UAX Universidad Alfonso X el Sabio

Unidad Didáctica 1-6: Proyecto Full Stack

Ingeniería Web 1

1. Objetivo

El objetivo de este proyecto es que los estudiantes desarrollen una aplicación web sencilla pero funcional, aplicando las tecnologías y herramientas presentadas durante el curso: HTML, CSS, JavaScript, PHP, MySQL y Node.js (opcional). El proyecto está diseñado para alinearse con el contenido de cada unidad didáctica y clase, permitiendo que los estudiantes avancen de manera progresiva y organizada.

2. Proyecto: Creación de una Página Web Sencilla

A continuación, se presenta un plan detallado que especifica lo que los estudiantes deben hacer en cada clase. Esto les ayudará a gestionar su tiempo y asegurarse de que avanzan adecuadamente en el proyecto.

Unidad Didáctica 1 (Clases 1-2): Introducción y Contexto

Clase 1:

- **Tareas para los estudiantes:**
    - Reflexionar sobre posibles ideas para el proyecto web que desarrollarán.

Clase 2:

- **Tareas para los estudiantes:**
    - **Pensar y definir la idea del proyecto web que desarrollarán durante el curso.** Deberán tener claro qué tipo de aplicación desean crear (por ejemplo, blog personal, sistema de registro de usuarios, lista de tareas, etc.).

Unidad Didáctica 2 (Clases 3-4): Entornos de Desarrollo, HTML y CSS

Clase 3: Instalación y Configuración del Entorno

- **Objetivos:**
    - Instalar y configurar el entorno de desarrollo.
    - Conocer las herramientas que utilizarán durante el curso.
- **Actividades:**
    - **Instalación de herramientas:**
        - **Visual Studio Code (VSC):** Instalar VSC y añade los ficheros con sus extensiones para HTML, CSS, JavaScript y PHP. Download
            - Extensión **Live Server**: Para ver los cambios en tiempo real en el navegador.
        - **XAMPP:** Instalar XAMPP siguiendo el video tutorial.
        - **Configuración de XAMPP:** Instalar XAMPP en una ubicación con permisos (por ejemplo, C:\xampp).
        - **Activar Apache y MySQL:** Iniciar los servicios desde el Panel de Control de XAMPP.

        - **Git y GitHub**: Instalar Git y configurar GitHub siguiendo la guía detallada en el archivo `**Instalación Git Github.md**`.
    - **Configuración del proyecto:**
        - Crear la carpeta del proyecto en C:\xampp\htdocs\mi_proyecto.
        - Crear los archivos iniciales:
            - Crea un archivo **index.html** para la estructura principal de tu sitio web.
            - Crea un archivo **style.css** para los estilos y diseño de tu sitio web.
            - Crea un archivo **script.js** para añadir la lógica y funcionalidad de JavaScript.
            - Crea un archivo **index.php** si tu proyecto utiliza PHP para la lógica del lado del servidor.
- **Tareas para los estudiantes:**
    - Completar la instalación y configuración de todas las herramientas.
    - Asegurarse de que pueden acceder al curso del proyecto: http://localhost/mi_proyecto/IngenieriaWeb/index.html
        - Para acceder al curso del proyecto que he realizado tenéis que ir a mi repositorio **IngenieriaWeb**.
        - **Repositorio**: https://github.com/CanaletaFast/IngenieriaWeb
        - Para clonar el proyecto del repositorio a local debéis clonarlo:
            - **git clone** https://github.com/CanaletaFast/IngenieriaWeb.git
    - **Definir claramente el objetivo de su proyecto web**, detallando qué funcionalidad tendrá y qué tecnologías utilizarán.

Clase 4: Construcción de la Estructura Básica con HTML y CSS

- **Objetivos:**
    - Aprender la sintaxis básica de HTML y CSS.
    - Crear la estructura básica de la página web.
- **Actividades:**
    - **HTML:**
        - Aprender las etiquetas básicas de HTML5.

