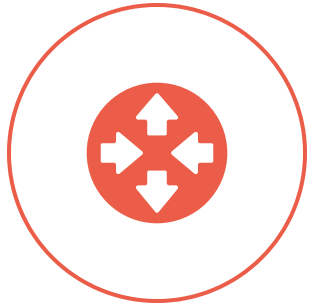
# Adaptadores Primarios y Secundarios

Esta presentación presenta el concepto de adaptadores primarios y secundarios en los sistemas informáticos, incluidos sus roles, funciones e importancia en los entornos informáticos modernos.

# Definición de Adaptadores

### Traducen interacciones

Traducen las interacciones con el usuario o con sistemas externos a comandos del dominio.



### Bidireccional

Los adaptadores traducen comandos del dominio a interacciones comprensibles para usuarios o sistemas externos, y viceversa.

### Desacoplan responsabilidades

Los adaptadores desacoplan las responsabilidades de la lógica del dominio y la interfaz con el usuario o sistemas externos.

Los adaptadores son componentes cruciales que permiten la comunicación entre diferentes partes de un sistema y desacoplan las responsabilidades, lo que facilita el mantenimiento y la escalabilidad del software.

# Adaptadores Primarios

- ## Interfaces de Entrada

  Los adaptadores primarios interactúan directamente con el usuario o sistema a través de interfaces de entrada, como controladores REST que reciben y procesan solicitudes HTTP.

- ## Procesamiento de Solicitudes

  Los adaptadores primarios reciben y procesan las solicitudes entrantes, como solicitudes HTTP, y las transforman en formato adecuado para su uso por el núcleo de la aplicación.

- ## Interacción Directa

  Los adaptadores primarios se encargan de la interacción directa con el usuario o sistema, sirviendo como una interfaz de entrada para que el núcleo de la aplicación pueda recibir y procesar las solicitudes.

# Adaptadores Secundarios

- ## Repositorios de datos
  Interfaces de salida que permiten interactuar con bases de datos relacionales, No SQL, o almacenes de datos para almacenar y recuperar información.

- ## Clientes de servicios externos
  Interfaces de salida que permiten consumir servicios web, APIs REST, o cualquier otro tipo de servicio externo a la aplicación.

- ## Interfaces de archivos
  Interfaces de salida que permiten leer y escribir archivos en el sistema de archivos local o en sistemas de archivos remotos.

- ## Interfaces de mensajería
  Interfaces de salida que permiten enviar y recibir mensajes a través de sistemas de mensajería como colas de mensajes o sistemas de publicación/suscripción.

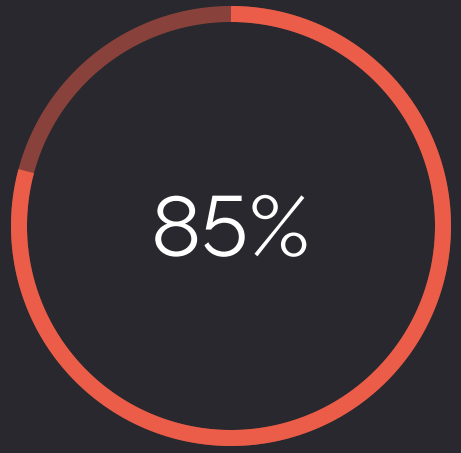
# Diferencias entre Adaptadores Primarios y Secundarios

Valores representados en porcentaje de implementación


Ubicación en la Aplicación


Traducción de Acciones de Usuario


Traducción de Operaciones de Dominio




Interacción con Sistemas Externos


# Ejemplo de Adaptador Primario

El controlador REST maneja las solicitudes de creación de pedidos mediante una interfaz estandarizada y bien definida. Esto facilita la integración con otros sistemas y permite una comunicación eficiente y confiable entre diferentes componentes de la aplicación.


# Ejemplo de Adaptador Secundario

Un repositorio que guarda y recupera entidades de una base de datos es un adaptador secundario que abstrae la lógica de acceso a los datos, permitiendo a la aplicación interactuar con ella de una manera más simple y organizada. Este tipo de adaptador encapsula la complejidad de las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y proporciona una interfaz más limpia y fácil de usar a la lógica de negocio de la aplicación.

