

# Package oriented design

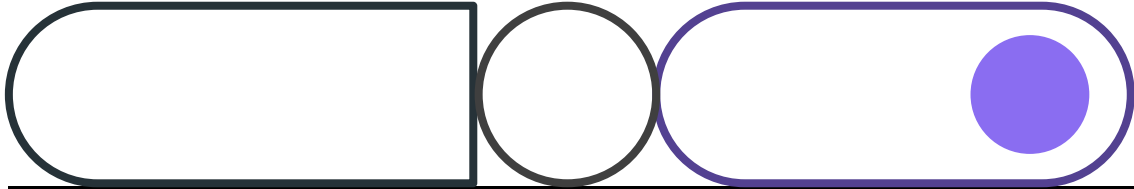
# Índice

- 01 [¿Qué es?](#)
- 02 [Controlador](#)
- 03 [Servicio](#)
- 04 [Repositorio](#)



01

¿Qué es?



Es una manera de estructurar los proyectos muy popular en Go, en donde se toma como filosofía los **paquetes** para la composición de una aplicación.



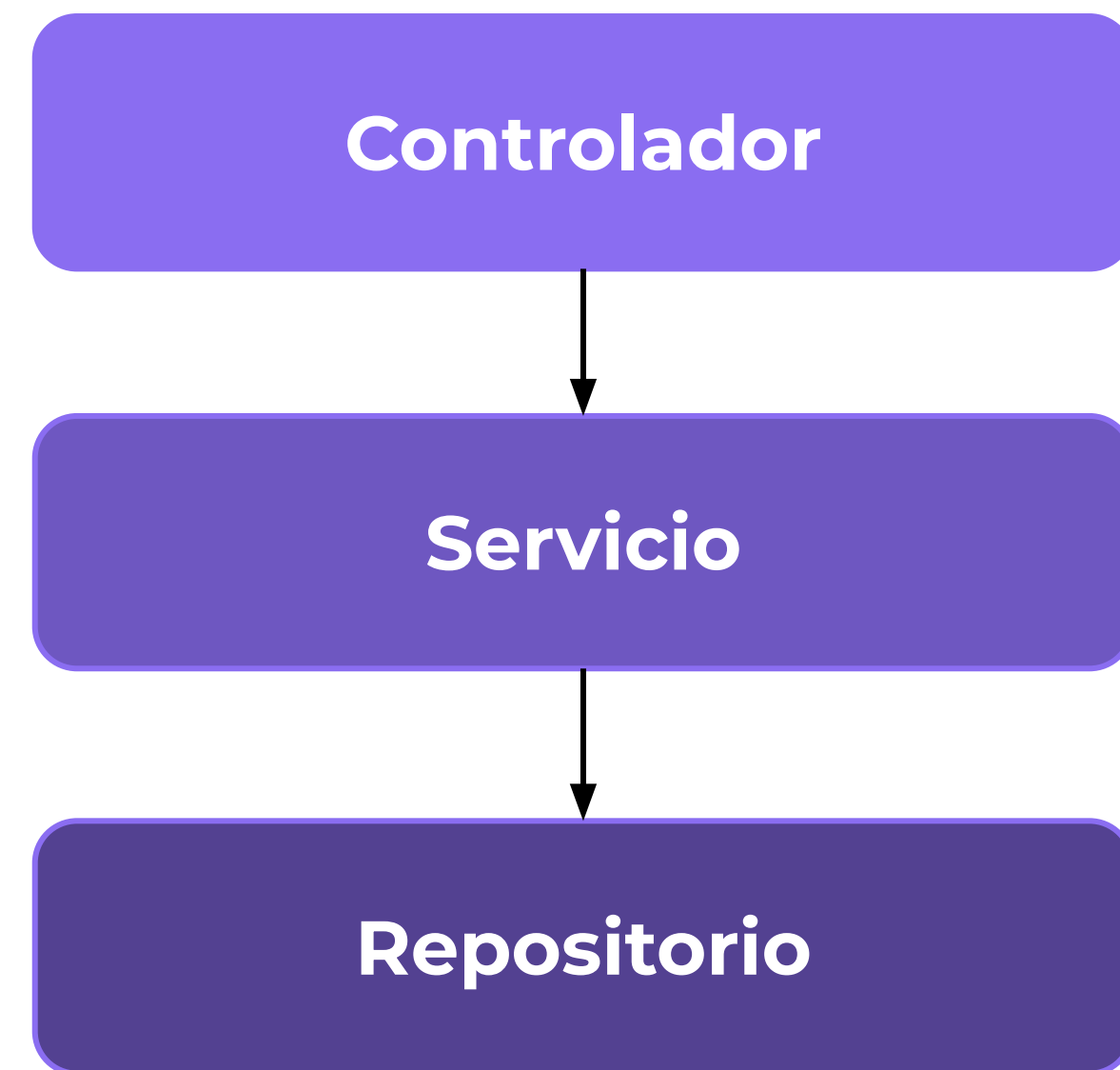
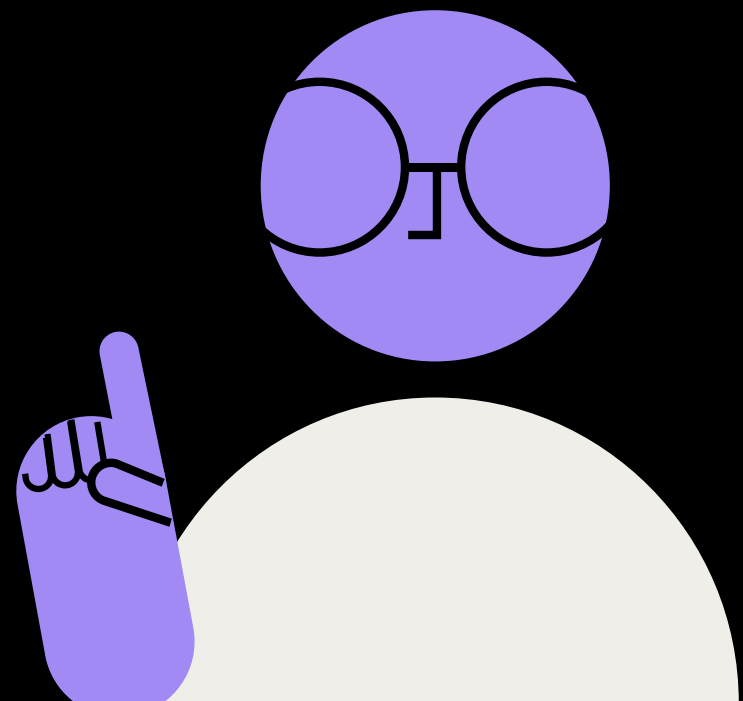
## ¿Para qué sirve?

