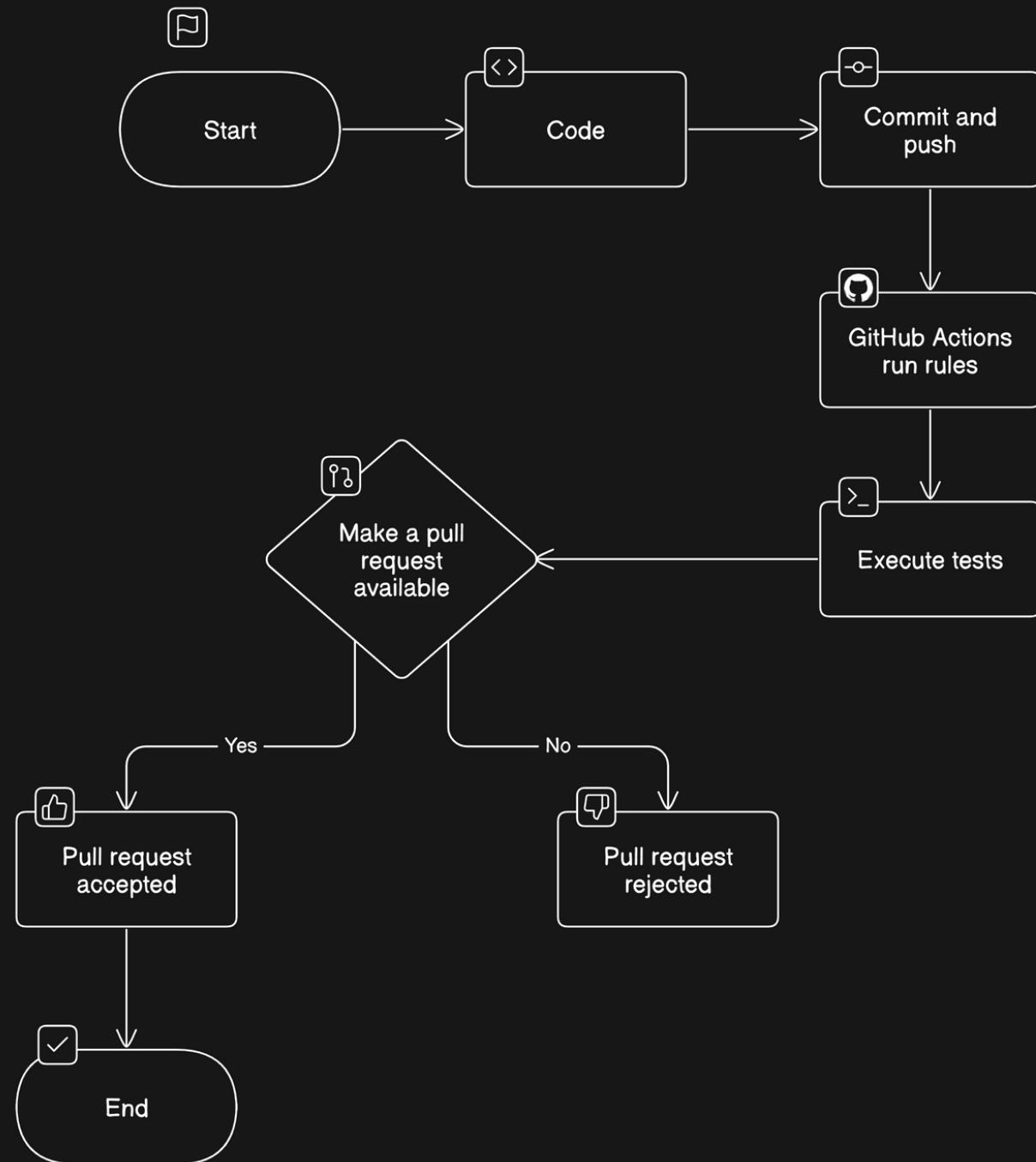
No requiere configurar servidores ni mantenerlos actualizados