# Beneficios de los Adaptadores

## Desacoplamiento de la lógica del dominio

Los adaptadores permiten separar la lógica de negocio de las interfaces de entrada y salida, lo que facilita la gestión y mantenimiento del código.

## Mejora de la modularidad

Al desacoplar los diferentes componentes, la aplicación se vuelve más modular y fácil de modificar, expandir o reemplazar en caso de ser necesario.

## Mejor testabilidad

Al aislar la lógica de negocio de las interfaces, es más sencillo realizar pruebas unitarias de los diferentes componentes de la aplicación.

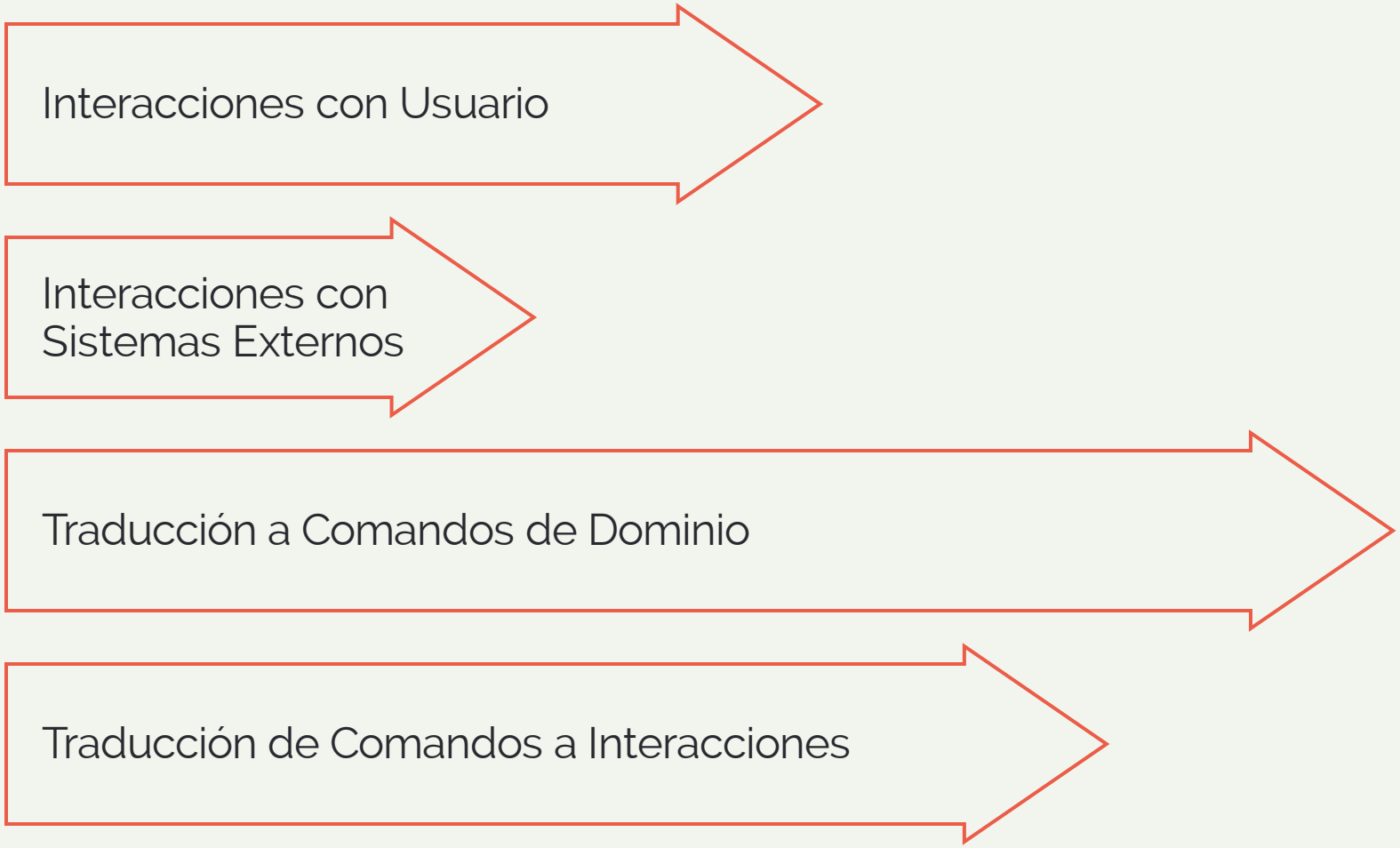
## Flexibilidad en la implementación

Los adaptadores permiten utilizar diferentes implementaciones de las interfaces de entrada y salida sin afectar la lógica del dominio, lo que aumenta la flexibilidad de la aplicación.

## Mejor organización del código

La separación de responsabilidades entre el dominio y las interfaces de entrada y salida ayuda a mantener un código más limpio, legible y fácil de entender.

# Resumen de Adaptadores Primarios y Secundarios



# Adaptadores Primarios y Secundarios

## Definición

El patrón DAO proporciona una abstracción para las operaciones de acceso a datos, permitiendo que el código de la lógica de negocio sea independiente de los detalles de persistencia.

## Responsabilidades

El DAO maneja la creación, lectura, actualización y eliminación de datos en la base de datos.

## Beneficios

Separa la lógica de negocio de la lógica de acceso a datos, facilitando el mantenimiento y la reutilización del código.

## Implementación

Utiliza una interfaz común para acceder a los datos, ocultando los detalles de implementación de la base de datos subyacente.

## Ejemplos

Puede ser utilizado en aplicaciones web, móviles, desktop, microservicios, etc.

# Adaptadores Primarios y Secundarios

## Definición

El patrón DTO se utiliza para transferir datos entre diferentes capas de una aplicación.

## Características

Los DTOs son objetos simples que solo contienen datos y no tienen lógica de negocio.

## Objetivo

Se utilizan para optimizar la comunicación entre capas, especialmente entre la capa de presentación y la capa de servicio.

