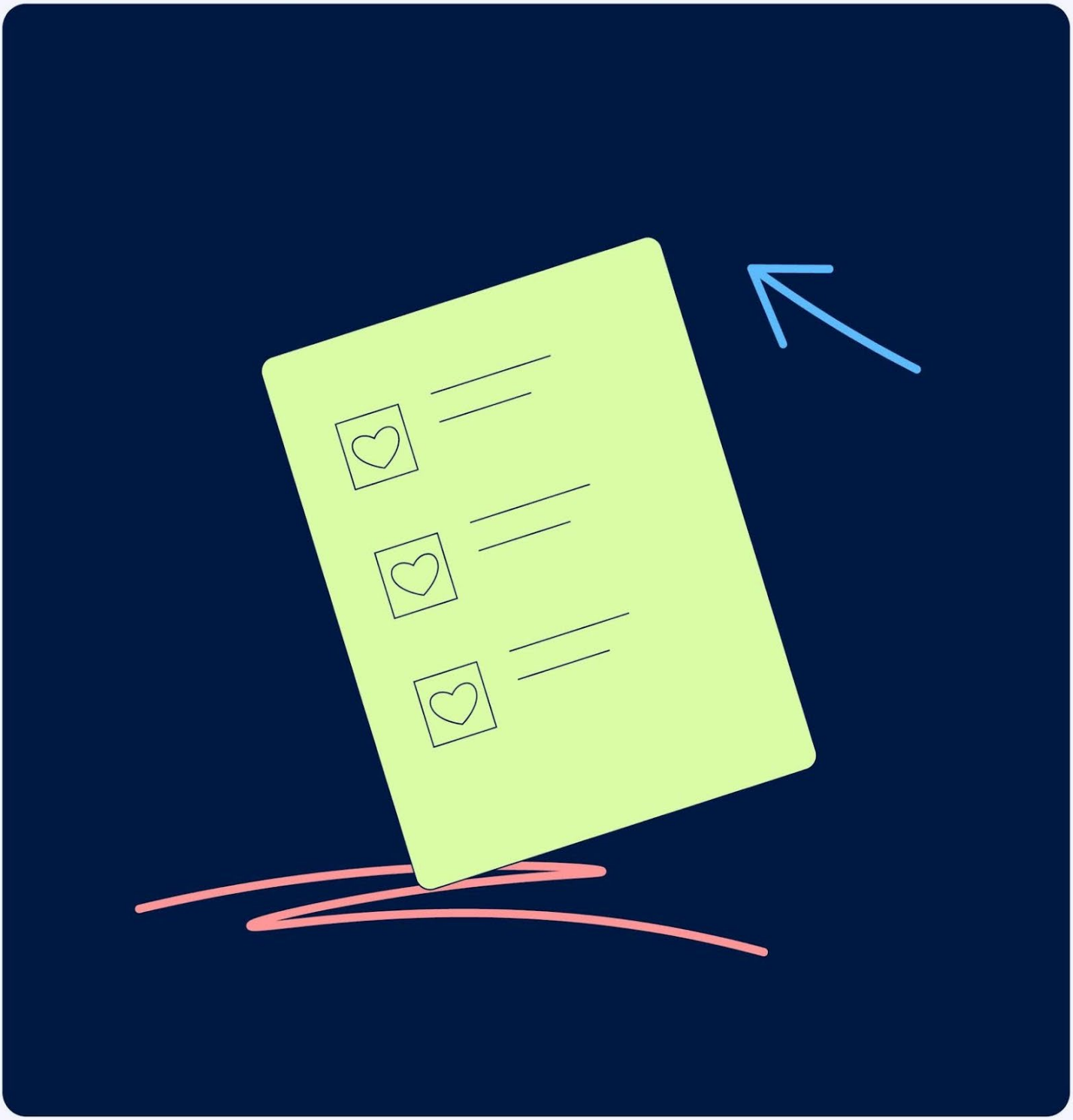

Problema Lab 08

Java

CONFIGURA UN ENDPOINT PARA ALMACENAR IMÁGENES EN TU APP DE REGISTRO DE VEHÍCULOS

Gracias a este lab hemos aprendido a manejar y a gestionar los archivos y recursos dentro del contexto de las aplicaciones REST API. También a enviar y recibir archivos desde *endpoints*.

Con estas lecciones podremos aportar más funcionalidad y nuevas opciones a nuestras aplicaciones, logrando que la adición o la descarga de datos pueda hacerse de forma masiva y ordenada. Interesante, ¿verdad?

Objetivos de este ejercicio

- Ampliar el ejercicio del lab anterior añadiendo más funcionalidades a la app de registro de vehículos.
- Aprender a configurar e incluir *endpoints* para la gestión de archivos tanto para su recepción como para su descarga en la aplicación en Java.
- Razonar qué estructura o pasos se deben seguir para alcanzar el objetivo propuesto.

Descripción de la actividad

Toca continuar con el proyecto de la app de registro de vehículos. De nuevo, toca ampliar la funcionalidad que ya tienes implementada para enriquecerla todavía más.

Para ello, te pedimos que trabajes la configuración para que se puedan **añadir imágenes** y **registros** a través de los archivos dentro de tu aplicación en Spring Boot. En concreto, debes incluir **cuatro *endpoints*** para poder gestionar estas funcionalidades que te comentamos.

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

- Crea un *endpoint* que **almacene una imagen** para un usuario que ya existe. Sus parámetros de llamada serán el id y un *MultipartFile*. Debes comprobar que el id existe. La imagen tendrás que almacenarla en la base de datos, en el formato que desees. A este *endpoint* podrán acceder tanto los clientes como los vendedores. Devuelve un mensaje para informar al usuario en caso de que la imagen se almacene con éxito.

- Establece otro *endpoint* adicional para poder **descargar la imagen** de un usuario a partir de su id. A este *endpoint* podrán acceder tanto los clientes como los vendedores.

- Genera uno más para poder **descargar los datos** de los coches almacenados en la base de datos en formato CSV. A este *endpoint* podrán acceder tanto los clientes como los vendedores.

- Por último, crea un último ***endpoint*** para poder **añadir coches** a la base de datos. A *endpoint* solo podrán acceder los vendedores que recibirán un archivo en formato CSV. Devuelve un mensaje en caso de que el archivo se cargue con éxito para informar al usuario.

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 empleadas durante la resolución del problema.

Criterios de valoración

Te compartimos algunos puntos que debes comprobar sí o sí cuando vayas a visualizar la solución del profesor y a autoevaluar tu ejercicio antes de darlo por superado.

- Primero, comprueba que la aplicación funciona de forma correcta y que el mensaje de inicio se muestra al arrancar el puerto 8080 sin fallos.
- Segundo, añade comentarios sobre cada funcionalidad para comprender mejor todo el proceso de desarrollo del código, ¿de acuerdo?

Si has sido capaz de implementar todos estos puntos de forma adecuada, ¡enhorabuena! Ya sabes manejar y almacenar archivos en tu aplicación en Java y Spring Boot y comprendes tanto el uso como las ventajas de esta a la hora de crear aplicaciones más accesibles y dinámicas. ¡Reto conseguido, querido estudiante!

Qualentum