**ENTORNO DE DESARROLLO Y PRUEBAS**

# Definición, tipos y ventajas

Un **entorno de desarrollo y pruebas** es un espacio controlado donde los desarrolladores y testers pueden crear, modificar y evaluar aplicaciones o sistemas antes de su lanzamiento al entorno de producción. Un entorno de desarrollo y pruebas es esencial para garantizar la calidad, seguridad y estabilidad de cualquier sistema o aplicación antes de su despliegue definitivo.

**Características principales**

* **Separación del entorno de producción:** Permite experimentar y detectar errores sin afectar a los usuarios finales ni a los datos reales.
* **Simulación de condiciones reales:** Reproduce la infraestructura, configuraciones y datos similares a los del entorno de producción para asegurar que el software funcione correctamente.
* **Facilita la colaboración:** Varios miembros del equipo pueden trabajar simultáneamente en el desarrollo y la validación de nuevas funcionalidades o correcciones de errores en un entorno con las mismas características para todos.
* **Automatización de pruebas:** Permite ejecutar pruebas automáticas y manuales para verificar el funcionamiento y la calidad del software.

**Tipos de entornos**

* **Entorno de desarrollo:** Usado por los programadores para escribir y modificar código. Suele ser flexible y permite cambios rápidos.
* **Entorno de pruebas (testing):** Usado para validar que el software cumple con los requisitos y funciona correctamente. Aquí se ejecutan pruebas unitarias, de integración, de sistema y de aceptación.

**Beneficios**

* **Reducción de errores en producción:** Permite identificar y corregir fallos antes de que lleguen a los usuarios finales.
* **Mejora la calidad del software:** Al probar en un entorno similar al real, se asegura un mejor desempeño y confiabilidad.
* **Ahorro de costos:** Detectar errores en etapas tempranas es menos costoso que solucionarlos en producción.

# Herramientas comunes

En un **entorno de desarrollo y pruebas** se utilizan diversas herramientas que facilitan la programación, el control de versiones, la automatización de pruebas y la gestión de proyectos. Algunas de las más comunes y relevantes incluyen:

**Editores y entornos de desarrollo integrados (IDE):**

* **Visual Studio Code**, **Atom** y **Cloud9** son populares por su facilidad de uso, personalización y soporte para múltiples lenguajes de programación.
* **Microsoft Visual Studio** es un entorno completo que integra herramientas de desarrollo y pruebas.
* Los IDEs suelen incluir **depuradores** y **compiladores** para facilitar la identificación y corrección de errores en el código.

“Es imprescindible que todo el equipo utilice el mismo editor de códico”

**Control de versiones:**

* **Git** y plataformas como **GitHub** permiten gestionar el código fuente, colaborar en equipo y mantener un historial de cambios.

**Herramientas de testing automatizado:**

* **Selenium**: Automatiza pruebas de interfaces web.
* **JUnit** (Java), **NUnit** (.NET), **PyTest** (Python) y **RSpec** (Ruby) son frameworks para pruebas unitarias y de integración en distintos lenguajes.
* **TestRail**: Centraliza la gestión y ejecución de pruebas en equipos DevOps.

**Gestión de proyectos y seguimiento de incidencias:**

* **Jira**: Permite organizar tareas, hacer seguimiento de errores y coordinar equipos de desarrollo y testing.

**Pruebas de rendimiento y carga:**

* **Apache JMeter**, **Loader.io**, **Locust**, **nGrinder** y **Parasoft Load Test** permiten simular usuarios concurrentes y evaluar el rendimiento de aplicaciones web y sistemas.

“Estas herramientas de pruebas pueden ser sustituídas por pruebas manuales durante las primeras fases del proyecto”

Estas herramientas, combinadas, permiten crear un entorno robusto y eficiente para el desarrollo y pruebas de software, asegurando calidad, colaboración y agilidad en los proyectos.

# Implementación del entorno

**Pasos para configurar un entorno de desarrollo y pruebas adecuado**

**1. Definir los requisitos del entorno**

* Determina qué sistemas operativos, lenguajes de programación, frameworks y herramientas serán necesarios.
* Identifica las versiones específicas que deben coincidir con el entorno de producción.

**2. Preparar la infraestructura**

* Decide si usarás servidores físicos, máquinas virtuales o servicios en la nube.
* Configura servidores, redes y almacenamiento según las necesidades del proyecto.
* Asegura la separación entre los entornos de desarrollo, pruebas y producción.

**3. Instalar y configurar herramientas básicas**

* Instala editores de código, sistemas de control de versiones (como Git), y gestores de dependencias.
* Configura bases de datos, servidores web y cualquier otro software requerido.
* Prepara scripts de automatización para facilitar la configuración repetible.

**4. Gestionar el control de versiones**

* Crea repositorios para el código fuente y la documentación.
* Define flujos de trabajo (branches) para desarrollo, pruebas y producción.

**5. Configurar datos de prueba**

* Genera o anonimiza datos similares a los reales para probar las funcionalidades sin comprometer información sensible.
* Automatiza la carga y limpieza de datos de prueba.

**6. Implementar herramientas de integración y despliegue continuo (CI/CD)**

* Configura pipelines para automatizar la construcción, pruebas y despliegue del software.
* Integra herramientas para ejecutar pruebas automáticas y análisis de calidad de código.

**7. Establecer mecanismos de monitoreo y registro (logs)**

* Configura sistemas de monitoreo para detectar errores y problemas de rendimiento.
* Asegura que los logs sean accesibles para el equipo de desarrollo y pruebas.

**8. Documentar la configuración**

* Mantén una documentación clara sobre cómo instalar, configurar y utilizar el entorno.
* Incluye instrucciones para la resolución de problemas comunes.