        - Creación de la estructura semántica: <header>, <nav>, <main>, <footer>.
        - **Aplicación en el proyecto:** Los estudiantes deben crear la estructura básica de su página según el proyecto que han definido.
    - **CSS:**
        - Introducción a CSS y cómo enlazarlo con HTML.
        - Aplicación de estilos básicos: colores, fuentes, márgenes y padding.
        - **Diseño responsivo básico:** Introducción a media queries para adaptar la página a diferentes tamaños de pantalla.
- **Tareas para los estudiantes:**
    - Completar la estructura HTML de su página principal.
    - Aplicar estilos con CSS para mejorar la apariencia visual.
    - **Entregar (GITHUB) un avance del proyecto** que incluya la estructura básica y estilos iniciales.

Unidad Didáctica 3 (Clases 5-6): JavaScript y el DOM

Clase 5: Introducción a JavaScript y el DOM

- **Objetivos:**
    - Entender la importancia de JavaScript en el desarrollo web.
    - Aprender a manipular el DOM con JavaScript.
- **Actividades:**
    - **JavaScript Básico:**
        - Sintaxis básica: variables, tipos de datos, operadores, estructuras de control.
        - Cómo enlazar un archivo script.js en HTML.
    - **DOM:**
        - Entender el Modelo de Objetos del Documento.
        - Selección y manipulación de elementos del DOM.
    - **Aplicación en el proyecto:**
        - Añadir interactividad básica, como mostrar un mensaje de bienvenida o cambiar estilos dinámicamente.

- **Tareas para los estudiantes:**
    - Crear el archivo script.js y enlazarlo con su HTML.
    - Implementar funciones que interactúen con el DOM en su proyecto.
    - **Entregar (GITHUB) un avance del proyecto** que incluya interactividad básica.

Clase 6: Validaciones y Eventos en JavaScript

- **Objetivos:**
    - Aprender a manejar eventos en JavaScript.
    - Implementar validaciones de formularios en el lado del cliente.
- **Actividades:**
    - **Eventos:**
        - Explicación de los eventos más comunes: click, submit, change.
        - Cómo agregar y manejar eventos en JavaScript.
    - **Validaciones:**
        - Implementar validaciones como campos obligatorios, formatos de email, contraseñas seguras.
        - Mostrar mensajes de error al usuario de manera amigable.
    - **Aplicación en el proyecto:**
        - Los estudiantes deben implementar validaciones en los formularios de su proyecto.
- **Tareas para los estudiantes:**
    - Completar las validaciones necesarias en su proyecto.
    - **Entregar (GITHUB) un avance del proyecto** con las validaciones funcionando.

Unidad Didáctica 4 (Clases 7-8): Back-End con PHP y MySQL

Clase 7: Introducción a PHP y Configuración de la Base de Datos

- **Objetivos:**
    - Aprender los fundamentos de PHP.
    - Configurar y utilizar una base de datos MySQL.
- **Actividades:**

- **PHP Básico:**
  - Sintaxis básica: variables, operadores, estructuras de control, funciones.
  - Cómo integrar PHP en archivos HTML.
- **Base de Datos:**
  - Acceso a phpMyAdmin a través de http://localhost/phpmyadmin/.
  - Creación de una base de datos (por ejemplo, mi_proyecto_db).
  - Creación de tablas necesarias para el proyecto (por ejemplo, tabla usuarios).
- **Aplicación en el proyecto:**
  - Escribir un script PHP (process.php) para manejar el envío de formularios.
  - Conectar PHP con MySQL usando mysqli o PDO.

- **Tareas para los estudiantes:**
  - Configurar la base de datos para su proyecto.
  - Implementar la conexión a la base de datos desde PHP.
  - **Entregar (GITHUB) un avance del proyecto** que demuestre la conexión exitosa a la base de datos.

Clase 8: Procesamiento de Formularios y Seguridad en PHP

- **Objetivos:**
  - Procesar datos de formularios en el lado del servidor.
  - Implementar buenas prácticas de seguridad.
- **Actividades:**
  - **Procesamiento de Formularios:**
    - Cómo recibir datos del formulario usando $_POST o $_GET.
    - Insertar datos en la base de datos con consultas SQL.
  - **Seguridad:**
    - Validación del lado del servidor: asegurarse de que los datos recibidos son válidos.

