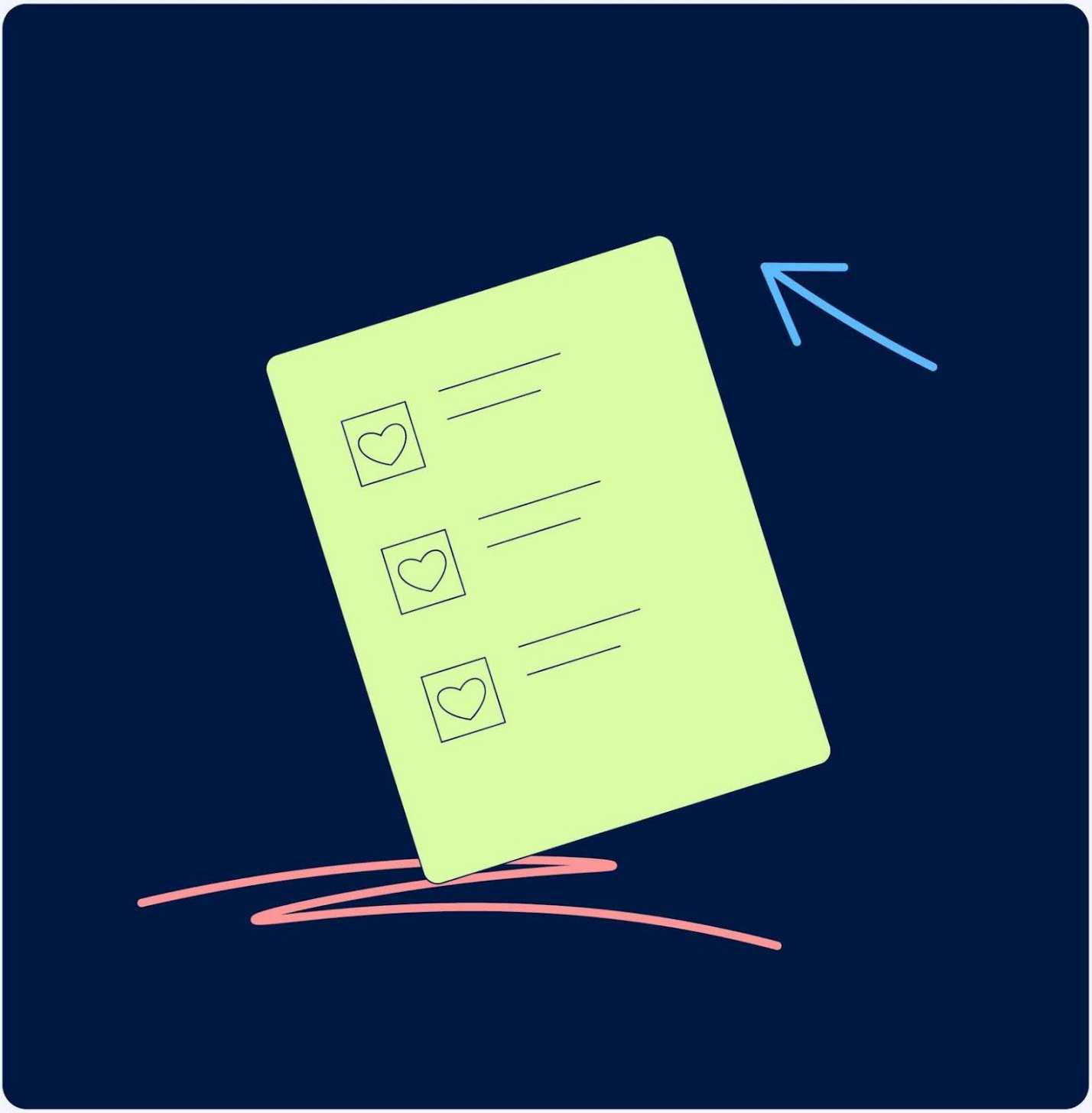

Problema Lab 03

Java

AÑADE ENDPOINTS A TU APP DE REGISTRO DE VEHÍCULOS

En este lab vamos a estudiar a fondo los *endpoints* y los métodos de llamada, la comprensión de ambos puntos facilita a su vez la elección del método más apropiado según cada función. Para ello, continuaremos trabajando en la aplicación de registro de vehículos anterior. Antes de empezar a leer el enunciado del problema, asegúrate de que controlas los conceptos que hemos visto a lo largo de este lab y en el fastbook.

Objetivos de este ejercicio

- Adquirir soltura en la creación de endpoints y en la definición de los métodos de consulta.
- Aprender a definir parámetros para una petición.
- Entender cómo añadir documentación con *Swagger*.

Descripción de la actividad

El objetivo perseguimos con este nuevo ejercicio que planteamos es que sigas desarrollando tu app de registro de vehículos. La idea es que continúes ampliando esta aplicación arquetípica donde, a estas alturas, ya debes tener definida su base principal.

¿Y cuál es el siguiente paso? Te pedimos que **añadas un listado de endpoints**. Cada uno de estos irá asociado a una finalidad específica. Además, también deberás añadir objetos que te servirán a su vez de esqueleto a la hora de definir y trabajar cada elemento.

A continuación, te compartimos las condiciones generales que debes tener en cuenta durante el proceso de trabajo. ¡Toma nota!

1. Crea un endpoint POST para la adición de vehículos el cual recibirá un objeto de tipo *Car* como parámetro. Si existe algún error durante el proceso de guardado, devolveremos un error ***500 – Internal Server Error.*** Si todo va como debería (en este y el resto de endpoints), devolveremos un ***200 – Ok***.

   Estas son las propiedades que deberá contener este objeto:

   - Id – Integer;
   - Brand – String;
   - Model – String;
   - Milleage – Integer;
   - Price – Double;
   - Year – Integer;
   - Description – String;
   - Colour – String;
   - FuelType – String;
   - numDoors – Integer.

2. Crea un endpoint GET para la consulta de vehículos el cual recibirá un id en la propia URL y devolverá el objeto solicitado. En caso de que el objeto no exista, deberás devolver un estado ***404 – Not Found***.

3. Crea un endpoint PUT para la actualización de vehículos el cual recibirá un id en la propia URL y un objeto Car como cuerpo de la petición y devolverá el objeto actualizado. En caso de que el objeto no exista, deberás devolver un estado *404 – Not Found*. Si existe algún error durante el proceso de actualización devolverá un error *500 – Internal Server Error.*

4. Crea un endpoint DELETE para la eliminación de un vehículo concreto. Recibirá un id en la propia URL y devolverás un 200 – Ok si todo ha ido bien. Si no existiese el coche a eliminar, devuelve un *404 – Not Found.*

5. Configura y añade Swagger a la aplicación para poder consultar sus endpoints, parámetros, objetos y respuestas.

Formato de entrega

Envía tu ejercicio en un archivo con extensión .java o, si has utilizado más de una clase para resolver la actividad, un archivo comprimido en formato .zip o .rar con el conjunto de clases que has utilizado durante la resolución del problema.

Criterios de corrección

Te compartimos algunos puntos que debes comprobar sí o sí cuando vayas a autoevaluar tu ejercicio antes de darlo por resuelto:

- en primer lugar, tienes que comprobar que la aplicación funciona de manera correcta (para ello, puedes probar y crear una lista de objetos en el *respository* y consultar, añadir, actualizar o eliminar elementos de esa lista);
- en segundo lugar, asegúrate de que los datos que se muestran son los solicitados;
- por último, confirma que la documentación se muestra de manera adecuada. Asegúrate de incluir comentarios sobre cada funcionalidad.

Qualentum