Package oriented design le permite a la persona desarrolladora saber a dónde pertenece cada paquete de un proyecto en Go.



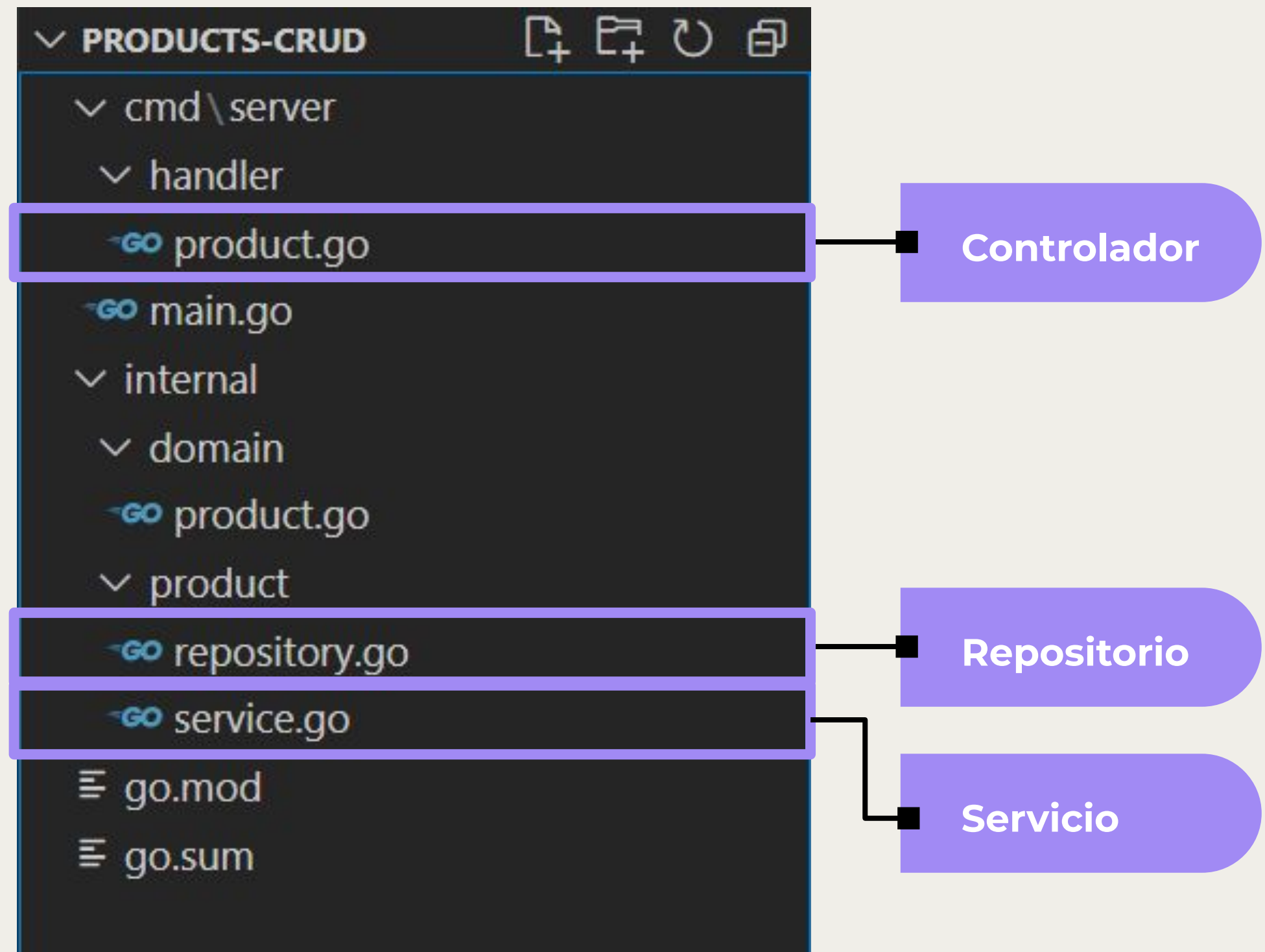
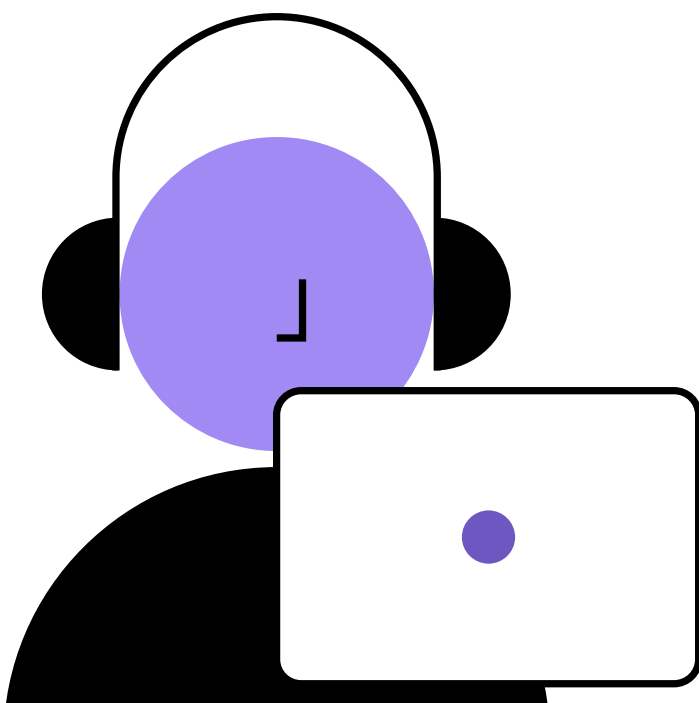
# Estructura de una aplicación

Dividiremos nuestra aplicación en tres partes:



# Estructura

Con esta metodología, vamos a guardar todos los controladores sobre la capa handler. El servicio y el repositorio quedarán dentro de la carpeta de nuestro elemento (en el caso del ejemplo, “producto”).



02

# Controlador




## ¿Qué es?

El controlador (o handler) se encarga de recibir la petición del cliente, validar valores requeridos, ejecutar los diferentes servicios y retornar una respuesta.

