Resumen para Examen Teórico - Ingeniería de Software II

# 📦 SRS (Especificación de Requisitos de Software)

* Qué es: Documento que describe qué debe hacer el sistema.
* IEEE 830: estándar que define cómo debe estructurarse.
* Características clave: completo, correcto, consistente, no ambiguo, comprobable, trazable, priorizado, modificable.
* Estructura típica:
* - Introducción (propósito, alcance, referencias)
* - Descripción general (usuarios, funciones, entorno)
* - Requisitos no funcionales (seguridad, rendimiento, portabilidad)
* - Mantenimiento y apéndices

# 🧪 Pruebas de Software

* ✔ Caja Negra (Black-box):
* - Evalúa funcionalidades sin mirar el código.
* - Técnicas: Partición Equivalente y Análisis de Valores Límite.
* ✔ Caja Blanca (White-box):
* - Usa el código como base. Busca caminos independientes, cobertura de decisiones y bucles.
* ✔ Camino Básico (McCabe):
* - Complejidad ciclomática: V(G) = A - N + 2, o nodos predicado + 1, o regiones.
* - Grafo de flujo, caminos independientes, casos de prueba.

# 🔍 Estrategias de Prueba

* Unidad: módulo aislado, con controladores y resguardos.
* Integración: ascendente / descendente / sándwich + pruebas de regresión.
* Validación: cumplimiento de requisitos.
* Sistema: prueba completa.
* Aceptación: Alfa (entorno controlado), Beta (real, sin devs).

# 🧰 Auditoría Informática

* Qué es: evaluación crítica para garantizar seguridad, eficacia y cumplimiento.
* Tipos: interna y externa.
* Etapas: planificación, recopilación de datos, evaluación, informe, seguimiento.
* Herramientas: Nessus, Wireshark, Sqlmap, etc.
* Metodologías: OCTAVE y MAGERIT.
* Principios clave: independencia, integridad, legalidad, secreto profesional, etc.

# 🛠 Mantenimiento de Software

* Tipos: Correctivo, Adaptativo, Perfectivo, Preventivo.
* Problemas comunes: falta de docu, código viejo, baja modularidad.
* Rejuvenecimiento: Re-documentación, Re-estructuración, Ingeniería inversa, Reingeniería.
* Ley de Lehman: el software tiende al desorden sin mantenimiento.

# 🧱 GCS (Gestión de Configuración del Software)

* Controla versiones, cambios y estados.
* Elementos clave: identificación, control de versiones, control de cambios, auditoría, reportes.
* Línea base: versión formalmente aprobada, solo modificable por proceso oficial.

# ⚠️ Gestión de Riesgos

* Proceso: Identificar → Analizar (probabilidad × impacto) → Planificar → Supervisar.
* Tipos: conocidos, predecibles, impredecibles.
* Línea de corte: define qué riesgos son críticos.

# 🧮 Métricas y Estimaciones

* Productividad = KLDC / persona-mes.
* Calidad = errores / KLDC.
* Costo = $ / KLDC.
* Estimaciones: juicio experto, Delphi, Planning Poker.
* Modelos: COCOMO I y II.

# 🎨 Diseño de Interfaz y UX

* UI: interfaz visible. UX: experiencia del usuario.
* Reglas de oro (Pressman + Nielsen): diálogo simple, feedback, prevención de errores, consistencia.
* Usabilidad: fácil de usar, eficiente, aprendible.

# 🏗 Diseño Arquitectónico

* Estilos: Repositorio, Cliente-servidor, Capas.
* Distribuidos: C-S, P2P, SOA.
* Impacta en: rendimiento, seguridad, mantenibilidad.

# 🧠 Diseño de Software

* Conceptos: Abstracción, Modularidad, Ocultamiento de info.
* Alta cohesión + Bajo acoplamiento = independencia funcional.
* Diagramas: Gantt, PERT/CPM.