Utilizar Entity Framework Core en controladores

Resumen

- **Introducción al patrón repositorio**
- **Aprender sobre código asíncrono**
- **Lectura, creación, actualización y eliminación de recursos mediante Entity Framework Core**
- **Uso de AutoMapper**

Introducción al patrón repositorio

Sin repositorio

- **Duplicación de código**
- **Código más propenso a errores**
- **Es más difícil probar la clase que utiliza el código**

Patrón Repositorio

El patrón repositorio

Una abstracción que reduce la complejidad y tiene como objetivo hacer que el código por una parte, sea seguro para la implementación del repositorio, e ignore los detalles sobre la persistencia.

Introducción al patrón repositorio

Sin repositorio

- **Duplicación de código**
- **Código más propenso a errores**
- **Es más difícil probar la clase que utiliza el código**

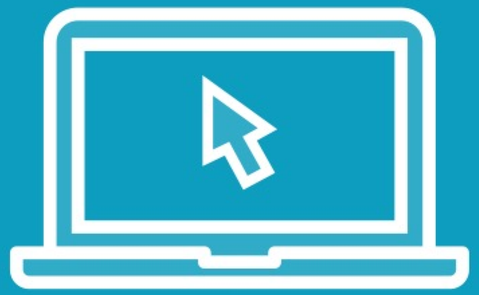
Repository pattern

- **No hay duplicación de código**
- **Código menos propenso a errores**
- **Mejor capacidad de prueba de la clase que lo utiliza**

Ignora detalles sobre la persistencia

Cambiar la tecnología de persistencia no es el objetivo principal.
Elegir la mejor para cada método de repositorio sí lo es.

Demo

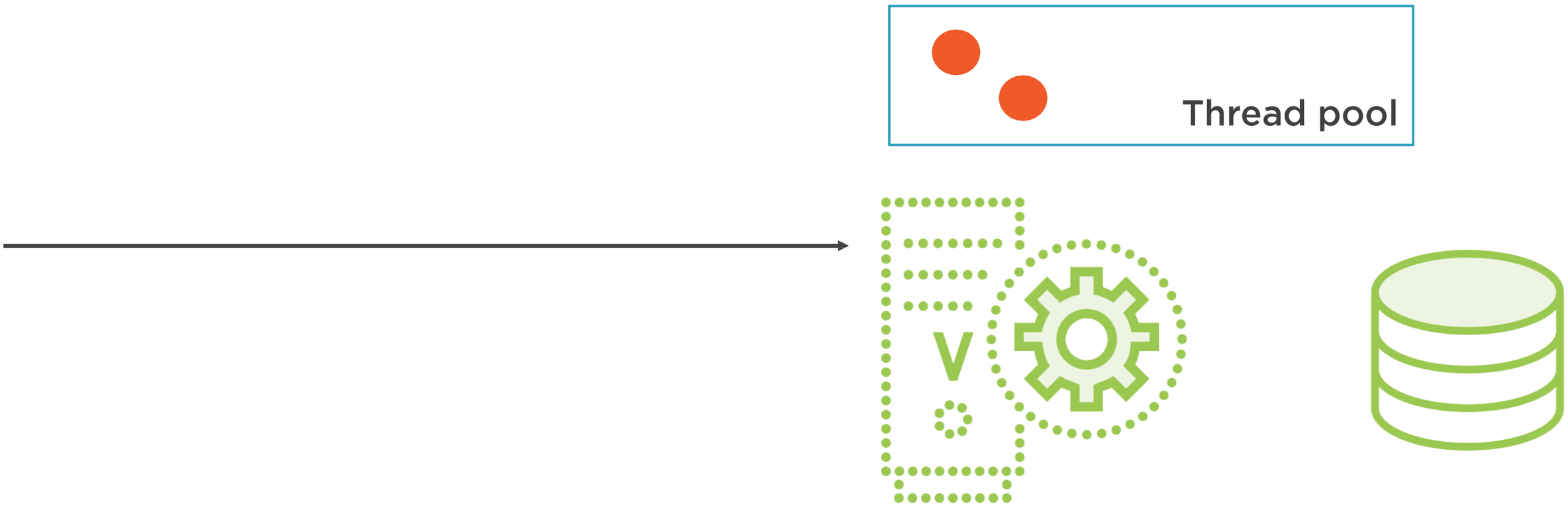


Introducción al patrón repositorio (parte 1)

