Explorando la Web: De la Arquitectura Cliente-Servidor a GitHub

Ingeniería | Ingeniería de sistemas | Aprendizaje Invertido

Descripción

Este plan de clase está diseñado para estudiantes universitarios de Ingeniería de Sistemas, con el propósito de que comprendan los fundamentos esenciales de Internet y el desarrollo web. A través de un enfoque de aprendizaje invertido, los estudiantes explorarán conceptos clave como la arquitectura cliente-servidor, el funcionamiento de protocolos HTTP/HTTPS, el rol de navegadores y servidores web, y la importancia de dominios, hosting y DNS. Además, se introducirán en las diferencias y funciones del frontend, backend y desarrolladores full stack, así como en las herramientas y flujos de trabajo modernos para desarrollo web, incluyendo Git y GitHub.

Este conocimiento es fundamental para cualquier profesional en ingeniería de sistemas, ya que permite entender cómo funcionan las aplicaciones web que usamos diariamente y cómo se construyen. La comprensión de estas bases facilita la colaboración en proyectos reales, mejora la capacidad para solucionar problemas y potencia la innovación tecnológica. Además, al conectar lo aprendido con prácticas reales y herramientas de la industria, los estudiantes podrán aplicar directamente estos conceptos en sus futuros proyectos profesionales.

Objetivos de Aprendizaje

- Analizar la arquitectura cliente-servidor y explicar el funcionamiento de HTTP/HTTPS en la comunicación web.
- Describir el rol de navegadores, servidores web, dominios, hosting y DNS en la infraestructura de Internet.
- Diferenciar entre frontend, backend y full stack, y reconocer las herramientas principales para el desarrollo web.
- Aplicar comandos básicos de Git y gestionar repositorios en GitHub para control de versiones colaborativo.
- Integrar conceptos teóricos con actividades prácticas para fortalecer competencias en desarrollo web.

Recursos Necesarios

- Computadoras o laptops con conexión a Internet (1 por estudiante o pareja)
- Proyector y pantalla para presentaciones
- Acceso a plataformas de video para recursos previos (YouTube, Vimeo)
- Repositorio con materiales de estudio (videos explicativos, lecturas breves) compartido en plataforma educativa
- Editor de código (Visual Studio Code recomendado)
- Cuenta gratuita de GitHub para cada estudiante
- Terminal o consola para comandos Git
- Material impreso con guías rápidas de comandos Git y conceptos clave

- Aplicación para creación de mapas mentales o pizarras colaborativas (Miro, Jamboard o similar)

Requisitos Previos

- Conocimientos básicos de informática e Internet
- Nociones elementales de programación (variables, estructuras básicas)
- Familiaridad previa con navegadores web y uso general de computadoras
- Habilidades básicas para buscar y analizar información en línea

Actividades

Sesión 1: Fundamentos de Internet y Arquitectura Web

Fase de Inicio

Tiempo estimado: 10 minutos

Propósito de la sesión:

Docente: Explica que en esta sesión se consolidarán bases teóricas fundamentales para entender cómo funciona Internet y la web, y cómo se estructura la comunicación entre cliente y servidor, preparando el terreno para el desarrollo web.

Activación de conocimientos previos:

- **Docente:** Pregunta inicial: "¿Alguna vez se preguntaron qué sucede cuando escriben una dirección web y presionan Enter? Describan en dos minutos lo que creen que ocurre detrás de escena."
- **Estudiantes:** Responden en voz alta o escriben en chat o pizarra digital sus ideas breves.

Motivación y enganche:

- **Docente:** Presenta una estadística impactante: "Actualmente, más del 60% de la población mundial usa Internet y cada segundo se realizan miles de peticiones HTTP que hacen posible que accedamos a sitios web y aplicaciones. Comprender esta tecnología es clave para cualquier ingeniero de sistemas."
- Presenta un video corto (3 minutos) que muestra visualmente la comunicación cliente-servidor y el flujo de HTTP.

Contextualización:

Docente: Conecta la temática con la vida diaria de los estudiantes: "Desde redes sociales hasta plataformas educativas y banca en línea, todo depende de estos conceptos que aprenderemos. Esto les permitirá entender mejor las tecnologías que usan y desarrollar soluciones innovadoras en su futuro profesional."