✓ **PRODUCTS-CRUD**

✓ `cmd\server`

✓ `handler`

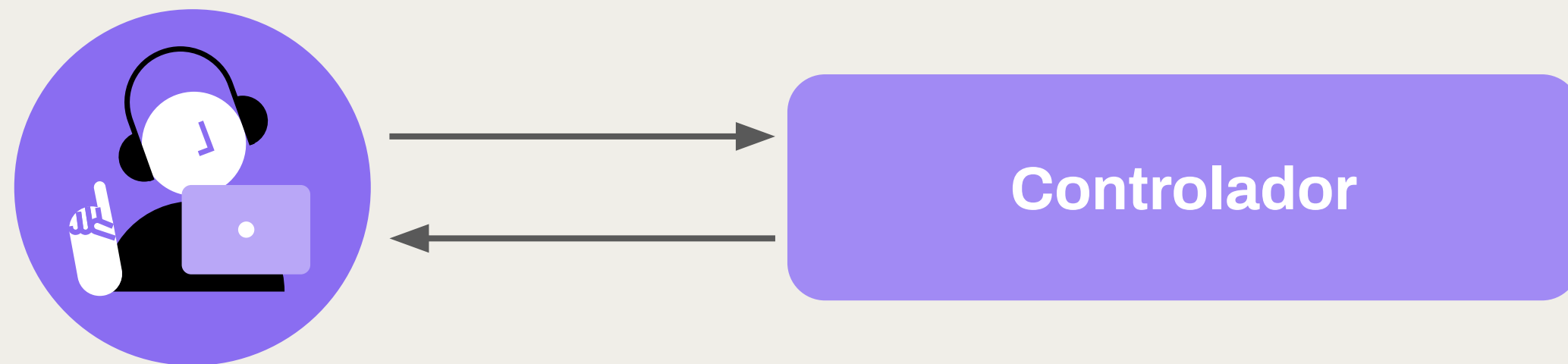
 `product.go`



# ¿Para qué sirve?

Todas las peticiones que reciba nuestra aplicación web deben pasar por el controlador.

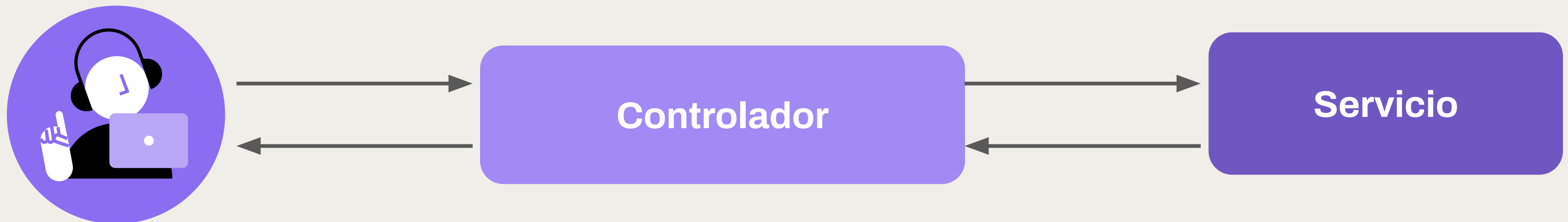
El controlador se encarga de verificar la petición y, si encuentra algún problema, devuelve una respuesta informando al cliente.



# Controlador

En caso de que la información de la petición sea correcta, el controlador le pasa la tarea al servicio.

Una vez que el servicio termina, le devuelve la respuesta al cliente.