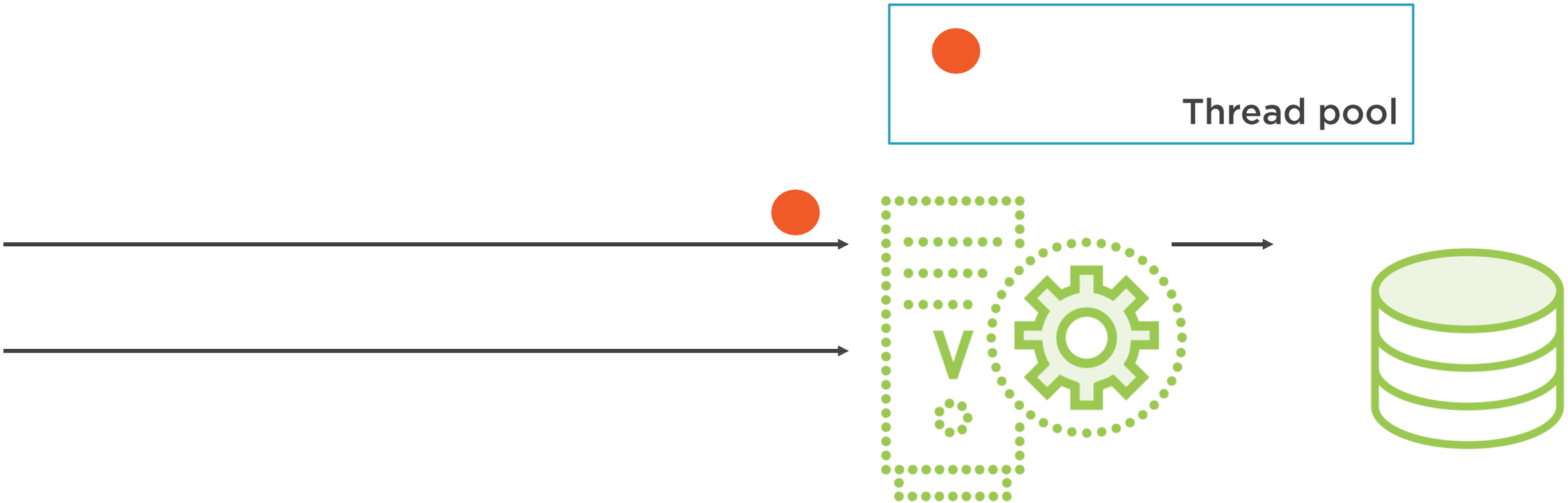
La finalidad del código asíncrono

Liberación de subprocesos para que puedan utilizarse en otras tareas, lo que mejora la escalabilidad de la aplicación.

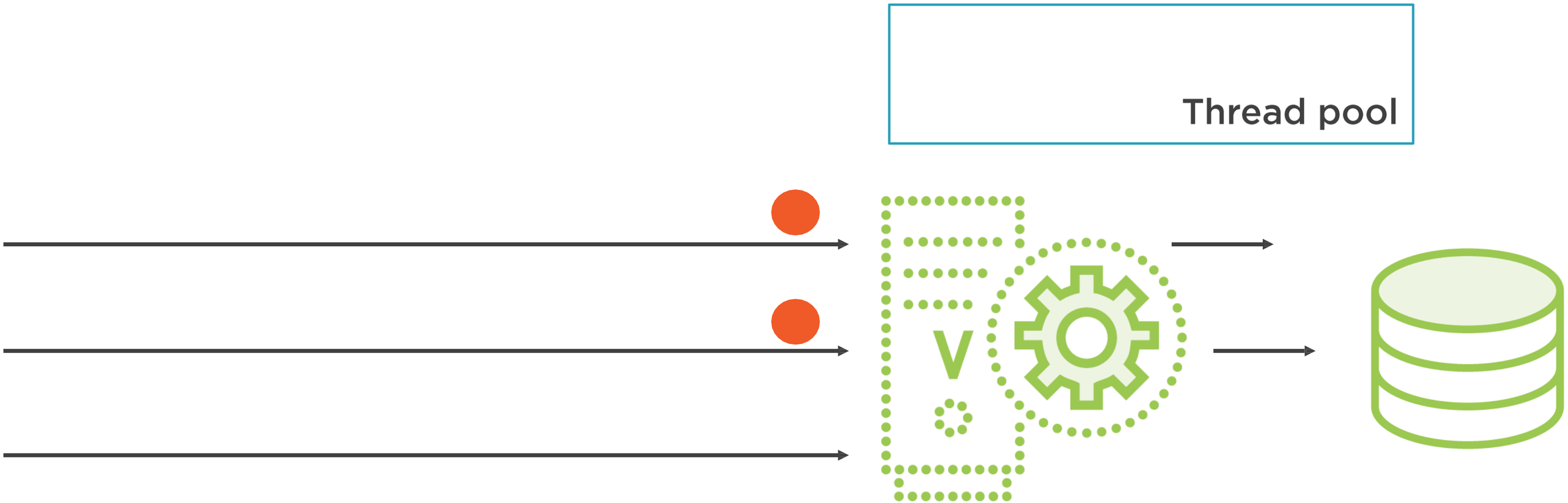
Administración de peticiones de forma síncrona