Fase de Desarrollo

Tiempo estimado: 100 minutos

Presentación del contenido:

Docente: Recuerda que los estudiantes ya revisaron previamente videos y lecturas sobre conceptos básicos. Se inicia con una breve aclaración y preguntas para resolver dudas, evitando exposición magistral.

Actividad 1: Mapas conceptuales colaborativos

- **Objetivo:** Analizar y sintetizar la arquitectura cliente-servidor y protocolos HTTP/HTTPS.
- **Instrucciones:**
  - Dividir la clase en grupos de 3-4 estudiantes.
  - En una herramienta colaborativa (Miro o Jamboard), cada grupo crea un mapa conceptual que incluya: cliente, servidor, protocolos HTTP y HTTPS, navegador y servidor web.
  - Indicar que incluyan ejemplos y relaciones entre conceptos.
  - Docente supervisa, realiza preguntas guía: "¿Por qué es importante HTTPS? ¿Qué diferencia hay entre navegador y servidor?"
- **Organización:** Grupos de 3-4 estudiantes
- **Producto:** Mapa conceptual digital compartido con toda la clase
- **Tiempo:** 40 minutos
- **Rol del docente:** Facilitar, guiar con preguntas, apoyar y corregir malentendidos.

Transición:

Docente: "Ahora que entendemos la base técnica, vamos a explorar cómo se estructuran los sitios web y qué roles desempeñan los desarrolladores en cada capa."

Actividad 2: Debate guiado sobre Frontend, Backend y Full Stack

- **Objetivo:** Diferenciar roles en el desarrollo web y herramientas asociadas.
- **Instrucciones:**
  - Presentar brevemente definiciones y ejemplos de frontend, backend y full stack.
  - Dividir la clase en tres grupos, cada uno defiende la importancia de uno de los roles.
  - Cada grupo prepara argumentos en 10 minutos.
  - Realizar debate de 20 minutos con participación guiada del docente.
  - Finalizar con resumen colaborativo en pizarra digital.
- **Organización:** Grupos de 4-5 estudiantes
- **Producto:** Síntesis escrita y debate oral
- **Tiempo:** 40 minutos
- **Rol del docente:** Moderar, reforzar conceptos, estimular participación.

Diferenciación:

- Para estudiantes que terminan antes: Proponer que investiguen ejemplos reales de herramientas frontend y backend usadas en la industria y compartan brevemente.
- Para estudiantes con dificultades: Asignar roles específicos en el debate con apoyo para el desarrollo de argumentos y recursos simplificados.

Fase de Cierre

Tiempo estimado: 10 minutos

Síntesis:

Docente: Solicita a cada estudiante escribir en una nota digital o física "Las 3 cosas más importantes que aprendí hoy".

Reflexión metacognitiva:

- ¿Cómo me ayudó entender la arquitectura cliente-servidor a comprender mejor el uso de Internet?
- ¿Qué diferencias fundamentales encontré entre frontend y backend?
- ¿Cómo puedo aplicar estos conceptos en un proyecto real de desarrollo web?

Retroalimentación:

Docente: Lee algunas respuestas en voz alta, ofrece retroalimentación positiva y aclara dudas surgidas.

Transferencia y tarea:

Docente: Explica que en la próxima sesión se profundizará en herramientas para desarrollo web y control de versiones con Git y GitHub. Asigna como tarea configurar una cuenta de GitHub y revisar videos adicionales proporcionados.

Sesión 2: Herramientas y Control de Versiones en Desarrollo Web

Fase de Inicio

Tiempo estimado: 10 minutos

Propósito de la sesión:

Docente: Explica que en esta sesión se aprenderá a aplicar herramientas esenciales para el desarrollo web, incluyendo la gestión de código con Git y GitHub, conectando con los conceptos teóricos previos.

Activación de conocimientos previos:

- **Docente:** Realiza una encuesta rápida: "¿Quién creó ya su cuenta de GitHub? ¿Qué dificultades encontraron?"
- **Estudiantes:** Participan y comentan brevemente.

Motivación y enganche:

- **Docente:** Muestra un ejemplo real de historial de versiones en GitHub de un proyecto popular, explicando cómo facilita colaboración y control de cambios.

Contextualización:

Docente: Destaca la importancia de estas herramientas para el trabajo colaborativo en proyectos de ingeniería y desarrollo de software moderno.