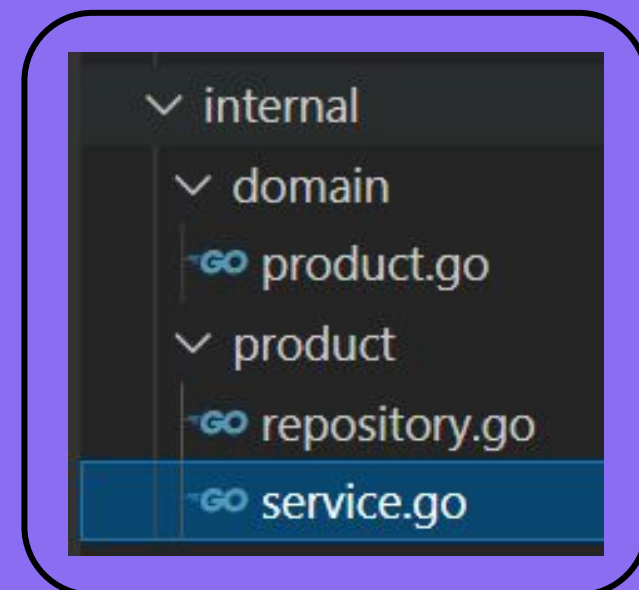


03

Servicio

## ¿Qué es?

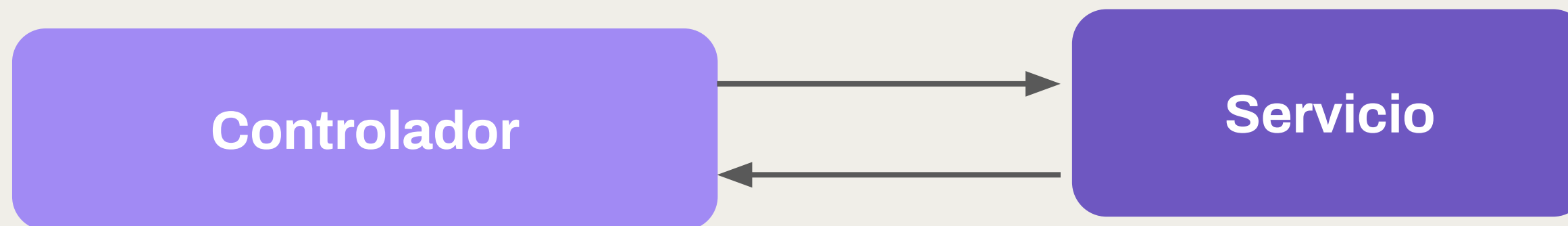
El servicio es el encargado de realizar las funciones principales de la aplicación: procesamiento de datos, implementación de funciones de negocio y administración de recursos externos.



# Procesamiento de datos

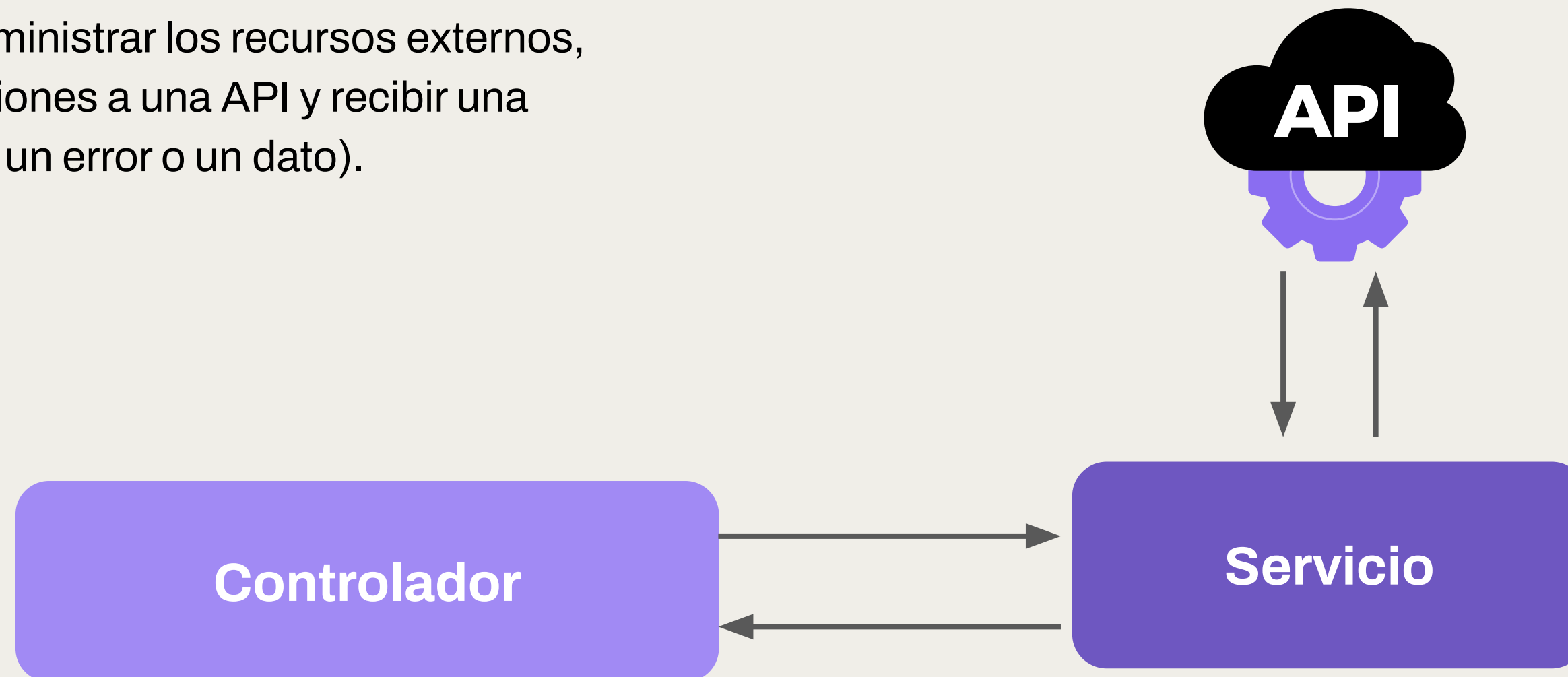
El servicio recibe los datos enviados a través del controlador y se encarga de realizar las funcionalidades principales de la aplicación con la lógica de negocio. No debemos enviarle el **request**, sino los datos necesarios para realizar el procesamiento de datos y enviar una respuesta al controlador.