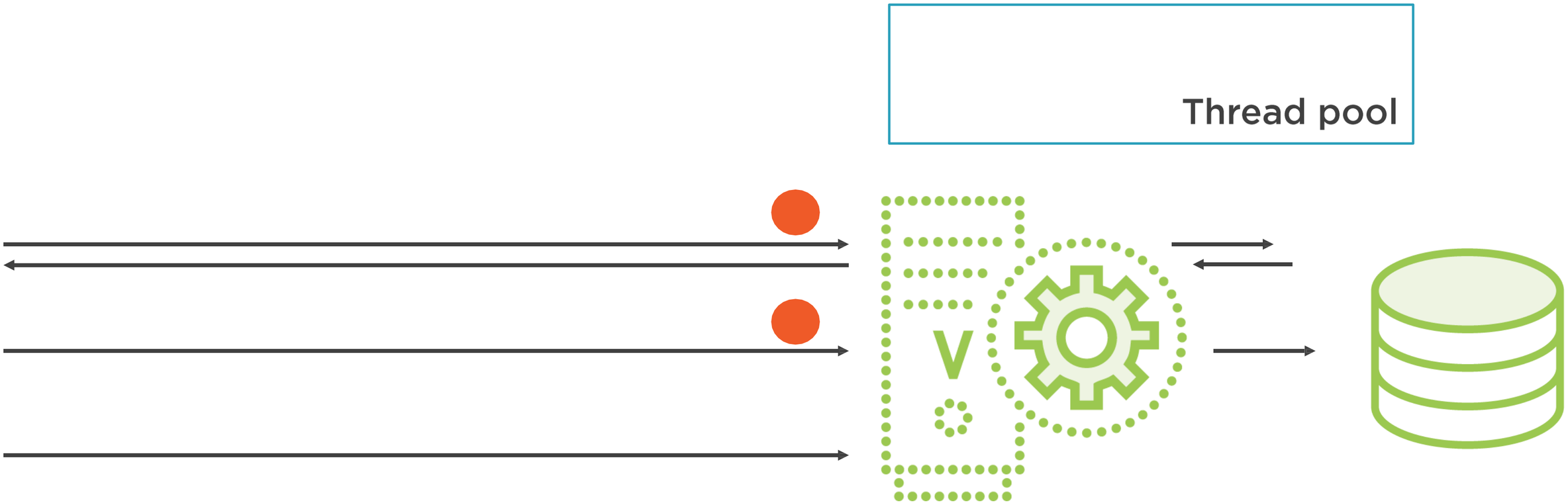
Administración de peticiones de forma síncrona



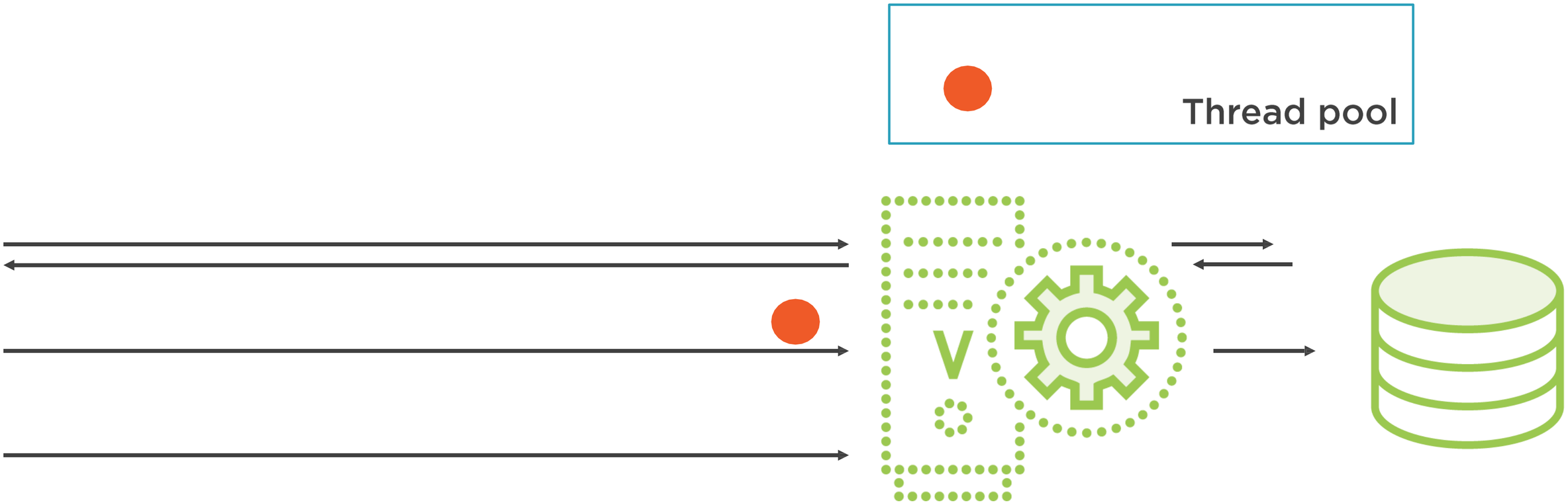
Administración de peticiones de forma síncrona



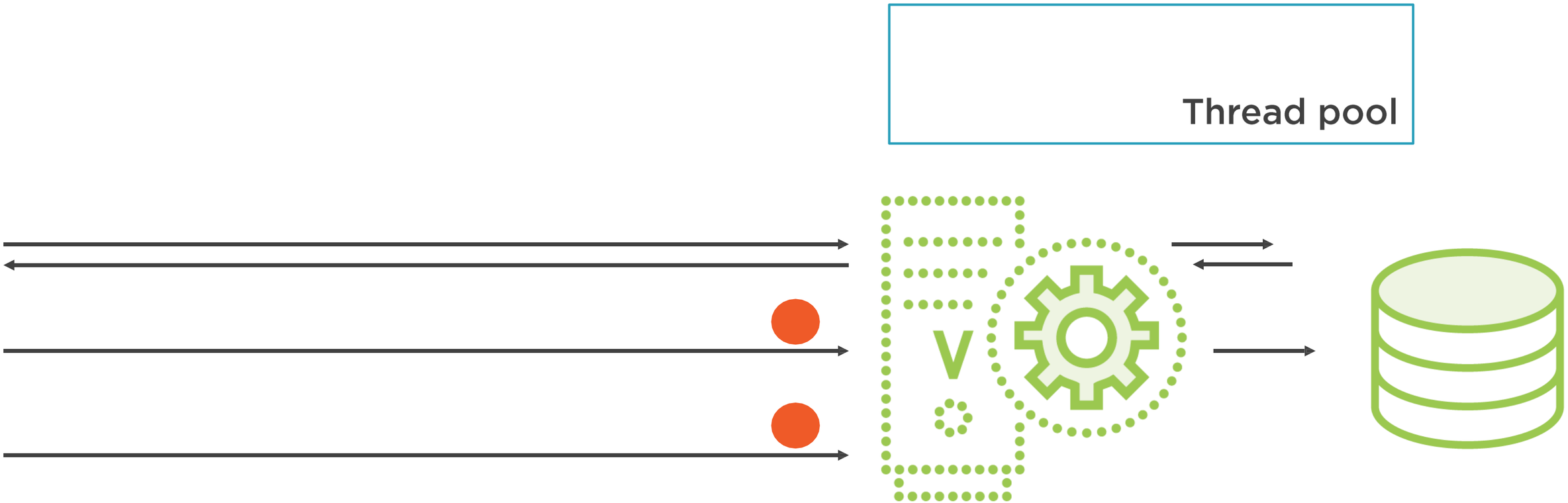
Administración de peticiones de forma síncrona



