CODERFLEX

JavaScript
Consigna del Proyecto Final

CODERHOUSE

Simulador interactivo en JavaScript

Para finalizar el curso y poner en práctica todos los conocimientos adquiridos, te proponemos **desarrollar una aplicación web JavaScript** interactiva, que simula un proceso comercial o profesional.

Objetivos

- Desarrollar un simulador interactivo que funcione de igual forma en la cual funciona una aplicación web real, utilizando las diferentes herramientas y técnicas del lenguaje JavaScript aprendida a lo largo del curso.

  Ten presente simular aquellas funcionalidades del simulador interactivo que no tendremos disponibles por una cuestión de limitaciones técnicas.

Requisitos

Este trabajo cuenta con una única instancia:

1. Aplicación frontend basada en un simulador interactivo

1. Aplicación web basada en un simulador interactivo

El proyecto debe ser entregado mediante un link a un repositorio de Github. Si no manejas esta herramienta, puedes presentar un archivo en formato .ZIP (no se aceptan RAR).

La carpeta de tu proyecto debe tener la siguiente estructura:

- **documentos HTML**

  Puede contener un único documento HTML, o varios, de acuerdo a cómo has estructurado tu proyecto.

- **documentos CSS**

  Debe contener al menos un archivo CSS o, en su defecto, un framework CSS como Bootstrap, Tailwind, Materialize o cualquier otro que consideres necesario para desarrollar una interfaz gráfica acorde para tu proyecto.

- **Archivos JavaScript**
  - Debe contener al menos dos archivos JavaScript, referenciados en el o los documentos HTML donde desarrollas la interfaz gráfica de tu simulador.
  - Debe contener al menos un archivo en formato .JSON, el cual será tu base de datos simulada. Si utilizas un servicio de backend en nube, asegúrate de que el mismo esté abierto a ser utilizado desde cualquier URL.

- **Recursos adicionales**
  - Incluye una subcarpeta de assets con las imágenes, videos, y otros elementos multimedia necesarios para la funcionalidad de tu simulador interactivo.
  - Si tu proyecto cuenta con alguna particularidad o pre-configuración a realizar antes de probarlo, incluye un archivo readme.md en el mismo para guiar al docente corrector sobre los previos a realizar antes de probarlo.

A. Objetivos específicos

- **Uso de DOM**

  Tu aplicación JavaScript debe interactuar con HTML utilizando DOM y los eventos necesarios.

- **Arrays**

  Todos los arrays de objetos que utilices dentro de tu aplicación deben ser convertidos a un archivo en formato .JSON y accedidos mediante la tecnología Fetch.

  Lógica de tu aplicación

  El simulador interactivo debe completar todo el circuito o proceso de negocio, de acuerdo a la temática del mismo. Debes obviar temas más complejos como ser un registro de usuario y login, pero no puedes obviar armar un circuito completo de una compra online, o de la cotización de productos o servicios.

- **Uso de herramientas de terceros**

  Debes incluir al menos una librería JS externa y debes eliminar el uso de herramientas más limitadas del lenguaje, como ser los cuadros de diálogo Prompt, Confirm y Alert, además de eliminar todo rastro de mensajes en la consola JS de las Herramientas para Desarrollador (DevTools).

- **Código claro**

  Todo tu código debe ser claro y estar correctamente estructurado para su lectura. Puedes dejar comentarios breves en el código, pero no puedes dejar código en desuso/comentado que complique la lectura y análisis del mismo durante el período de corrección.

  Evita utilizar herramientas de Minificación de código para esta instancia como también incluir recursos pesados dentro del proyecto.

B. Sugerencias

Guíate por estas sugerencias más detalladas sobre el proyecto que deberás desarrollar, que te ayudarán a cumplir con las principales pautas.

Debe contener las siguientes características:

- **Uso de DOM**

  JavaScript debe estar integrado con HTML mediante DOM y Eventos. No debes mostrar productos o servicios de tu simulador generados de forma estática en HTML. Todo esto debe provenir de JavaScript, creando el HTML dinámico mediante Template Strings + Literals.