Fase de Desarrollo

Tiempo estimado: 100 minutos

Presentación del contenido:

Docente: Breve repaso de comandos básicos de Git y flujo de trabajo en GitHub, con enfoque en la práctica.

Actividad 1: Taller práctico de comandos Git

- **Objetivo:** Aplicar comandos básicos de Git para controlar versiones.
- **Instrucciones:**
  - El docente presenta paso a paso comandos: git init, git add, git commit, git status, git log.
  - Los estudiantes replican en sus computadoras con un proyecto de prueba.
  - Docente ofrece apoyo individual y responde dudas.
  - Ejercicio adicional: crear un archivo README y realizar commits con mensajes descriptivos.
- **Organización:** Individual
- **Producto:** Repositorio local con historial de commits
- **Tiempo:** 45 minutos
- **Rol del docente:** Supervisar, asistir, corregir errores técnicos.

Actividad 2: Publicar y colaborar en GitHub

- **Objetivo:** Subir un proyecto a GitHub y gestionar colaboraciones básicas.
- **Instrucciones:**
  - Los estudiantes crean un repositorio remoto en GitHub.
  - Conectan su repositorio local al remoto usando git remote.
  - Realizan push de sus commits.
  - Se simula colaboración: los estudiantes forman parejas, clonan el repositorio de su compañero, hacen cambios y crean pull requests.
  - Docente guía la revisión y aceptación de pull requests.
- **Organización:** Parejas

- **Producto:** Repositorios en GitHub con colaboraciones y pull requests
- **Tiempo:** 45 minutos
- **Rol del docente:** Guiar, resolver problemas, asegurar comprensión del flujo de trabajo colaborativo.

Diferenciación:

- Para estudiantes avanzados: Explorar ramas (branches) y resolución básica de conflictos.
- Para estudiantes que requieren apoyo: Sesión adicional corta con tutoría personalizada en comandos Git básicos.

Transición:

Docente: "Con estas herramientas listas, están preparados para iniciar cualquier proyecto web con buenas prácticas de desarrollo y colaboración."

Fase de Cierre

Tiempo estimado: 10 minutos

Síntesis:

Docente: Solicita a los estudiantes escribir en un chat o pizarra digital: "Un paso que me pareció más útil para el control de versiones y por qué".

Reflexión metacognitiva:

- ¿Cómo el uso de Git y GitHub mejora el trabajo en equipo en proyectos de software?
- ¿Qué dificultades encontré y cómo las resolví durante el taller práctico?
- ¿De qué manera puedo aplicar estas herramientas en futuros proyectos personales o académicos?

Retroalimentación:

Docente: Comenta respuestas seleccionadas, ofrece consejos para seguir practicando y recursos adicionales.

Transferencia y tarea:

Docente: Propone como reto crear un mini proyecto web integrando frontend y backend básico, gestionado con Git y subido a GitHub, para presentar en la siguiente unidad.

Evaluación

Tipo de evaluación:

- **Diagnóstica:** Activación de conocimientos en inicio de la Sesión 1 para conocer ideas previas.
- **Formativa:** Durante las actividades prácticas de creación de mapas conceptuales, debate, taller Git y colaboración en GitHub, con observación directa y retroalimentación continua.

- **Sumativa:** Evaluación final basada en los productos entregados: mapa conceptual, síntesis del debate, repositorios de Git y GitHub con historial y colaboraciones.

Criterios de evaluación:

- Capacidad para explicar la arquitectura cliente-servidor y protocolos HTTP/HTTPS (Objetivo 1).
- Claridad al diferenciar roles frontend, backend y full stack y sus herramientas (Objetivo 3).
- Habilidad para aplicar comandos básicos de Git y gestionar repositorios en GitHub (Objetivo 4).
- Participación activa y reflexión crítica durante debates y actividades colaborativas (Objetivo 5).

Instrumentos sugeridos:

- Rúbrica para evaluar mapas conceptuales y síntesis de debate.
- Lista de cotejo para seguimiento de comandos Git realizados correctamente.
- Observación directa y registro anecdótico durante actividades prácticas.
- Autoevaluación escrita al final de cada sesión.

Evidencias de aprendizaje:

- Mapas conceptuales digitales que reflejen comprensión de conceptos.
- Registro escrito o digital del debate con argumentos fundamentados.
- Repositorios locales y remotos con historial de commits y colaboraciones visibles.
- Respuestas y reflexiones metacognitivas que evidencien integración de conocimientos.