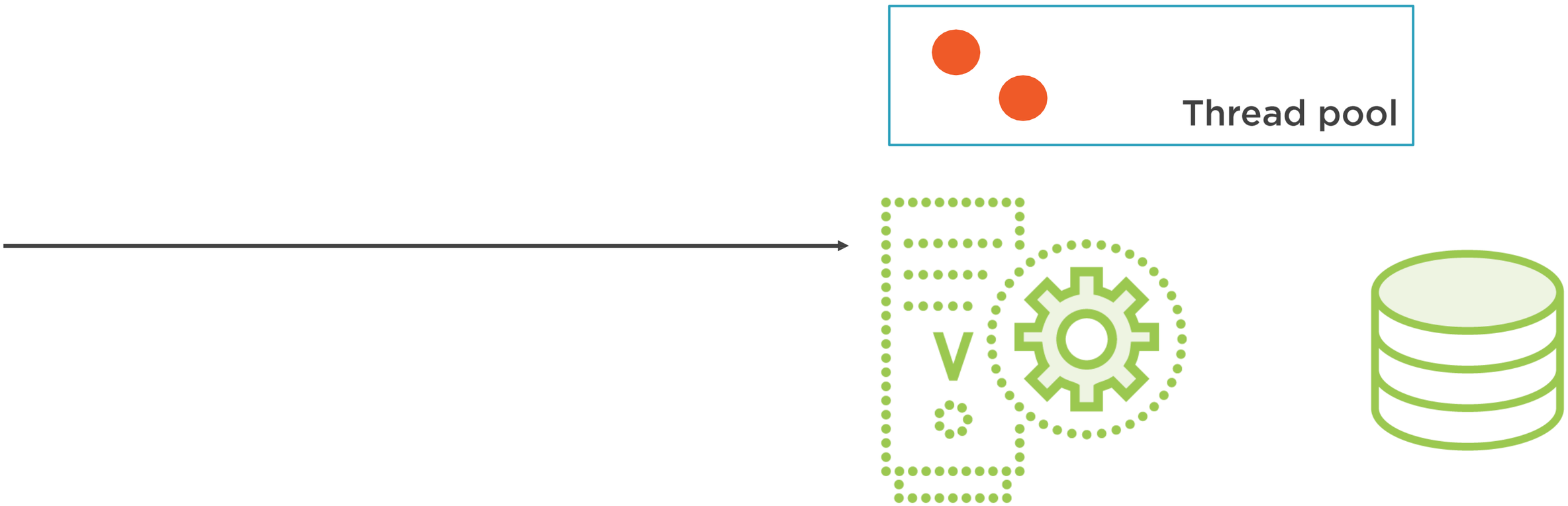
Administración de peticiones de forma síncrona



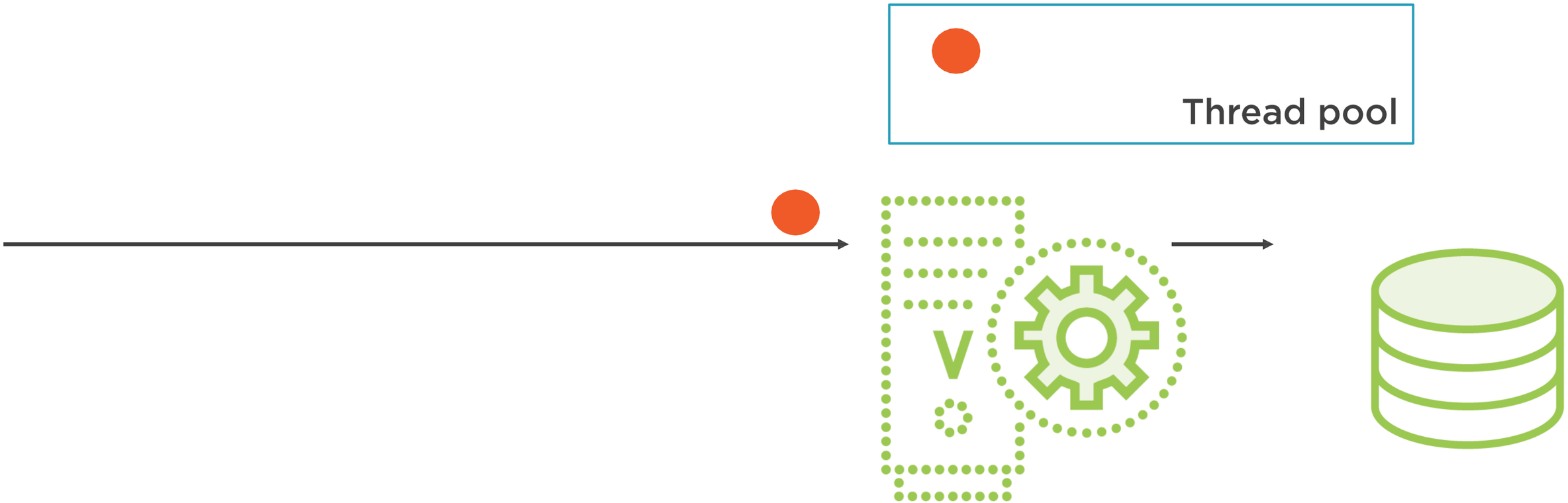
Administración de peticiones de forma síncrona