- **Interfaz visual**

  Tus documentos HTML deben integrar CSS nativo o un Framework como los mencionados al inicio de este documento. Tu aplicación debe contener una estética visual más o menos acorde, y no debe ser solamente una estructura HTML.

  Tus documentos HTML no deben contener código JS.

- **Valida y controla los errores**

  Integra las herramientas necesarias para controlar errores, como ser el uso de try - catch - finally. No muestres errores de JavaScript en la interfaz de usuario, muestra errores utilizando mensajes del estilo UX. El usuario no tiene porqué conocer los aspectos técnicos de tu aplicación.

- **Utiliza archivos JSON como base de datos/backend**

  Si bien puedes sumar servicios en nube como Mockapi o Firebase, preferimos que armes todo tu “backend” en archivos JSON. Evitas que un servicio en nube pueda estar caído al momento de corregir tu entrega, o que la misma no pueda ser accedida desde una computadora diferente a la tuya por restricciones de IP, Puerto, o cualquier otro factor técnico.

B. Sugerencias

- **Circuito completo**

  Realiza un análisis previo a crear tu proyecto, de cómo funcionan servicios similares al simulador que deseas realizar. Esto te permite enfocarte en desarrollar las funciones más importantes y pensar con tiempo cómo simular aquellas limitaciones técnicas que puedan existir para con tu simulador.

- **- complejo + dinámico**

  No pierdas tiempo realizando pantallas de registro - login - formulario para completar datos al finalizar una compra o cotización, ni otras secciones de tu simulador que no aporten a la interactividad del mismo.

  Si la lógica de tu simulador depende de un registro y un login, o de completar datos para el envío de información o el pago de una compra, ten a bien pre-cargar el contenido de estos formularios y cajas de texto para que la persona que corrige no deba llenarlos de forma manual.

  Recuerda que no solo vemos el código, también probamos la experiencia de uso de la aplicación, por lo tanto si tenemos que llenar formularios, o registrarnos, o loguearnos con datos específicos, debemos dedicar mucho más tiempo a las correcciones y cualquier error involuntario en estos procesos puede hasta invalidar la aprobación de tu proyecto.

  Enfoca tu energía al 100% en que la lógica del proceso de negocio de tu simulador funcione correctamente. No te enfoques en estos detalles secundarios.

- **Guía de uso**

  Si bien tu proyecto puede ser fácil y sencillo, agrega una guía de uso en un archivo **readme.md**, redactada de forma simple. Esto ayuda al corrector a entender ante qué tipo de aplicación web se encuentra, para tener presente la lógica de negocio que debe evaluar.

Simulador interactivo en JavaScript

Ejemplos

Te compartimos algunas ideas a tener presente cuando desarrolles tu simulador interactivo. Estas ideas están basadas en la experiencia y lógica de negocio que se espera a través de diferentes aplicaciones web que son las más elegidas por estudiantes, como presentación de un proyecto integrador.

Quieres construir un simulador de Ecommerce. Entonces:

- Debes poder agregar productos a un carrito
- Si el usuario interrumpe la compra, utiliza webstorage para guardar el carrito parcial, y recupéralo cuando el usuario ingrese nuevamente para continuar el proceso de compra
- Permite que el usuario vacíe el carrito en cualquier momento. Puede ser eliminando el total de productos, o eliminándolos de forma individual
- Al finalizar el proceso de compra, debes mostrar el monto total de la compra y ofrecer un botón COMPRAR. Este último debe simular el fin del proceso de compra eliminando el carrito de Storage y mostrándole al usuario un mensaje de agradecimiento
- Puedes simular un proceso de completitud de formulario de datos personales y la elección de un modo de pago. Si optas por lo primero, pre-completa los datos del formulario con información de fantasía. Si optas por lo segundo, no lo conectes a ninguna pasarela de pagos real y precarga cualquier formulario que solicite datos de una tarjeta, usuario de PayPal, etcétera