## Beneficios

Permiten reducir la complejidad y mejorar el rendimiento de la aplicación al transferir solo los datos necesarios entre capas.

## Ejemplo de Uso

Un DTO puede utilizarse para transferir información de un usuario desde la capa de presentación a la capa de servicio sin exponer detalles internos de la implementación.

# Adaptadores Primarios y Secundarios

Los adaptadores de servicio son una capa intermedia que traducen las solicitudes externas en comandos del dominio y devuelven las respuestas del dominio al formato esperado por el cliente. Estos adaptadores convierten las llamadas externas (entrantes) en operaciones del dominio, actuando como un puente entre el mundo exterior y la lógica interna de la aplicación.

# Ejercicio Práctico con Java y Quarkus

- ## Controlador REST para Órdenes (Adaptador Primario)

  Implementación de un controlador REST que expone endpoints para crear, leer, actualizar y eliminar órdenes. Este controlador actúa como un adaptador primario, recibiendo las solicitudes del cliente y delegando la lógica de negocio al servicio del dominio.

- ## DAO para Acceso a Datos (Adaptador Secundario)

  Implementación de un DAO (Data Access Object) que abstrae el acceso a la capa de persistencia de datos. Este DAO actúa como un adaptador secundario, proporcionando una interfaz estandarizada para que el servicio del dominio pueda interactuar con la base de datos.

- ## Uso de DTOs para Transferencia de Datos

  Implementación de DTOs (Data Transfer Objects) para transferir datos entre las diferentes capas de la aplicación. Los DTOs encapsulan la información necesaria para cada operación, evitando exponer detalles internos de la entidad de dominio.

- ## Integración del Adaptador Primario con el Servicio del Dominio

  Integración del controlador REST (adaptador primario) con el servicio de dominio responsable de la lógica de negocio. El controlador recibe las solicitudes del cliente, las valida y las envía al servicio del dominio para su procesamiento. El servicio del dominio, a su vez, utiliza el adaptador secundario (DAO) para acceder a la capa de persistencia.