**ACTIVIDAD DIDÁCTICA: RETO DE DESARROLLO DE SOFTWARE**

**Contexto:**

En esta actividad, los estudiantes asumirán el rol de un equipo de desarrollo de software, donde deberán trabajar colaborativamente para implementar un pequeño proyecto utilizando un enfoque ágil. La actividad está diseñada para desarrollarse en **1 hora y 30 minutos**, y seguirá un esquema de trabajo que les permita cubrir todas las etapas de un proyecto real: análisis de requerimientos, diseño, codificación, pruebas, y entrega.

**Objetivo del Reto:**

Desarrollar una aplicación web sencilla para gestionar una lista de tareas (To-Do List) que permita al usuario agregar, eliminar y marcar tareas como completadas.

**Duración total:** 1hora y 40 minutos

**PASO A PASO DEL RETO**

**Preparación:**

* Configurar el ambiente de desarrollo con herramientas básicas como Visual Studio Code, Node.js, o cualquier framework básico que los estudiantes conozcan (React, Vue.js, etc.).
* Crear un repositorio en GitHub con la estructura básica del proyecto para que los estudiantes clonen y comiencen a trabajar rápidamente.

**Guía para los Estudiantes**

**Parte 1: Introducción al reto (10 minutos)**

1. **Objetivo:**  
   Se les plantea el reto: desarrollar una aplicación web para gestionar una lista de tareas (To-Do List).
   * La aplicación debe permitir agregar, eliminar y marcar tareas como completadas.
   * Al final, deberán entregar el código y hacer una demostración funcional del proyecto.
2. **Roles:**  
   Se asignarán roles dentro del equipo para garantizar una división de tareas eficiente:
   * **Líder del equipo**: Responsable de organizar el flujo de trabajo y la coordinación.
   * **Desarrollador Frontend**: Encargado de la interfaz de usuario.
   * **Desarrollador Backend**: Encargado de la lógica y la estructura de datos (si se utiliza backend).
   * **Tester**: Realiza las pruebas y asegura la calidad del código.

**Parte 2: Planificación y análisis de requerimientos (15 minutos)**

1. **Revisión de la funcionalidad básica**:
   * Agregar tareas (input de texto).
   * Eliminar tareas (botón de eliminación).
   * Marcar tareas como completadas (checkbox).
2. **User Stories (historias de usuario)**: Los estudiantes deben escribir al menos 3 historias de usuario sencillas como:
   * "Como usuario, quiero agregar una tarea a mi lista para organizar mis actividades."
   * "Como usuario, quiero poder marcar una tarea como completada para saber qué he terminado."
   * "Como usuario, quiero eliminar tareas de la lista para mantenerla limpia."
3. **Estimación del tiempo**:
   * Cada equipo debe estimar el tiempo que tomará implementar cada funcionalidad y asignar prioridades (por ejemplo, agregar tareas es prioritario frente a eliminar tareas).

**Parte 3: Desarrollo de la solución (45 minutos)**

1. **Configuración del entorno** (5 minutos):
   * Todos los integrantes deben clonar el repositorio preparado y ejecutar el entorno de desarrollo local.
2. **División del trabajo** (5 minutos):
   * El líder organiza a los miembros del equipo en pequeñas tareas, basadas en las historias de usuario. Cada persona o subgrupo se enfocará en una parte del proyecto (frontend, backend, pruebas).
3. **Codificación** (35 minutos):
   * **Frontend**: Diseñar la interfaz en HTML/CSS, implementar los inputs y botones necesarios.
   * **Lógica (JavaScript)**: Escribir la lógica para agregar y eliminar tareas, así como marcar tareas como completadas.
   * **Versionado en GitHub**: Cada cambio importante debe ser registrado en el repositorio de GitHub para control de versiones.
   * **Pruebas unitarias**: El tester debe escribir pruebas simples para verificar que la funcionalidad de agregar, eliminar y marcar tareas funciona correctamente.

**Parte 4: Pruebas y correcciones (15 minutos)**

1. **Pruebas funcionales** (10 minutos):
   * Cada equipo debe probar la aplicación asegurándose de que todas las historias de usuario se cumplan.
   * Probar diferentes casos: agregar varias tareas, eliminarlas, marcar algunas como completadas.
2. **Corrección de errores** (5 minutos):
   * Si se detectan fallos, el equipo debe corregirlos inmediatamente, priorizando la funcionalidad principal.

**Parte 5: Presentación del proyecto (10 minutos)**

1. **Demostración**:
   * Cada equipo debe hacer una presentación rápida de 3 minutos, mostrando el funcionamiento de su aplicación.
   * Explicar brevemente las decisiones técnicas y cómo lograron implementar cada historia de usuario.
2. **Revisión del código**:
   * El profesor puede hacer una breve revisión del código para asegurar que se sigue una estructura clara y buenas prácticas.

**Cierre y Evaluación (5 minutos)**

1. **Retroalimentación**:
   * Se resalta el trabajo en equipo, el flujo de trabajo y las soluciones técnicas propuestas.
   * Los estudiantes comparten los retos que encontraron y cómo los resolvieron.
2. **Evaluación**:
   * Se evaluará a los estudiantes en función de su capacidad para trabajar en equipo, cumplir con los requerimientos funcionales, utilizar buenas prácticas de desarrollo y la calidad del código entregado.

**Conclusión**

Esta actividad permite practicar el ciclo de vida de un proyecto de software completo en una sesión de 1.5 horas. La organización por equipos y la asignación de roles simulan un ambiente de trabajo real en una empresa de software, donde cada miembro contribuye a un objetivo común. Al final de la actividad, los estudiantes habrán experimentado tanto el desarrollo técnico como el trabajo colaborativo bajo presión de tiempo.