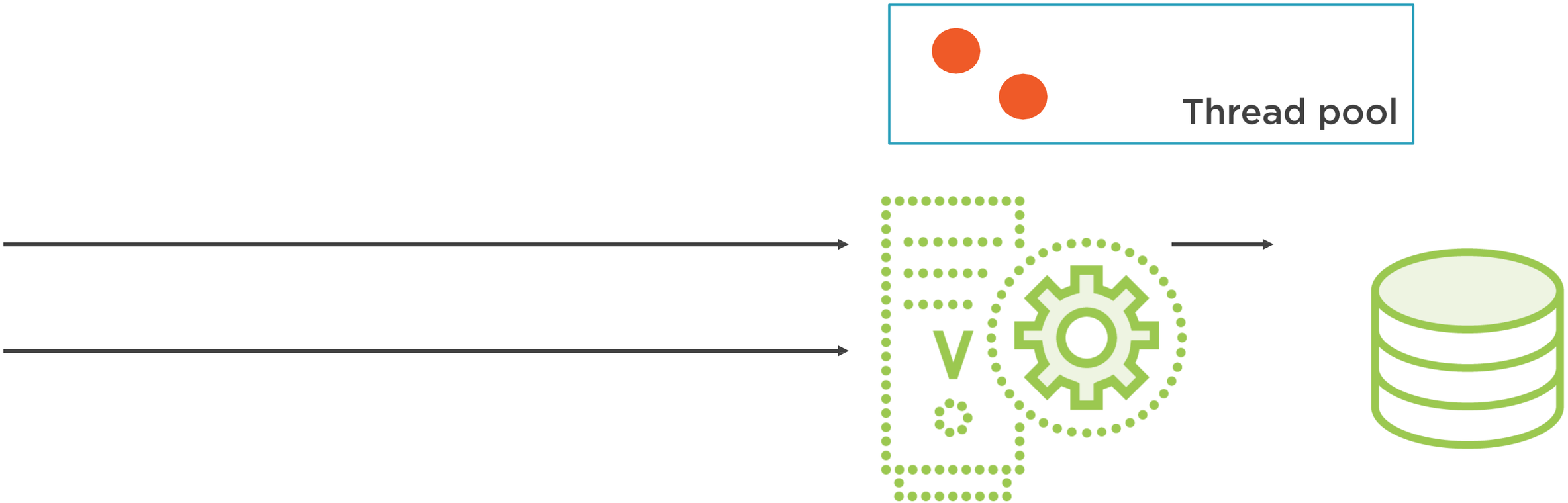
Administración de peticiones de forma síncrona



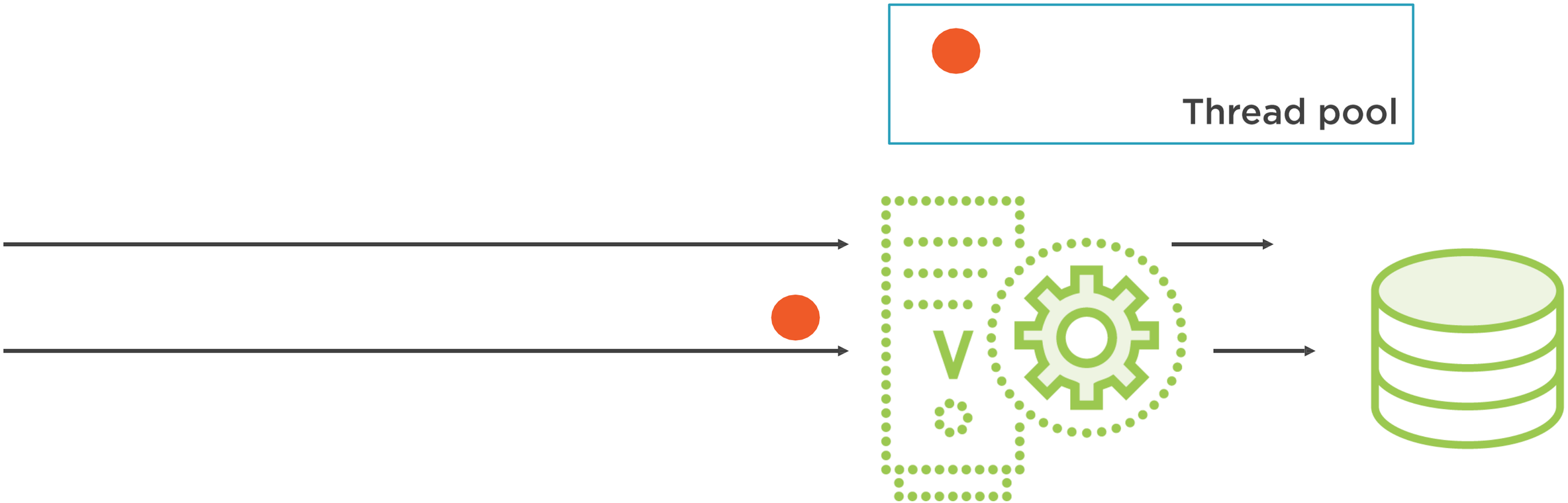
Administración de peticiones de forma síncrona