        - Protección contra inyecciones SQL utilizando prepared statements (mysqli_prepare, bind_param).
        - Sanitización de entradas con htmlspecialchars().
    - **Almacenamiento de Contraseñas:**
        - Uso de password_hash() para almacenar contraseñas de forma segura.
        - Verificación de contraseñas con password_verify().
    - **Aplicación en el proyecto:**
        - Los estudiantes deben implementar el registro de usuarios de manera segura en su proyecto.
- **Tareas para los estudiantes:**
    - Procesar los formularios y almacenar datos en la base de datos de manera segura.
    - **Entregar (GITHUB) un avance del proyecto** que incluya el registro de usuarios funcional y seguro.

Unidad Didáctica 5 (Clases 9-10): Profundización en PHP y Introducción a Node.js

Clase 9: Programación Orientada a Objetos en PHP

- **Objetivos:**
    - Introducir los conceptos de Programación Orientada a Objetos (POO) en PHP.
    - Refactorizar el código para utilizar clases y objetos.
- **Actividades:**
    - **POO en PHP:**
        - Explicación de clases, objetos, propiedades y métodos.
        - Conceptos de herencia y encapsulación.
    - **Aplicación en el proyecto:**
        - Los estudiantes deben crear clases para manejar la lógica de su aplicación (por ejemplo, una clase Usuario para manejar operaciones relacionadas con usuarios).
- **Tareas para los estudiantes:**
    - Refactorizar su código PHP para utilizar POO.

- **Entregar (GITHUB) un avance del proyecto** que demuestre el uso de POO.

Clase 10: Introducción a Node.js y su Aplicación en el Proyecto

- **Objetivos:**
  - Conocer Node.js como entorno de ejecución de JavaScript en el lado del servidor.
  - Comprender cuándo y por qué usar Node.js en lugar de PHP.
- **Actividades:**
  - **Node.js Básico:**
    - Instalación de Node.js y configuración básica.
    - Crear un servidor básico con Node.js utilizando el módulo http.
    - Explicación de npm y cómo gestionar paquetes.
  - **Aplicación en el proyecto (Opcional):**
    - **Para estudiantes interesados**, se propone volver implementar una parte del backend utilizando Node.js y Express.js.
    - Conectar Node.js con la base de datos MySQL utilizando paquetes como mysql o sequelize.
- **Tareas para los estudiantes:**
  - **Opcional:** Configurar un servidor Node.js y volver implementar una funcionalidad del proyecto.
  - **Entregar (GITHUB) un avance del proyecto** que muestre la implementación con Node.js, si es aplicable.

Unidad Didáctica 6 (Clases 11-12): Diseño Responsivo, Intercambio de Datos y AJAX

Clase 11: Diseño Responsivo Avanzado y Formatos de Intercambio de Datos

- **Objetivos:**
  - Mejorar el diseño responsivo utilizando técnicas avanzadas.
  - Aprender sobre formatos de intercambio de datos: XML, JSON y YAML.
- **Actividades:**
  - **Diseño Responsivo Avanzado:**

        - Uso de Flexbox y CSS Grid para crear diseños más complejos y adaptables.
        - Prácticas para asegurar la compatibilidad con diferentes navegadores y dispositivos.
    - **Formatos de Intercambio de Datos:**
        - Explicación de XML, JSON y YAML.
        - Ventajas y desventajas de cada formato.
        - Cómo utilizar JSON en aplicaciones web.
    - **Aplicación en el proyecto:**
        - Los estudiantes deben mejorar el diseño de su aplicación para que sea totalmente responsivo.
        - Preparar el backend para enviar y recibir datos en formato JSON.
- **Tareas para los estudiantes:**
    - Implementar mejoras en el diseño responsivo de su proyecto.
    - Adaptar el backend para trabajar con JSON.
    - **Entregar (GITHUB) un avance del proyecto** que muestre estas mejoras.