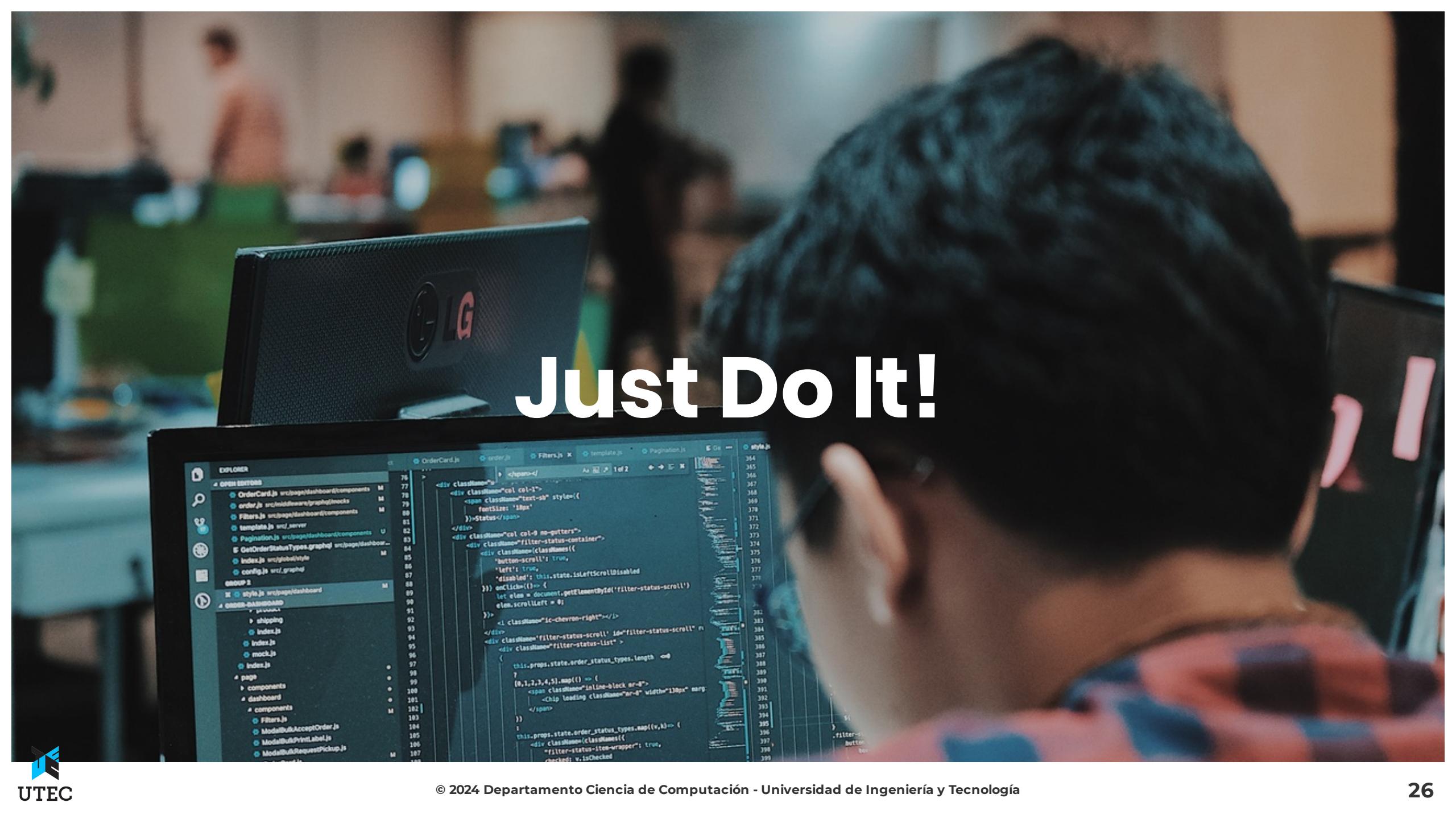


**Los flujos de trabajo se definen en archivos YAML**  
Fácil de versionar y compartir entre proyectos

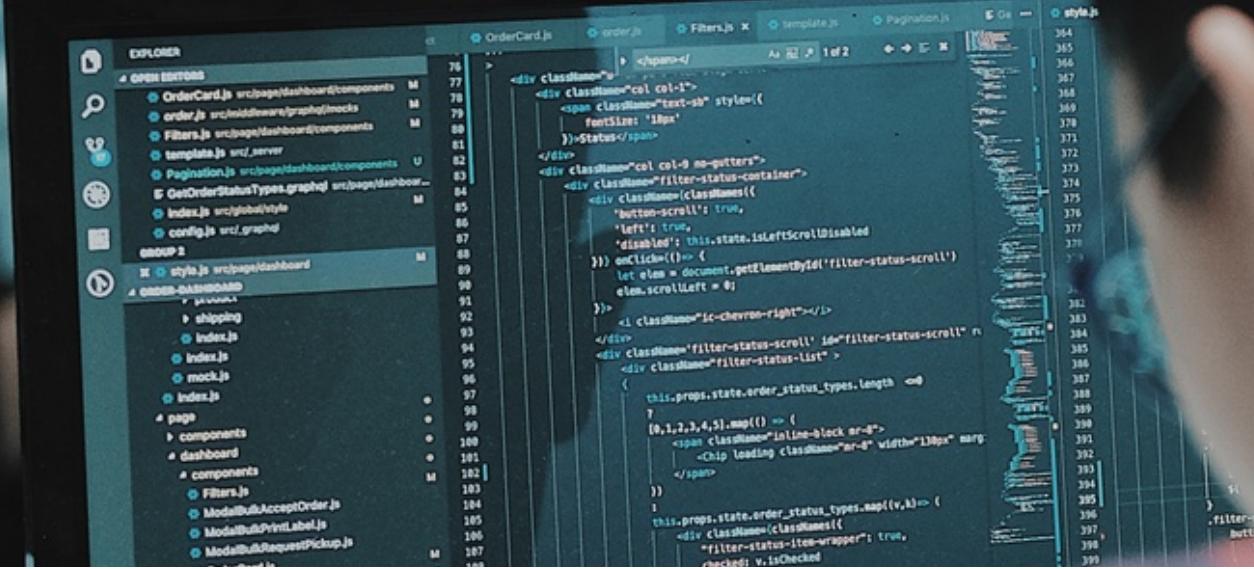
**GitHub Actions permite automatizar tareas de desarrollo de software de forma simple y escalable utilizando la infraestructura de GitHub.**

```
name: Java CI with Maven
on:
  pull_request:
    branches: [ "main" ]
jobs:
  build:
    runs-on: ubuntu-latest
    permissions:
      contents: read
    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 21
        uses: actions/setup-java@v3
        with:
          java-version: '21'
          distribution: 'temurin'
          cache: maven
      - name: Build and test with Maven
        run: mvn -B clean verify --file pom.xml
```





# Just Do It!



```
EXPLORER
  OPEN EDITORS
    OrderCard.js src/pages/dashboard/components M
    order.js src/middleware/graphQL/mock M
    Filters.js src/pages/dashboard/components M
    template.js src/server M
    Pagination.js src/pages/dashboard/components U
    GetOrderStatusTypes.graphql src/pages/dashboard M
    Index.js src/global/style M
    config.js src/graphQL M
    style.js src/pages/dashboard M
    GROUP 2
      src/pages/dashboard M
        shipping
          Index.js
          Index.js
          mock.js
          Index.js
        page
          components
            dashboard
              components
                Filters.js
                ModalBulkAcceptOrder.js
                ModalBulkPrintLabel.js
                ModalBulkRequestPickup.js
                Index.js
```

```
<div>
  <div>
    <div>
      <span></span>
    </div>
    <div>
      <div>
        <div>
          <span></span>
        </div>
        <div>
          <div>
            <div>
              <button>
                <span></span>
              </button>
              <div>
                <span></span>
              </div>
            </div>
            <div>
              <div>
                <div>
                  <div>
                    <div>
                      <div>
                        <div>
                          <div>
                            <div>
                              <div>
                                <div>
                                  <div>
                                    <div>
                                      <div>
                                        <div>
                                          <div>