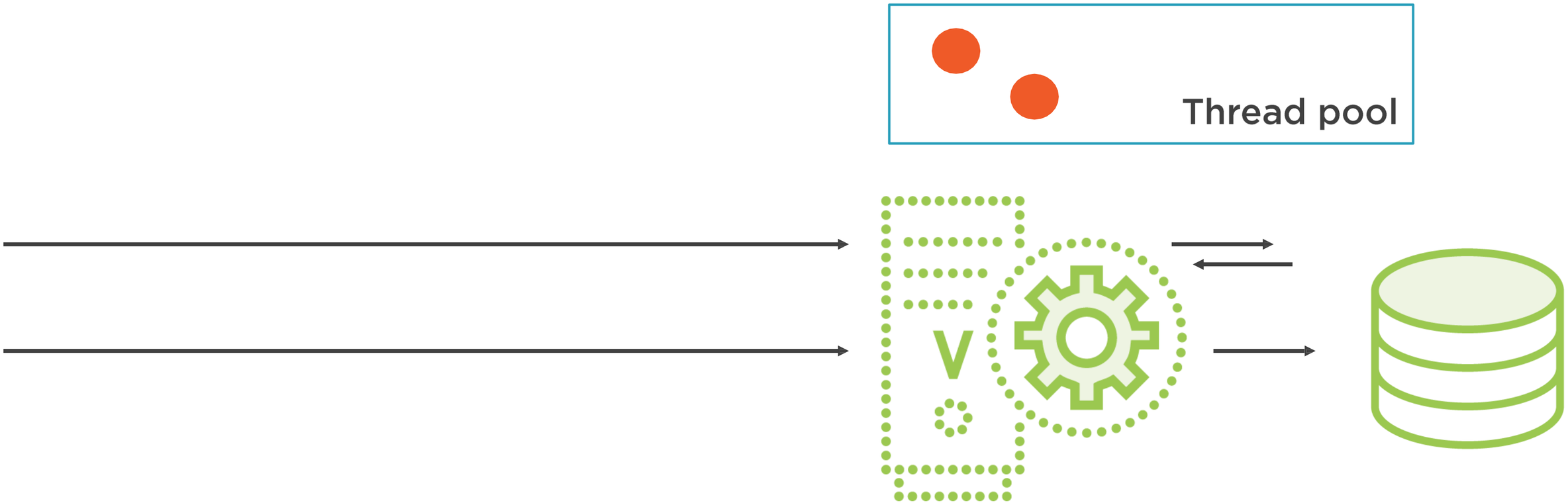
Administración de peticiones de forma síncrona



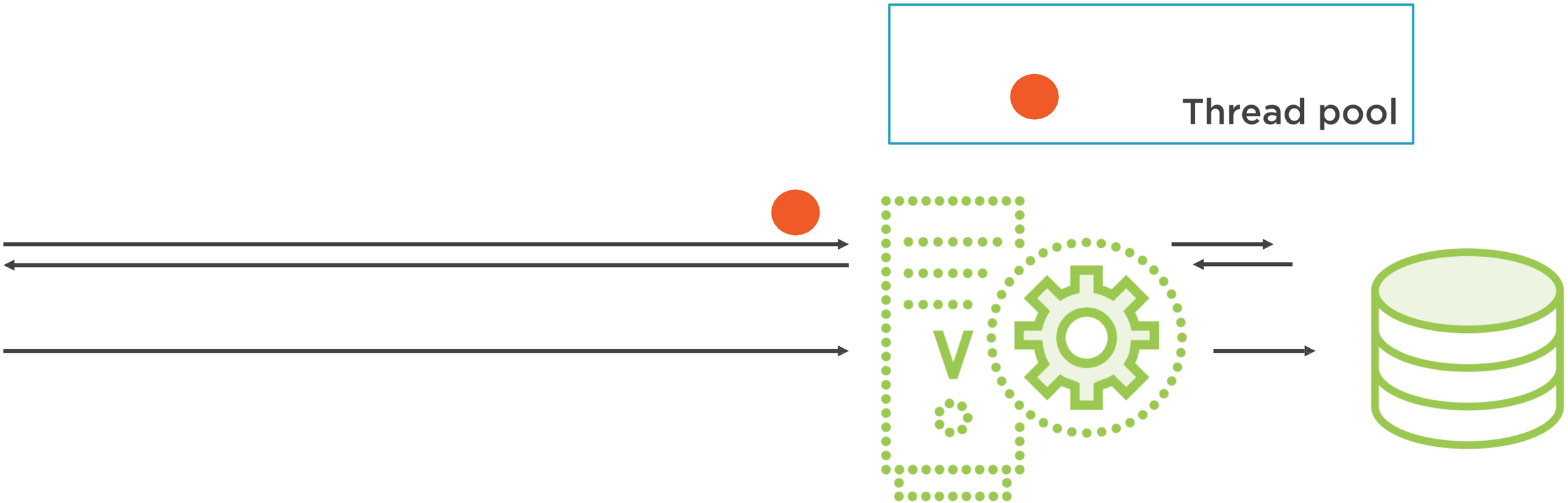
Administración de peticiones de forma síncrona



