INGENIERÍA DE SOFTWARE II

¡Bienvenidos, 706!

Dr. Eric Melecio Castro Leal

Licenciatura en Informática

Agenda de Hoy

1. **Recordando Ingeniería de Software I:** Un repaso de los conceptos fundamentales que forman la base de nuestro curso.
2. **Introducción a Ingeniería de Software II:** ¿Qué sigue ahora?
3. **Unidad 1: Implementación:** El primer paso para materializar el software.

Recordando Ingeniería de Software I

El objetivo principal fue: *Aplicar los métodos y técnicas existentes para la obtención de requerimientos, análisis y diseño de un proyecto de software.*

1. Planificación/Análisis de Requisitos

- **Inicio:** El diagrama comienza con la identificación de una necesidad o problema que el software debe resolver.
- **Análisis de Requisitos:** Esta etapa implica la recopilación y el análisis de los requisitos del cliente y las partes interesadas. Se definen las funcionalidades, el rendimiento, la seguridad y otras características importantes del software.
- **Documentación de Requisitos:** Los requisitos se documentan en un documento de especificación de requisitos de software (SRS).
- **Decisión:** ¿Los requisitos son claros y completos? Si no, se regresa al análisis de requisitos. Si sí, se avanza a la siguiente etapa.

2. Diseño

- **Diseño de la Arquitectura:** Se define la arquitectura general del software, incluyendo los módulos, las interfaces y las tecnologías a utilizar.
- **Diseño Detallado:** Se diseñan los componentes individuales del software, incluyendo las clases, las funciones y las bases de datos.
- **Documentación del Diseño:** El diseño se documenta en un documento de diseño de software (SDD).
- **Revisión del Diseño:** Se revisa el diseño para identificar posibles problemas o mejoras.
- **Decisión:** ¿El diseño es adecuado y factible? Si no, se regresa al diseño de la arquitectura o al diseño detallado. Si sí, se avanza a la siguiente etapa.

3. Implementación/Codificación

- **Codificación:** Se escribe el código del software según el diseño.
- **Pruebas Unitarias:** Se prueban los componentes individuales del software para verificar que funcionan correctamente.
- **Integración:** Se integran los diferentes componentes del software.
- **Decisión:** ¿El código cumple con los estándares y funciona correctamente? Si no, se regresa a la codificación. Si sí, se avanza a la siguiente etapa.

4. Pruebas

- **Pruebas de Integración:** Se prueban los componentes integrados para verificar que funcionan correctamente juntos.
- **Pruebas del Sistema:** Se prueba el sistema completo para verificar que cumple con los requisitos.
- **Pruebas de Aceptación:** El cliente o usuario final prueba el software para verificar que cumple con sus expectativas.
- **Registro de Defectos:** Se registran todos los defectos encontrados durante las pruebas.
- **Decisión:** ¿El software cumple con los requisitos y las expectativas del cliente? Si no, se regresa a la implementación/codificación para corregir los defectos. Si sí, se avanza a la siguiente etapa.

5. Despliegue/Implementación

- **Despliegue:** Se implementa el software en el entorno de producción.
- **Configuración:** Se configura el software para que funcione correctamente en el entorno de producción.
- **Migración de Datos:** Se migran los datos del sistema anterior al nuevo sistema.
- **Decisión:** ¿El despliegue fue exitoso y el software funciona correctamente en el entorno de producción? Si no, se regresa a la implementación/codificación o a las pruebas para corregir los problemas. Si sí, se avanza a la siguiente etapa.

7. Obsolescencia

- **Fin de Soporte:** Se anuncia el fin del soporte para el software.
- **Retiro:** Se retira el software del uso.
- **Fin:** El diagrama termina con la obsolescencia del software.

¿Qué exploramos en Ing. de Software I?

Fundamentos de la Ingeniería de Software:

Definiciones, evolución y factores de calidad (Norma ISO 25000).

El proceso de software y sus etapas.

Modelos de Procesos de Software:

Tradicionales: Cascada y Modelo en V.

Incrementales: Modelo Incremental y Desarrollo Rápido de Aplicaciones (DRA).

Evolutivos: Prototipos y Espiral.