El controlador espera recibir el dato que está necesitando, o un error en caso de que haya surgido algún problema en el servicio.



# Administración de recursos externos

Se encarga de administrar los recursos externos, como enviar peticiones a una API y recibir una respuesta (ya sea un error o un dato).



# Administración de recursos externos

En caso de necesitar información de la base de datos, el **servicio** envía su petición al **repositorio**. Este se encarga de enviarle una respuesta al **servicio**, ya sea el dato que necesita el **servicio** o un error.





04

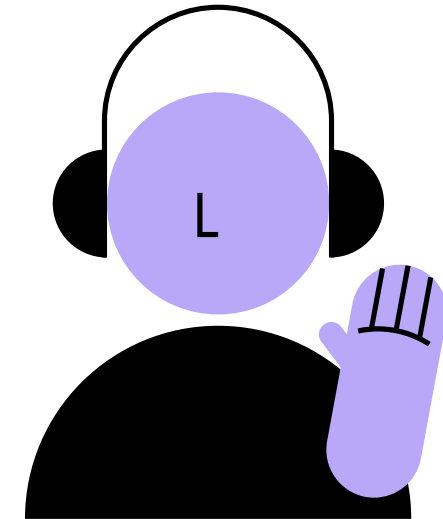
# Repositorio

## ¿Qué es?

El repositorio se encarga de abstraer el acceso a los datos, siendo el encargado de interactuar con la base de datos.

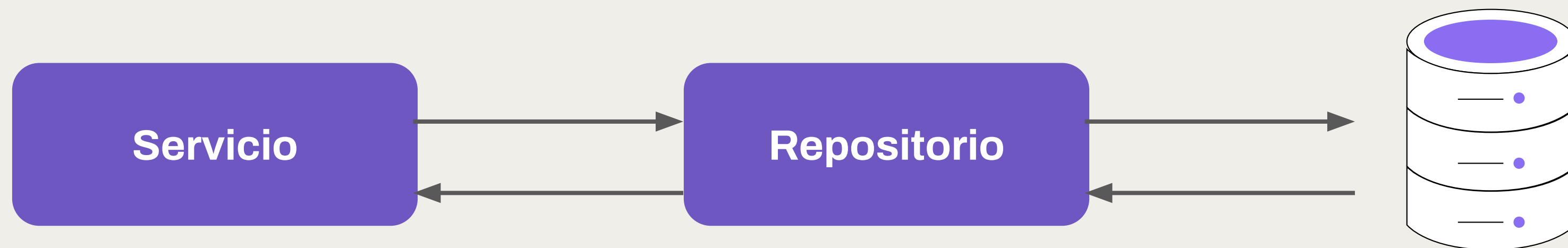


# ¿Para qué sirve?



Siempre que el **servicio** requiera información de la base de datos, deberá pedirla a través del **repositorio**.