                                            <div>
                                              <div>
                                                <div>
                                                  <div>
                                                    <div>
                                                      <div>
                                                        <div>
                                                          <div>
                                                            <div>
                                                              <div>
                                                                <div>
                                                                  <div>
                                                                    <div>
                                                                      <div>
                                                                        <div>
                                                                          <div>
                                                                            <div>
                                                                              <div>
                                                                                <div>
                                                                                  <div>
                                                                                    <div>
                                                                                      <div>
                                                                                        <div>
              </div>
```

# Kahoot!

**TestContainers + CI/CD with  
GitHub Actions Workflows**



# Anuncios



# ¡Premiamos tu apoyo!

Conviértete en un Debp Junior



## Objetivo

Si tienes un buen desempeño en el curso, ayuda a tus compañeros a resolver sus dudas sobre las actividades del curso



## ¿Qué hacer?

Los dos estudiantes que resuelva más dudas en el canal de #preguntas del Discord serán los nuevos Debp Juniors



## ¿Premio?

¡Así es! Los nuevos Debp Juniors ganarán una polera UTEC al final del ciclo

# Fechas importantes

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	Enunciado del Proyecto	8	9	10	11	12
Primera propuesta de proyecto	14	15	16	17	18	19
Presentación de Propuestas	21	22	23	24	25	26
Deadline: Lab 2 – E2E	28	29	30			27
		HOY				