Clase 12: AJAX y Fetch API para Comunicación Asíncrona

- **Objetivos:**
    - Aprender a realizar solicitudes asíncronas al servidor sin recargar la página.
    - Implementar AJAX utilizando Fetch API.
- **Actividades:**
    - **AJAX y Fetch API:**
        - Explicación de cómo funciona AJAX y su importancia en aplicaciones modernas.
        - Uso de Fetch API para realizar solicitudes GET y POST.
        - Manejo de respuestas y errores en solicitudes asíncronas.
    - **Aplicación en el proyecto:**
        - Los estudiantes deben implementar funcionalidades que utilicen Fetch API para interactuar con el servidor.

        - Por ejemplo, cargar datos dinámicamente, enviar formularios sin recargar la página, etc.

- **Tareas para los estudiantes:**
    - Implementar al menos una funcionalidad en su proyecto que utilice Fetch API.
    - **Entregar el proyecto completo** con todas las funcionalidades implementadas y documentadas.

Evaluación del Proyecto

Criterios de Evaluación:

- **Unidad Didáctica 2 (HTML y CSS):** 20%
    - Estructura HTML correcta y semántica.
    - Aplicación de estilos CSS coherentes y diseño responsivo básico.
- **Unidad Didáctica 3 (JavaScript y DOM):** 20%
    - Implementación de interactividad y validaciones en el lado del cliente.
    - Uso adecuado de eventos y manipulación del DOM.
- **Unidad Didáctica 4 (PHP y MySQL):** 25%
    - Conexión correcta a la base de datos.
    - Procesamiento seguro de formularios y almacenamiento de datos.
    - Implementación de medidas de seguridad básicas (inyección SQL, XSS, almacenamiento de contraseñas).
- **Unidad Didáctica 5 (PHP Avanzado y Node.js):** 15%
    - Uso de POO en PHP para mejorar la estructura del código.
    - **Opcional:** Implementación de funcionalidades con Node.js.
- **Unidad Didáctica 6 (Diseño Responsivo y AJAX):** 20%
    - Mejora del diseño responsivo utilizando técnicas avanzadas.
    - Implementación de comunicación asíncrona con Fetch API.

Notas y Entregas:

- **Evaluación Continua:**
    - Cada semana, los estudiantes deben subir al repositorio GITHUB la parte correspondiente a la unidad didáctica en curso. Esto ayudará a asegurar que

vayan completando el proyecto de manera progresiva, lo que influirá en el porcentaje de su nota final.

- Habrá **entregas parciales** al finalizar cada unidad didáctica para verificar el progreso y ofrecer retroalimentación.

- **Nota Final:**
  - La nota final se calculará sumando las puntuaciones obtenidas en cada unidad.
  - **Criterios adicionales:**
    - **Funcionalidad Completa:** Si el proyecto cumple con todos los requisitos y funciona correctamente, se considera un 10.
    - **Calidad del Código:** Código limpio, comentado y bien estructurado.
    - **Originalidad y Creatividad:** Implementación de funcionalidades adicionales o mejoras creativas al proyecto.

Todas las instalaciones y cursos

Recursos Adicionales para el Aprendizaje

- **HTML y CSS:**
  - W3Schools: HTML, CSS,
  - SoyDalto: Curso de HTML y CSS
  - Scrimba: Curso Interactivo de HTML y CSS
- **JavaScript:**
  - W3Schools: JavaScript
  - Scrimba: Curso Interactivo de JavaScript
- **PHP y MySQL:**
  - W3Schools: PHP, MySQL
  - PHP Manual: Documentación Oficial
    - PHP: Curso
    - MySQL: Curso
- **Node.js:**
  - W3Schools: Node.js
  - Node.js: Documentación Oficial

Instalaciones adicionales:

Visual Studio Code: Download

- Extensión **Live Server**: Para ver los cambios en tiempo real en el navegador.

XAMPP: Download

- Curso: https://www.youtube.com/watch?v=uHqlIDhQ4hc

Git y GitHub:

- Curso de Git y GitHub: YouTube
- **Git y GitHub**: Instalar Git y configurar GitHub siguiendo la guía detallada en el archivo `**Instalación Git Github.md**`.

Conclusión y Siguientes Pasos

Este plan detallado proporciona a los estudiantes una guía clara de lo que se espera de ellos en cada clase y unidad didáctica. Al alinear las actividades del proyecto con el contenido de las clases, se asegura que los estudiantes pueden aplicar inmediatamente lo que aprenden y avanzar de manera consistente.