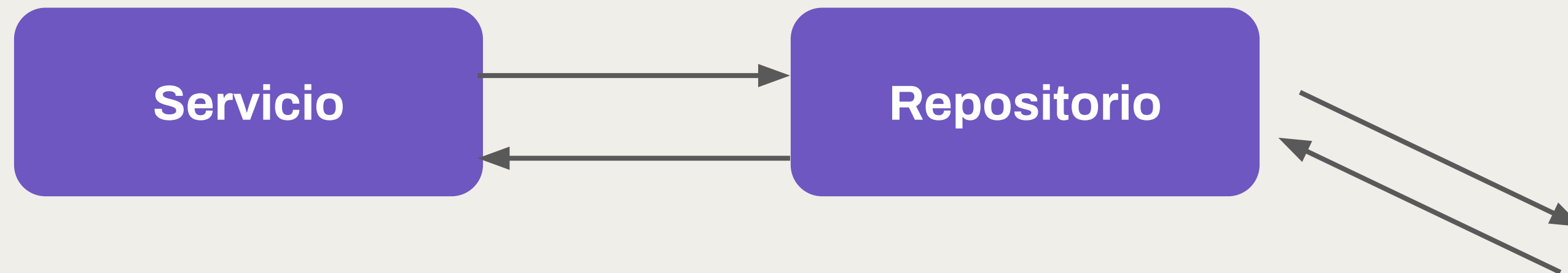
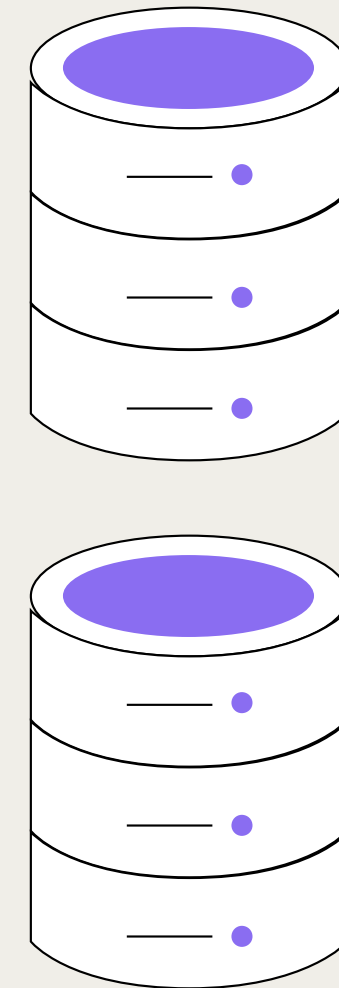
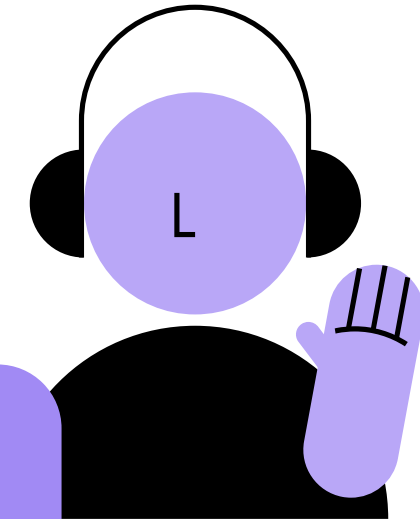
El **repositorio** es el que se encarga de realizar las operaciones en la base de datos y devolverle la información al **servicio**.



# Repositorio

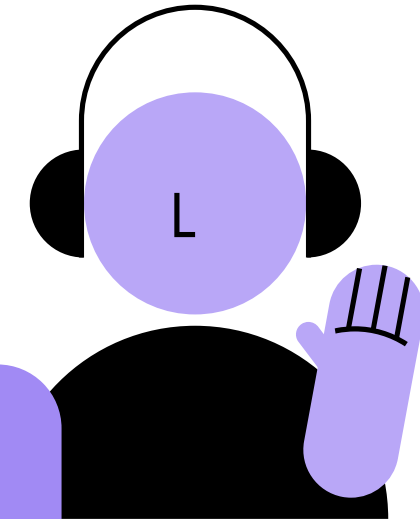
Si nosotros necesitamos hacer un cambio de base de datos...

El **servicio** solo manda la petición y el **repositorio** es el que se encarga de ir a buscar la información al lugar que tenga que buscarla.

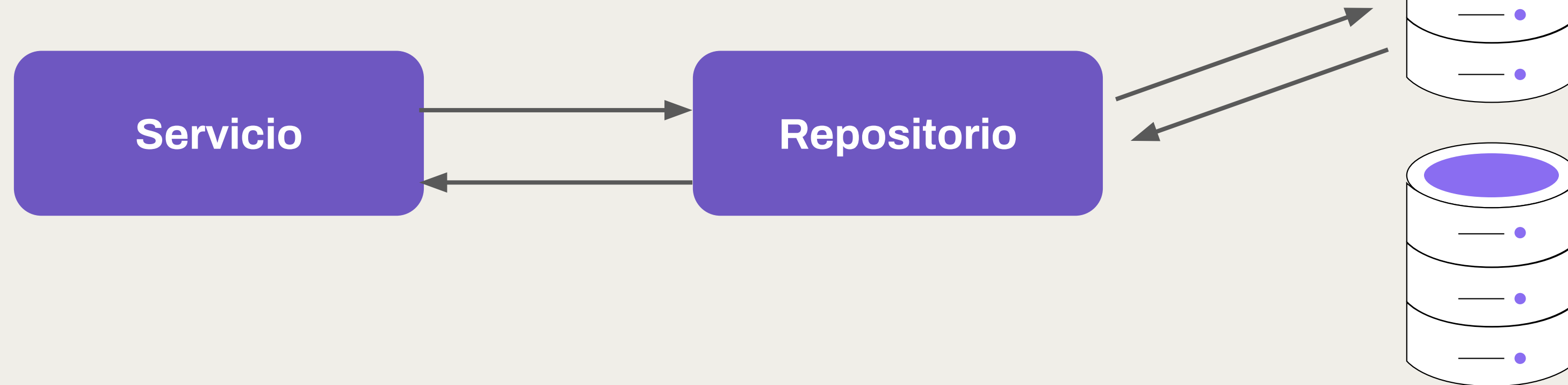


# Repositorio

El cambio solo lo debemos hacer en el **repositorio**.  
Ese cambio no debe afectar el **servicio**.