# Nuevos Horarios de Asesorías

**Sun Apr 21**

**Mon Apr 22**

**Tue Apr 23**

**Wed Apr 24**

**3 PM – 5 PM  
(M604)**

**Thu Apr 25**

**4 PM – 6 PM  
(A502)**

**Fri Apr 26**

**11 AM – 1 PM  
(M604)**

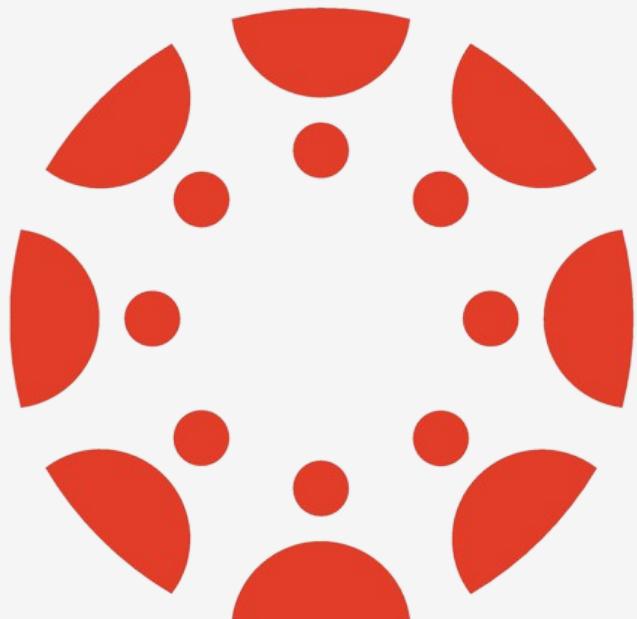
**Sat Apr 27**

**11 AM – 1PM  
(M601)**

# Quiz

**TRANSFORMATEC**

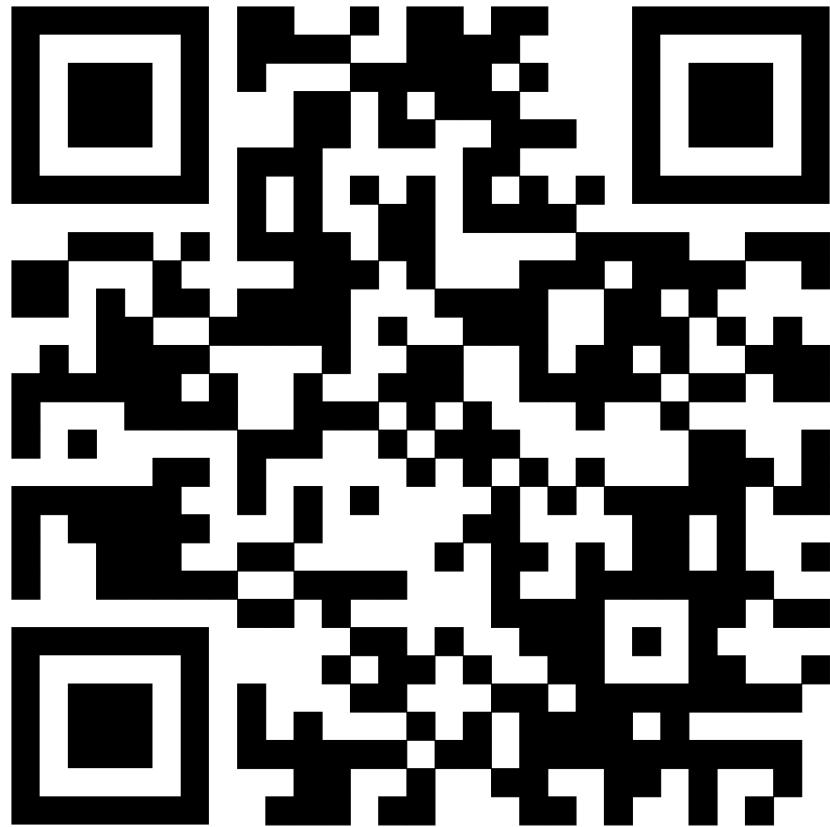
# Evaluación de Auditorio (EA)



## Quiz de Canvas

- **30 minutos**
- **1 intento**
- **Sábado 27 de abril**

# Feedback sesión de Teoría



## Encuesta

Ayúdanos a mejorar las clases de teoría



CS 2031 – DBP

# Gracias

**TRANSFORMATEC**