Simulador interactivo en JavaScript

Quieres construir un simulador de Ecommerce. Entonces:

- Muestra un listado de productos y/o servicios a cotizar. Construye tu listado de productos y/o servicios en un archivo JSON externo, para que este pueda crecer en opciones sin tener que volver a tocar el código de tu simulador interactivo
- Permite al usuario seleccionar uno o más elementos de dicho listado
- Muestra el total del costo de los productos/servicios seleccionados
- Agrega un botón COTIZAR o ADQUIRIR productos/servicios
- Permite re-cotizar otros productos/servicios
- Utiliza webstorage para mostrar un historial de cotizaciones del usuario
- Cierra el circuito de cotización con algún formulario simple para el envío de datos (*nombre y apellido + correo electrónico*), pre completando la información solicitada
- Finalizado el proceso anterior, agradece al usuario su interés por los productos, y deja la pantalla de tu simulador preparada para que pueda realizarse otra cotización (*limpia campos, totales, etcétera)*

Simulador interactivo en JavaScript

Quieres construir un simulador de Home Banking. Entonces:

- Simula una cuenta bancaria en tu moneda local con saldo igual o mayor a **$ 0.00**. Controla siempre que la cuenta bancaria no quede con saldo negativo si no es una cuenta corriente
- Mantén un histórico de movimientos realizados sobre dicha cuenta (ingresos y egresos), detallando un concepto, fecha y hora de la operación, número genérico de operación y monto (positivo o negativo)
- Maneja el listado de conceptos de movimientos desde un archivo externo, pudiendo agregar o quitar conceptos sin tener que modificar el código de tu simulador interactivo
- En toda referencia de dinero, utiliza el símbolo monetario y el formato numérico (*separador de miles y decimal*) correspondiente a tu país
- Utiliza web storage para almacenar y listar los conceptos de movimientos, listado de contactos o cuentas de otros bancos para transferir dinero, etcétera
- Permite al usuario generar movimientos (*recibiendo transferencias, simulando el pago de tarjetas, simulando el pago de impuestos, transfiriendo dinero a otras cuentas, cotizando préstamos personales*). No son necesarias todas las funcionalidades, pero sí aquellas más comunes en todo sistema de banca electrónica

Simulador interactivo en JavaScript

Quieres construir un conversor de monedas/crypto. Entonces:

- Utiliza algún webservice público que muestra las cotizaciones de monedas en tiempo real (*abundan los servicios online gratuitos*)
- Permite seleccionar una moneda de origen y una de destino para realizar la conversión entre ambas
- Ofrece al usuario un historial de conversiones de moneda realizados, almacenando en webstorage estas operaciones con información adicional como (*Fecha y hora, Tipo de cambio al momento de realizar la conversión, algún ID de operación*)
- En toda referencia de dinero, utiliza el símbolo monetario y el formato numérico (*separador de miles y decimal*) correspondiente a tu país y/o al país de la moneda a convertir
- Agrega alguna funcionalidad adicional, como ser: comprar moneda, solicitando información de su CBU/Swift bancario (ficticio), y valida que los datos ingresados cumplan con un formato estándar (aproximado a los reales)
- Cierra este último proceso con mensajes de confirmación de la compra, transferencia ficticia a su cuenta bancaria, agradecimiento por el uso del servicio, etcétera. Y recuerda vaciar los campos involucrados en el proceso, luego de finalizar cada operación

Simulador interactivo en JavaScript

Ejemplos

Para guiarte, te compartimos algunos ejemplos de trabajos finales:

- Task Master (cotizar trabajos o proyectos)
- Tienda de bebidas
- Booking vacacional
- Simulador de préstamo personal

Criterios de evaluación

Para la evaluación de tu Proyecto Final, tendremos en cuenta los siguientes criterios de evaluación.