Siguientes Pasos para los Estudiantes:

- **Documentación:**
    - Mantener una documentación del proyecto que incluya:
        - Descripción del proyecto.
        - Tecnologías utilizadas.
        - Funcionalidades implementadas.
        - Instrucciones para ejecutar el proyecto.
- **Preparación para la Presentación Final:**
    - Al finalizar el curso, los estudiantes deben estar preparados para presentar su proyecto.
    - La presentación debe incluir una demostración de la aplicación y una explicación de las decisiones de diseño y técnicas utilizadas.

Node.js (Opcional)

Clarificación sobre Node.js en el Proyecto

¿Por qué incluir Node.js?

- **Exposición a Tecnologías Modernas:**
    - Node.js es ampliamente utilizado en la industria y permite ejecutar JavaScript en el lado del servidor.
    - Ofrece una perspectiva diferente al desarrollo backend comparado con PHP.
- **Comprensión de Diferentes Entornos:**
    - Al aprender Node.js, los estudiantes pueden entender las diferencias y similitudes con PHP.
    - Esto amplía sus habilidades y conocimientos, preparándolos mejor para el mercado laboral.

¿Qué deben hacer los estudiantes con Node.js?

- **Instalación y Configuración:**
    - Instalar Node.js y familiarizarse con npm (Node Package Manager).
- **Crear un Servidor Básico:**
    - Utilizar el módulo http de Node.js para crear un servidor simple que responda a solicitudes.
- **Opcionalmente, Usar Express.js:**
    - Instalar y configurar Express.js, un framework minimalista para aplicaciones web en Node.js.
    - Reescribir algunas rutas o endpoints de su aplicación utilizando Express.js.
- **Conexión a la Base de Datos:**
    - Utilizar paquetes como mysql o sequelize para conectar Node.js a la base de datos MySQL.
    - Realizar operaciones básicas de CRUD (Crear, Leer, Actualizar, Eliminar).

Aplicación Práctica en el Proyecto:

- **Reimplementar Funcionalidades Clave:**
    - Por ejemplo, reescribir el proceso de registro o inicio de sesión utilizando Node.js.
    - Esto permitirá a los estudiantes comparar el manejo de rutas, controladores y modelos entre PHP y Node.js.

Recursos para Aprender Node.js:

- **Tutoriales y Documentación:**

- W3Schools: Node.js Tutorial
- Node.js: Documentación Oficial
- Express.js: Guía Rápida

- **Videos y Cursos:**
  - SoyDalto: Curso de Node.js
  - Scrimba: Curso Interactivo de Node.js

3. Instrucciones de entrega

- **Extensión**:
  - El proyecto deberá ser entregado en formato comprimido .zip o .rar con todo el contenido necesario (código fuente, base de datos, documentación, etc.).
- **Nombre del fichero**:
  - El archivo comprimido deberá nombrarse con el siguiente formato: **NombreApellido_ProyectoFinal_WebApp.zip** (por ejemplo, **JuanPerez_ProyectoFinal_WebApp.zip**).
- **Nombre de los documentos**:
  - Dentro del archivo comprimido deberá incluirse un documento de texto llamado **documentacion_proyecto.md** o **documentacion_proyecto.pdf**, donde se detallen:
    - Descripción del proyecto.
    - Tecnologías utilizadas.
    - Funcionalidades implementadas.
    - Instrucciones detalladas para la ejecución del proyecto.
- **Formato de entrega**:
  - La entrega debe realizarse a través del repositorio de GitHub.
  - El enlace del repositorio de GitHub debe incluirse en el documento de documentación y también ser enviado por correo electrónico al profesor antes de la fecha límite.
  - Asegurarse de que el repositorio incluya:
    - El código fuente actualizado y funcional.

- La base de datos exportada en formato .sql si utiliza MySQL (por ejemplo, mi_proyecto_db.sql).
- La documentación mencionada anteriormente.
- Un archivo README.md con instrucciones para clonar y ejecutar el proyecto localmente.

UAX Universidad Alfonso X el Sabio

G R A C I A S

UAX.COM