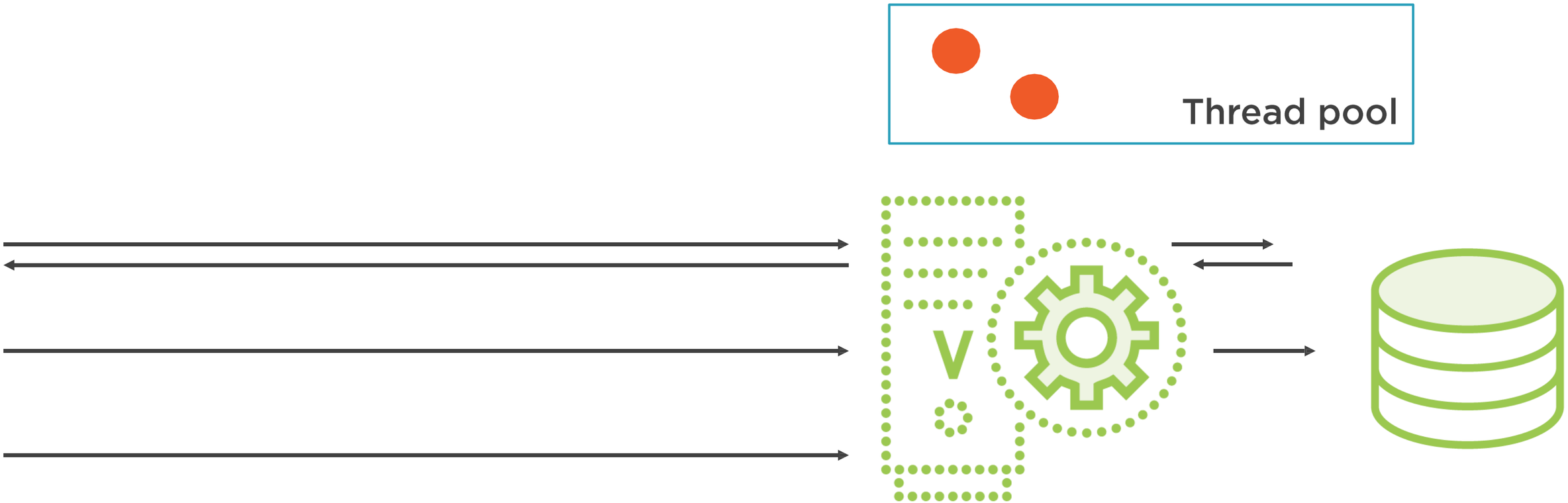
Administración de peticiones de forma síncrona



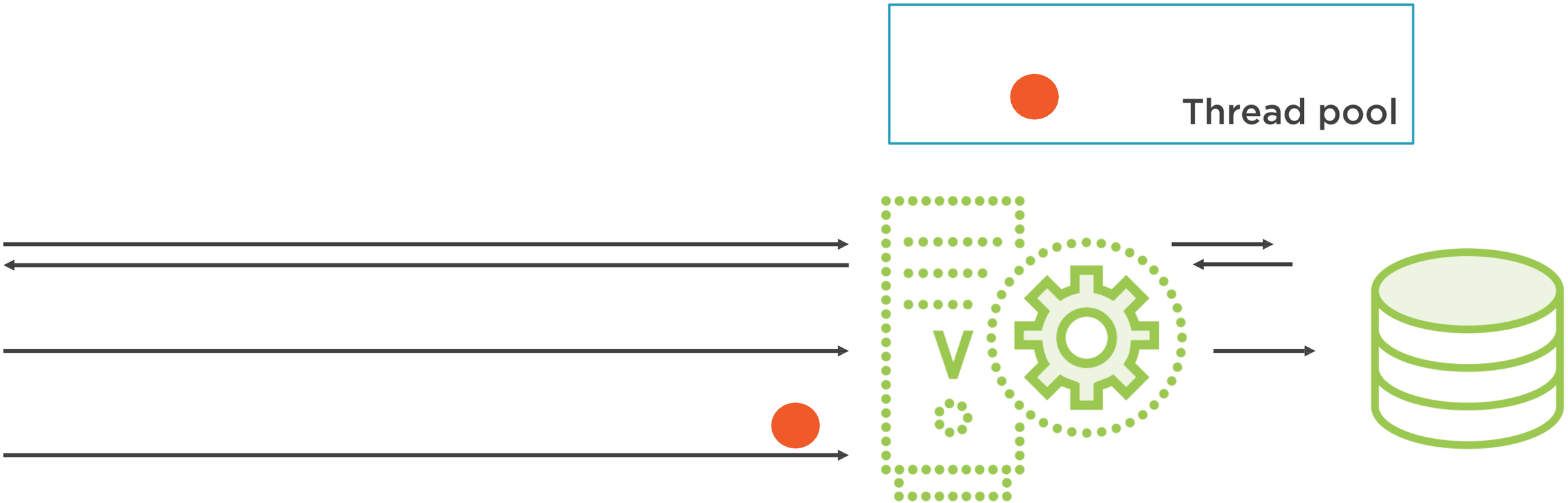
Administración de peticiones de forma síncrona



