UD2E7 – Taller mecánico

Vamos a desarrollar una aplicación web dinámica para la gestión de un taller mecánico utilizando JavaScript orientado a objetos (POO), módulos ES6 y un enfoque Single Page Application (SPA). El objetivo es reforzar los conocimientos de programación orientada a objetos.

NOTA: en este proyecto tienes ciertas libertades para codificar la solución. Se te proponen una estructura inicial que debes respetar y a partir de ahí ampliar según consideres.

El modelo de datos se divide en varias entidades: Vehículo, Propietario, Reparación y Trabajo. La relación entre Vehículo y Propietario es 1:1, y la relación entre Reparación y Trabajo es 1:N.

Los datos iniciales se (vehículos y reparaciones) están definidos en un archivo externo (datos-taller.js), permitiendo que la aplicación cargue su estado inicial desde ahí.

Estructura lógica del proyecto

- Clase Vehículo
- Clase Propietario
- Clase Reparación
- Clase Trabajo
- Clase BD
- Clase GestiónMecánica

La clase “GestiónMecánica”. Tiene la siguiente estructura:

Propiedades privadas

- **clienteBD**, objeto de tipo BD, sólo lectura.
- **contenedor**, elemento HTML donde montar la aplicación dinámicamente, sólo lectura.

Métodos públicos

- **iniciarApp(selector)**, busca el documento el elemento HTML indicado por el selector, y se le asocia a la propiedad “contenedor”. Si no lo encuentra mostrará un mensaje de información indicando que no se ha podido iniciar la aplicación.

Métodos privados

- **generaHTMLBase()**, devuelve una cadena HTML con el código HTML de base que incluye el menú principal (inicio, listado vehículos, listado no terminadas, listado no pagadas, listado presupuestos) y un contenedor para resultados (donde se mostrarán los distintos formularios).
- **generarHTMLInicio()**, devuelve una cadena HTML con el código HTML de la página de inicio. La página de inicio incluirá un buscador que permita filtrar por matricula o teléfono. El resultado de una búsqueda será la lista de vehículos.

- **generarHTMLVehiculos(vehiculos)**, devuelve una cadena HTML con el código HTML del listado de vehículos, cada entrada dispondrá de las opciones: ver vehículo, ver reparaciones y borrar vehículo.
  El listado incluirá un botón para crear un nuevo vehículo.
- **generarHTMLVehiculo(vehiculoId=null)**, devuelve una cadena HTML con el código HTML del formulario de propiedades del vehículo. Si el vehiculoId es null estamos creando un nuevo vehículo. Habrá un recuadro para los datos del propietario.
  Cuando el formulario no sea de creación incluirá una opción para ver las reparaciones del vehículo.
- **generarHTMLReparacionesVehiculo(vehiculoId)**, devuelve una cadena HTML con el código HTML del listado de reparaciones del vehículo ordenadas por fecha descendente.
  En la parte superior se mostrará la matricula y teléfono del propietario del vehículo, así como la opción de ver vehículo.
- **generarHTMLReparaciones(reparaciones)**, devuelve una cadena HTML con el código HTML del listado de reparaciones, cada entrada dispondrá de las opciones: ver y borrar reparación.
- **generarHTMLReparación(reparaciónId=0, vehiculoId=0)**, devuelve una cadena HTML con el código HTML del formulario de reparación. Si la reparación es nueva (reparacionId = 0) se empleará el parámetro vehículoId para establecer la relación.
  Incluirá un recuadro/formulario para añadir un nuevo trabajo a la colección de trabajos.
  Incluirá una lista de trabajos, cada trabajo contará con la opción de borrar trabajo.
  NOTA: la manera fácil de añadir un nuevo trabajo es, SIN RECARGAR, mediante JS añadir un nuevo elemento a la lista. Reparación y trabajos son un único formulario y con el submit van las dos entidades.
- Añade los métodos necesarios para gestionar la asignación de eventos y operaciones CRUD de las distintas entidades.

La clase “BD” actúa como nuestro acceso a datos. Tiene la siguiente estructura:

Propiedades privadas

- **“vehículos”** y **“reparaciones”**. Colecciones con los objetos. Se cargan al instanciar la clase a partir de los datos de prueba.
- **“siguienteVehiculoId”** y **“siguienteReparacionId”**. Índice autoincremental para la creación de vehículos y reparaciones. (¿Me vendría bien que fueran un getter? -> ¿cómo lo implementaría correctamente?).

Métodos

- **obtenerVehiculos()**, devuelve la lista de vehículos.
- **obtenerVehiculo(filtro, valor)**, devuelve el vehículo que cumple con alguno de los criterios de filtrado. Posibles valores para “filtro”: “vehiculoId”, “matricula” y “teléfono”. El parámetro “valor” tendrá un valor acorde al filtro seleccionado.
- **crearVehiculo(vehiculo)**
- **borrarVehiculo(vehiculoId)**

- **obtenerReparaciones(filtro, valor)**, devuelve la lista reparaciones, si hay un criterio de filtrado se aplica. Posibles valores para “filtro”: “fecha”, “pagado” y “terminado”. El parámetro “valor” tendrá un valor acorde al filtro seleccionado.
- **obtenerReparacion(reparacionId)**
- **crearReparación(vehiculoId, reparación)**
- **borrarReparación(reparacionId)**

Las clases Vehículo, Propietario, Reparación y Trabajo son objetos planos (sin código). Es posible que te convenga definir algún método o propiedad adicional.

La aplicación se cargará inicialmente desde un único archivo HTML (**index.html**) que contendrá un contenedor. Se cargará el JavaScript y se invocará al método “iniciarApp”, a partir de este momento todo el código de la página será gestionado por el módulo.

AMPLIACIÓN: (Este apartado se verá en próximos temas). Si te sobra tiempo y te atreves prueba a añadir opciones de serialización y deserialización al almacenamiento del navegador, objeto “local storage”. Es un almacenamiento asociativo en el que puedes almacenar la información como cadena JSON.

Recomendaciones

Crea un gráfico con el modelo de datos.

Crea un gráfico con las ventanas de la aplicación y la relación entre ellas. También te puede ayudar identificar que métodos te van a hacer falta en cada una de las ventanas.

Piensa en las restricciones iniciales y en el objetivo final, todo lo que falta debes aportarlo acorde a lo solicitado.

Crea la estructura del proyecto, piensa como lo vas a construir y añade comentarios.

Se plantea el uso de un formulario con 2 entidades con una relación de 1:N, plantéate probarlo de manera independiente.