Ágiles: Principios, Scrum, XP.

Ingeniería de Requerimientos:

La base de todo proyecto: ¿Qué necesita el usuario?

Obtención, análisis, especificación (Casos de Uso) y validación de requerimientos.

Análisis y Diseño:

Análisis: Modelado del problema (Diagramas de Flujo de Datos, Diagrama de Clases del Dominio).

Diseño: Creación de la solución (Arquitectura MVC, Diseño de Componentes, Diseño de UI).

Iniciando Ingeniería de Software II

Dejamos atrás el "qué" y el "cómo se verá" para enfocarnos en el **"cómo se construirá"**.

Pasamos de los planos y diagramas a la selección de herramientas y la escritura de código.

UNIDAD 1: IMPLEMENTACIÓN

El puente entre el diseño y el producto final.

1.1. Selección del Entorno de Desarrollo

La primera gran decisión técnica. El entorno de desarrollo no es solo un editor de texto; es el conjunto de herramientas que nos permitirá construir, probar y desplegar el software de manera eficiente.

Una mala elección en esta etapa puede generar retrabajos, problemas de rendimiento y mayores costos.

UNIDAD 1: IMPLEMENTACIÓN

1.1.1. Evaluación y Selección de Plataforma, Lenguaje y Entorno

A. Plataforma de Ejecución: ¿Dónde vivirá nuestro software?

- **Criterios de evaluación:**
  - **Sistema Operativo:** ¿Windows, macOS, Linux, o será multiplataforma?
  - **Hardware:** ¿Se ejecutará en servidores de alto rendimiento, computadoras de escritorio, o dispositivos móviles con recursos limitados?
  - **Arquitectura:** ¿Será una aplicación de escritorio, una aplicación web, un servicio en la nube (Cloud-Native), o una app móvil?
  - **Costo y Licenciamiento:** ¿Plataformas open-source o propietarias?

UNIDAD 1: IMPLEMENTACIÓN

C. Entorno de Programación (IDE): Nuestro taller de trabajo.

- **Criterios de evaluación:**
  - **Soporte del Lenguaje:** Autocompletado, resaltado de sintaxis y análisis de código estático.
  - **Herramientas de Depuración (Debugger):** Esencial para encontrar y corregir errores.
  - **Integración con Control de Versiones:** Conexión nativa con Git/GitHub.
  - **Gestión de Dependencias y Construcción:** Facilidades para manejar librerías (npm, Maven, Pip).
  - **Extensibilidad:** ¿Se puede personalizar con plugins?
  - **Ejemplos populares:** Visual Studio Code, IntelliJ IDEA, Eclipse, PyCharm.