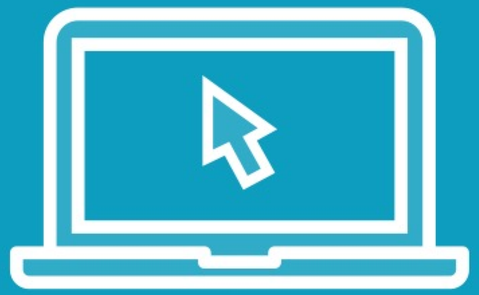
Administración de peticiones de forma síncrona



Administración de peticiones de forma síncrona

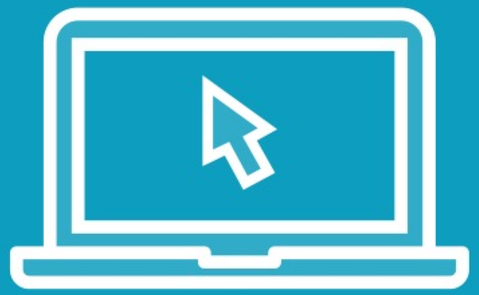


Demo



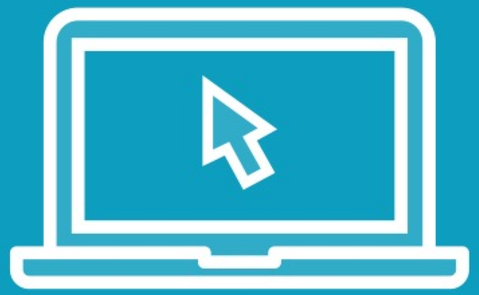
Introducción al patrón repositorio (parte 2)

Demo



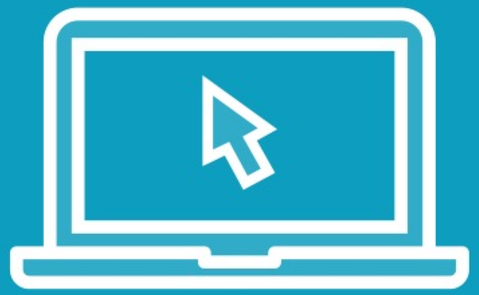
Devolución de datos del repositorio al solicitar recursos (parte 1)

Demo



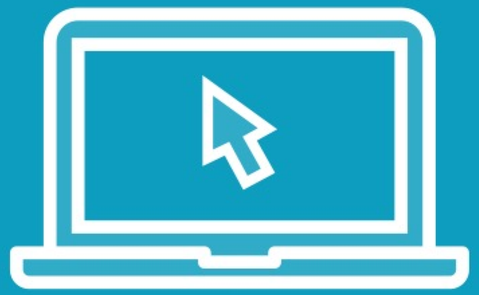
Uso de AutoMapper para mapear entre entidades y DTOs

Demo



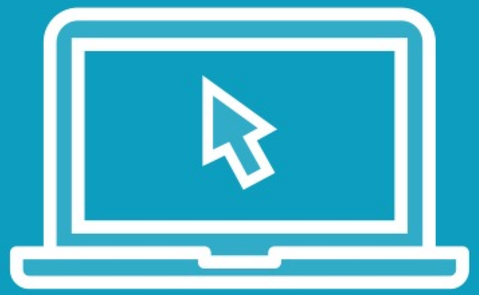
Devolución de datos del repositorio al solicitar recursos (parte 2)

Demo



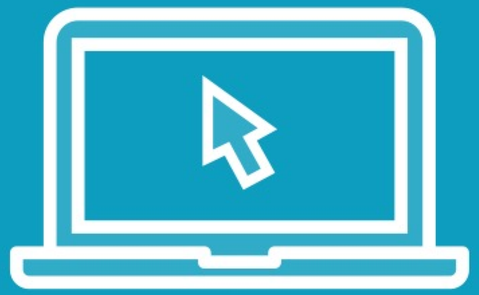
Crear un recurso

Demo



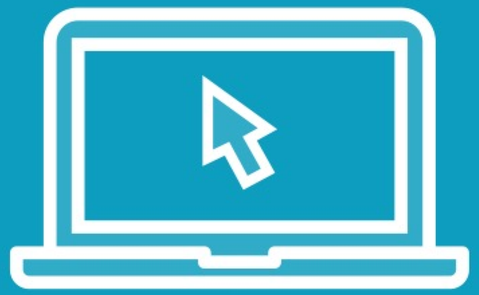
Actualizar un recurso

Demo



Actualizar parcialmente un recurso

Demo



Eliminar un recurso

Resumen

El repositorio es una abstracción que reduce la complejidad y tiene como objetivo hacer que el código por una parte, sea seguro para la implementación del repositorio, e ignore los detalles sobre la persistencia.

Resumen

El uso de código asíncrono para las operaciones de E/S permite liberar los subprocesos más rápidamente, lo que mejora la escalabilidad.

El uso de AutoMapper reduce en gran medida el código de asignación propenso a errores

A continuación:
Búsqueda, filtrado y paginación de recursos