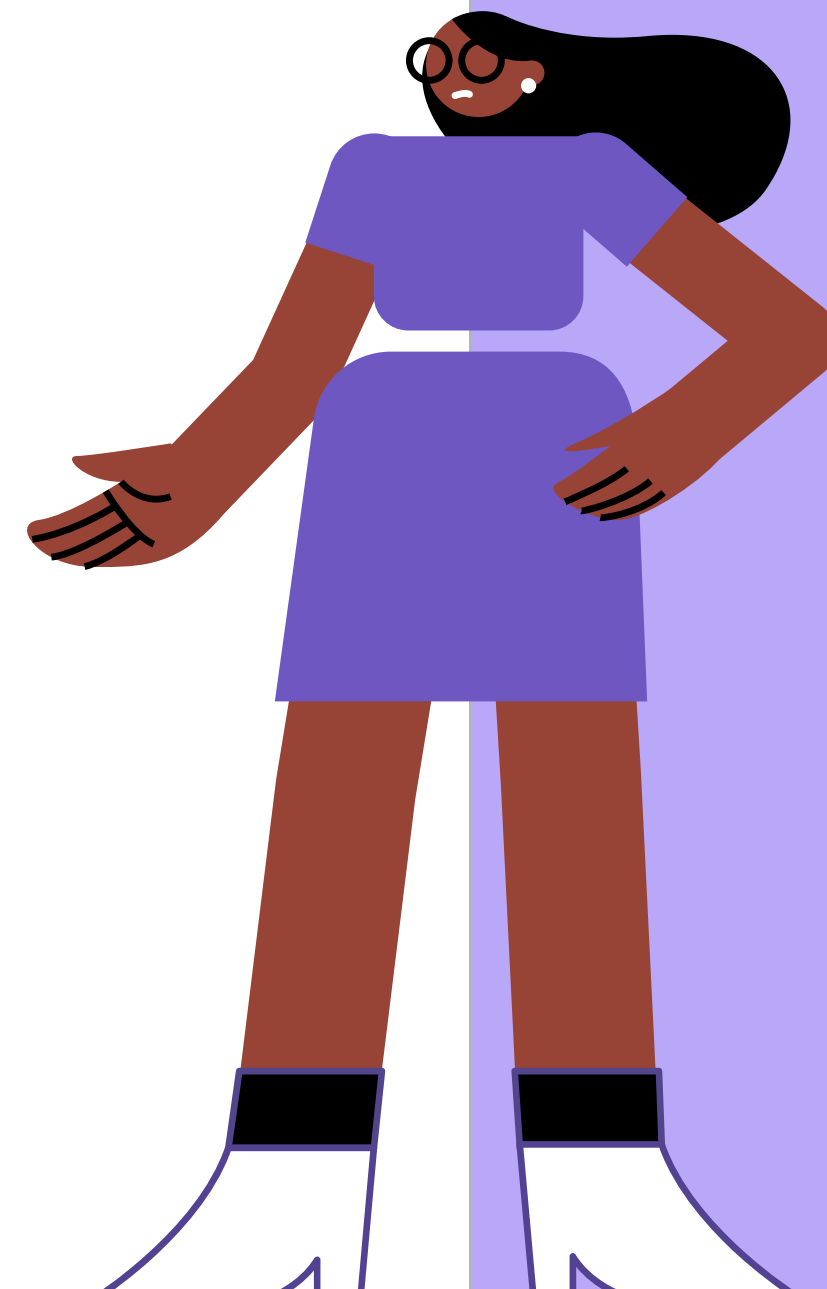
El **servicio** solo manda la petición y el **repositorio** es el que se encarga de ir a buscar la información al lugar que tenga que buscarla.



# Conclusiones

En esta presentación terminamos de profundizar sobre los paquetes.

Además, aprendimos sobre package oriented design, una forma de organizar nuestras aplicaciones para que puedan crecer sin problemas y ser mantenibles con facilidad.



¡Muchas gracias!