**🔍 CONTROL DE VERSIONES**

El control de versiones permite gestionar los cambios del código fuente de forma segura y colaborativa.

**Importancia:**

1. Trabajo simultáneo sin conflictos.
2. Historial de cambios.
3. Facilita CI/CD.
4. Colaboración remota.
5. Copias de seguridad.

**Pull Request:**

Permite revisar cambios antes de fusionarlos (GitHub).

**🎨 DISEÑO DE INTERFACES (UI)**

Proceso de creación visual para facilitar la interacción usuario-sistema.

**Objetivos:**

* Experiencia visual atractiva
* Interacción intuitiva
* Mejorar accesibilidad y usabilidad

**Principios:**

* Estructura
* Simplicidad
* Visibilidad
* Flexibilidad
* Reutilización

**Accesibilidad:**

* Garantiza acceso universal
* Contraste adecuado para personas con discapacidad visual

**🛠️ AUTOMATIZACIÓN DE PRUEBAS Y CI/CD**

**Automatización de pruebas:**

Uso de herramientas/scripts para ejecutar pruebas sin intervención manual.

**Beneficios:** rapidez, detección temprana de errores, consistencia, cobertura, menor costo.

**CI/CD**

* **CI** (Integración Continua): integración frecuente y pruebas automáticas.
* **CD** (Entrega/Despliegue Continua): software listo para producción constantemente.

**Tipos de Pruebas:**

* Rendimiento
* Carga
* Estrés
* Usabilidad
* Regresión
* Compatibilidad
* Aceptación del Usuario (UAT)
* Mantenimiento

**Casos y Pruebas Ejemplares:**

1. App bancaria > UAT
2. E-commerce con alto tráfico > Carga
3. Corrección de bug > Regresión
4. Seguridad > Estrés y seguridad
5. Compatibilidad > Dispositivos móviles

**Ciclo de CI/CD:**

CI: Code, Build, Plan, Test CD: Release, Deploy, Operate, Monitor

**Desafíos:**

* Costos iniciales
* Complejidad de configuración
* Resistencia al cambio

**Microservicios:**

Ventajas: despliegue independiente, pruebas aisladas. Desafíos: integración y gestión de pipelines.

**Herramientas:**

* Pruebas: Selenium, PyTest, Cypress, JMeter, Appium
* CI/CD: Jenkins, GitHub Actions, GitLab CI, Travis CI, CircleCI

**📊 GESTIÓN DE PROYECTOS**

**¿Qué es un Proyecto?**

Esfuerzo temporal para crear un resultado único con objetivos claros y duración definida.

**Gestión:**

Planificar, programar y controlar actividades.

**Etapas Iniciales:**

* Análisis de situación actual
* Participación de usuarios

**Criterios de selección:**

1. Respaldo
2. Momento oportuno
3. Mejora de objetivos
4. Práctico
5. Rentabilidad

**Variables Clave:**

* Alcance
* Tiempo
* Costo
* Calidad
* Riesgo

**Causas de Fracaso:**

* Mala planificación
* Falta de comunicación
* Objetivos poco realistas

**Especificaciones:**

* Documento con objetivos, planificación, criterios de finalización y viabilidad

**Viabilidad:**

1. Técnica
2. Económica (costos y beneficios tangibles/intangibles)
3. Operativa

**Planeación y Control:**

* Tareas, tiempos y dependencias
* Seguimiento y motivación del equipo

**Estimación:**

* Tiempo, esfuerzo, recursos
* Uso de Gantt y PERT

**Gantt:** cronograma visual **PERT:** redes con ruta crítica, ideal con incertidumbre

**🤖 METODOLOGÍAS ÁGILES**

**Manifiesto Ágil:**

12 principios para desarrollo colaborativo, flexible y con entrega continua.

**Cuándo usar metodologías ágiles:**

* Requisitos cambiantes
* Incertidumbre
* Entregas frecuentes

**Scrum:**

* Roles: Product Owner, Scrum Master, Dev Team
* Ceremonias: Sprint Planning, Daily, Review
* Sprints: iteraciones cortas con metas claras

**Kanban:**

* Flujo continuo
* Sin roles estrictos
* Límites por columna

**Scrum vs Kanban**

|  |  |  |
| --- | --- | --- |
| **Criterio** | **Scrum** | **Kanban** |
| Enfoque | Iterativo | Flujo continuo |
| Roles | Definidos | Flexibles |
| Planificación | Por sprint | Continua |
| Cambios | No durante el sprint | Permitidos |

**📊 PRUEBAS DE SOFTWARE**

**Objetivos:**

* Verificar cumplimiento de requisitos
* Detectar errores
* Garantizar calidad

**Ciclo de vida:**

1. Planificación
2. Diseño
3. Ejecución
4. Reporte
5. Cierre

**Tipos:**

* Unitarias
* Integración
* Funcionales
* Aceptación
* Rendimiento
* Regresión

**Caja Negra vs Caja Blanca:**

* Negra: sin conocer el código (funcional)
* Blanca: con conocimiento del código (estructural)

**Herramientas:**

* Selenium, Postman, JMeter, Appium, SQL Server

**Validación Web:**

* HTML/CSS válidos
* DevTools, pruebas de red, rendimiento, accesibilidad

**Pruebas especializadas:**

* Seguridad, usabilidad, compatibilidad, volumen, documentación

**Conclusión:**

* Pruebas + métricas + automatización = mayor calidad y eficiencia