UNIDAD 1: IMPLEMENTACIÓN

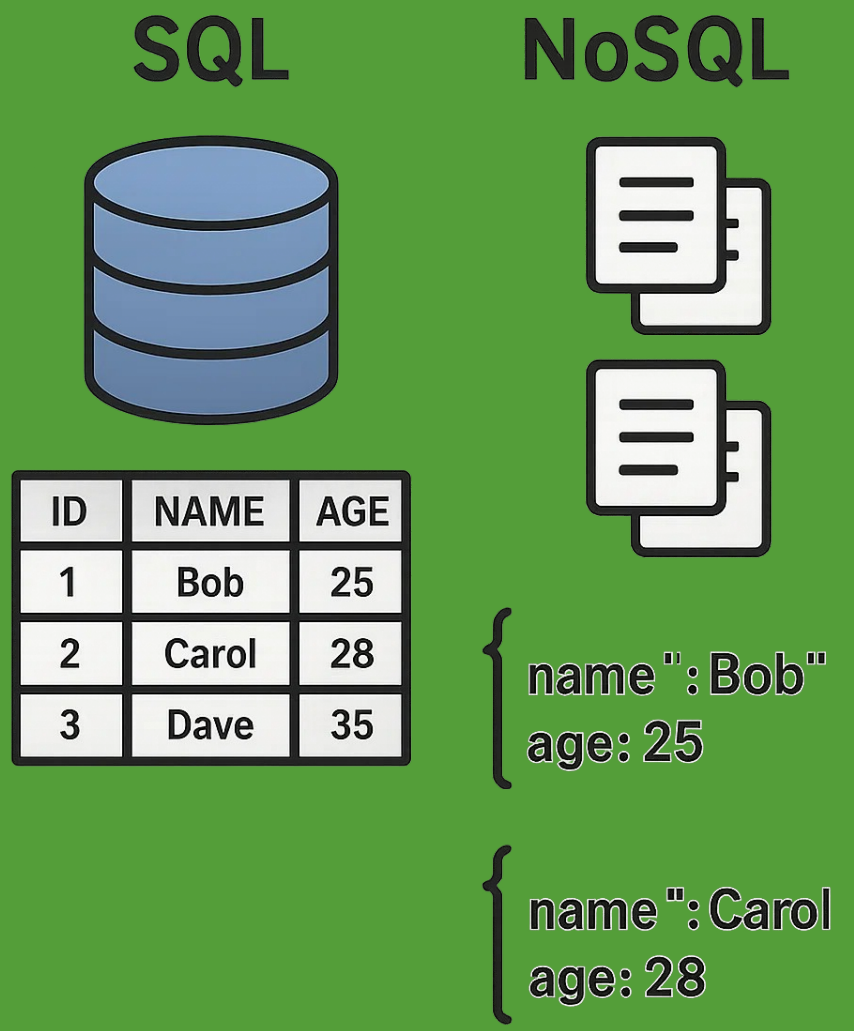
B. Lenguaje de Programación: ¿En qué idioma hablaremos con la máquina?

- **Criterios de evaluación:**
  - **Naturaleza del Proyecto:** No es lo mismo un sistema web (JavaScript, Python, Ruby) que una aplicación de análisis de datos (Python, R) o un sistema embebido (C, C++).
  - **Rendimiento:** ¿Se requieren operaciones de alta velocidad? (C++, Go, Rust).
  - **Ecosistema:** ¿Existen librerías, frameworks y herramientas que faciliten el desarrollo? (Ej. Django para Python, .NET para C#).
  - **Curva de Aprendizaje y Equipo:** ¿El equipo ya domina el lenguaje o se requiere capacitación?
  - **Comunidad y Soporte:** Una comunidad activa garantiza ayuda y evolución del lenguaje.

UNIDAD 1: IMPLEMENTACIÓN

1.1.2. Evaluación y Selección del Manejador de Bases de Datos

El corazón donde residen los datos de nuestra aplicación. La elección depende de la naturaleza y la estructura de la información que manejaremos.



UNIDAD 1: IMPLEMENTACIÓN

1. ¿Relacional (SQL) o No Relacional (NoSQL)?

SQL (Structured Query Language):

Estructura: Datos tabulares, altamente estructurados y con relaciones predefinidas (Ej. `Usuarios`, `Pedidos`).

Garantías: ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Transacciones seguras.

Ideal para: Sistemas donde la consistencia de los datos es crítica (sistemas financieros, ERPs, sistemas de inventario).

Ejemplos: MySQL, PostgreSQL, SQL Server, Oracle.

UNIDAD 1: IMPLEMENTACIÓN

NoSQL (Not Only SQL):

Estructura: Flexible. Documentos (JSON), clave-valor, grafos, columnas anchas.

Garantías: BASE (Basically Available, Soft state, Eventually consistent). Prioriza disponibilidad y escalabilidad.

Ideal para: Grandes volúmenes de datos no estructurados, alta velocidad de escritura/lectura, escalabilidad horizontal (Big Data, redes sociales, IoT).

Ejemplos: MongoDB, Redis, Cassandra, Neo4j.

UNIDAD 1: IMPLEMENTACIÓN

NoSQL (Not Only SQL):

Estructura: Flexible. Documentos (JSON), clave-valor, grafos, columnas anchas.

Garantías: BASE (Basically Available, Soft state, Eventually consistent). Prioriza disponibilidad y escalabilidad.

Ideal para: Grandes volúmenes de datos no estructurados, alta velocidad de escritura/lectura, escalabilidad horizontal (Big Data, redes sociales, IoT).

Ejemplos: MongoDB, Redis, Cassandra, Neo4j.

¿PREGUNTAS?