Read: **Command y Query Bus - ¿Qué son?**
Completed · 100 Points

Ya sabemos qué son las Queries y los Commands, ahora vamos a aprender por donde "se mueven".

Nos subimos al bus

Para analizar qué nos aporta el Command y Query bus, vamos a analizar las distintas alternativas disponibles

Lógica de negocio en El Controlador

Típicamente nos podemos encontrar aplicaciones que reciben la petición y de forma inmediata ejecutan la lógica que corresponde.

El problema que nos encontramos en este tipo de escenarios es en términos de **reutilización**. Desde el momento en el que la lógica de negocio está en el punto de entrada, **si tenemos otro punto de entrada adicional, no podremos reaprovecharla y deberemos duplicarla.**

Lógica de negocio en El Caso de Uso

En esta segunda iteración, encapsularemos la lógica de negocio en un servicio aislado al que le pasaremos los datos necesarios desde nuestros controladores.

El beneficio que nos aporta es que **ahora sí podemos tener múltiples puntos de entrada que ejecuten una misma lógica de negocio**.

El problema es que **no estaríamos rompiendo en ningún caso el flujo de ejecución de nuestras peticiones**.

Incorporación del Command/Query Bus

Ya sabemos qué es un Command y una Query. Un DTO que instancia el controlador (punto de entrada) que lanzaremos al bus correspondiente (QueryBus o CommandBus). Este bus será el encargado de trasladar el DTO a su Handler correspondiente. El Handler finalmente invocará nuestro caso de uso que habíamos extraído anteriormente.

El beneficio de esto es que el controlador está totalmente desacoplado de la lógica de negocio. **Si queremos cambiar el caso de uso a ejecutar, sólo deberemos modificar el Handler correspondiente para que ejecute el nuevo caso de uso**.

Otro beneficio es que ahora quién mapea los valores de entrada a objetos de dominio (Value Objects) para que el caso de uso los reciba, es el Handler. De esta forma los controladores no conocen nada de nuestro dominio y, por lo tanto, **podríamos incluso tener los controladores en un determinado lenguaje de programación, y los servicios que ejecuten la lógica de negocio en otro**.

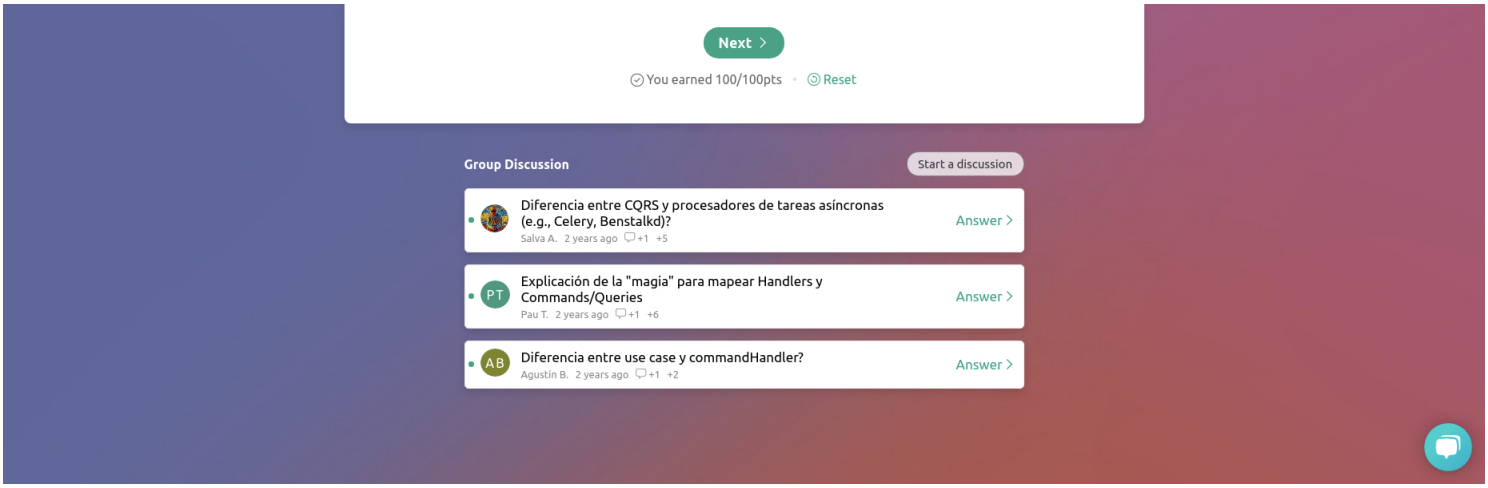
Además, desde el momento en el que introducimos un nivel de indirección adicional entre el punto de entrada y la lógica a ejecutar, podemos jugar con la asincronía. Con lo cuál, también podemos tener **ganancias en términos de rendimiento**.

Por último, también **podremos envolver nuestros handlers o casos de uso con middlewares para que ejecuten ciertas lógicas transversales** como logging, transaccionabilidad, etc.

¡Nos vemos en el siguiente vídeo para bajar al detalle de implementación!

Discuss: ¿Se te ocurren escenarios en los que aplicar un Command/Query bus? Answer >

Discuss: ¿Qué librerías conoces para implementar command/query bus? Answer >


Next
You earned 100/100pts
Reset
Group Discussion
Start a discussion
Diferencia entre CQRS y procesadores de tareas asíncronas (e.g., Celery, Benstalkd)?
Salva A. 2 years ago +1 +5
Answer
PT
Explicación de la "magia" para mapear Handlers y Commands/Queries
Pau T. 2 years ago +1 +6
Answer
AB
Diferencia entre use case y commandHandler?
Agustín B. 2 years ago +1 +2
Answer