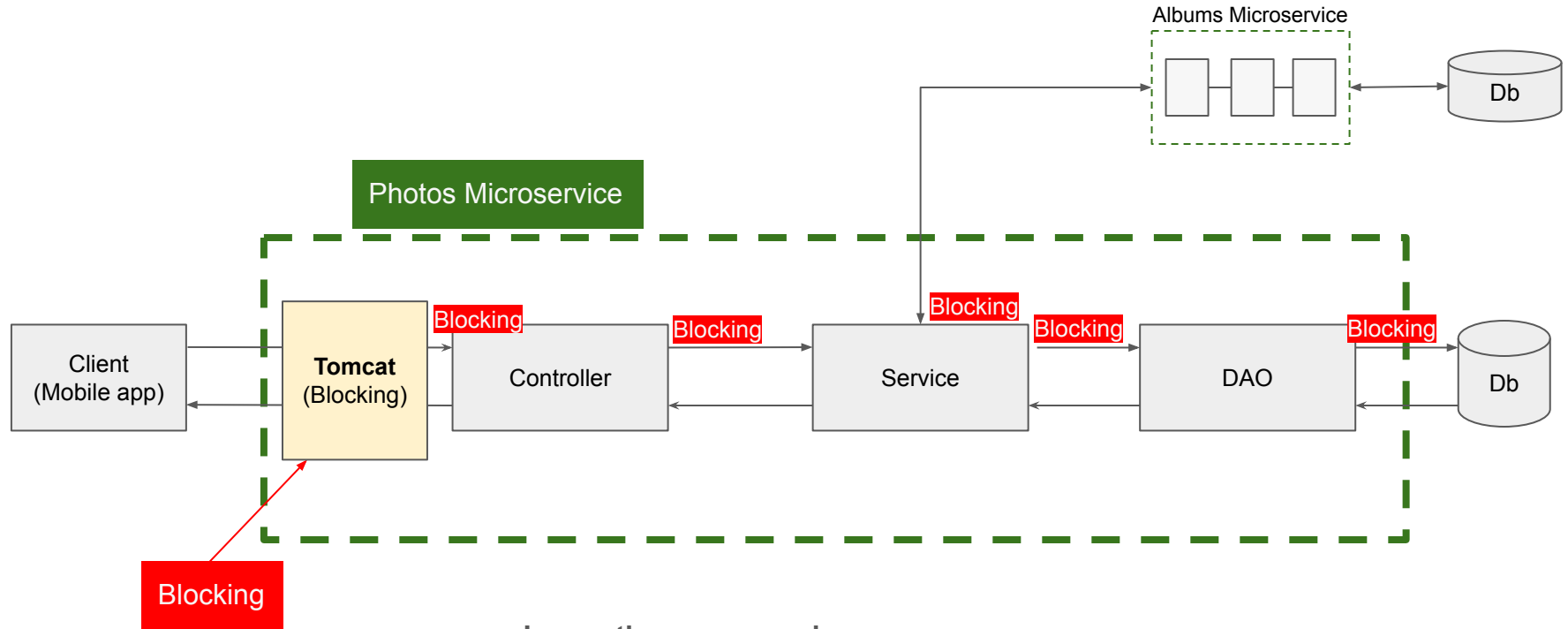


# Reactive Application

## Overview

# Traditional(Blocking) Spring MVC REST application



## Imperative programming:

- Sequential execution of code,
- Blocking I/O operations,
- Synchronous, step-by-step logic.

The diagram illustrates a non-blocking microservice architecture for a photo application. It features a central **Photos Microservice** (enclosed in a dashed green box) and an external **Albums Microservice**.

**Photos Microservice Components:**

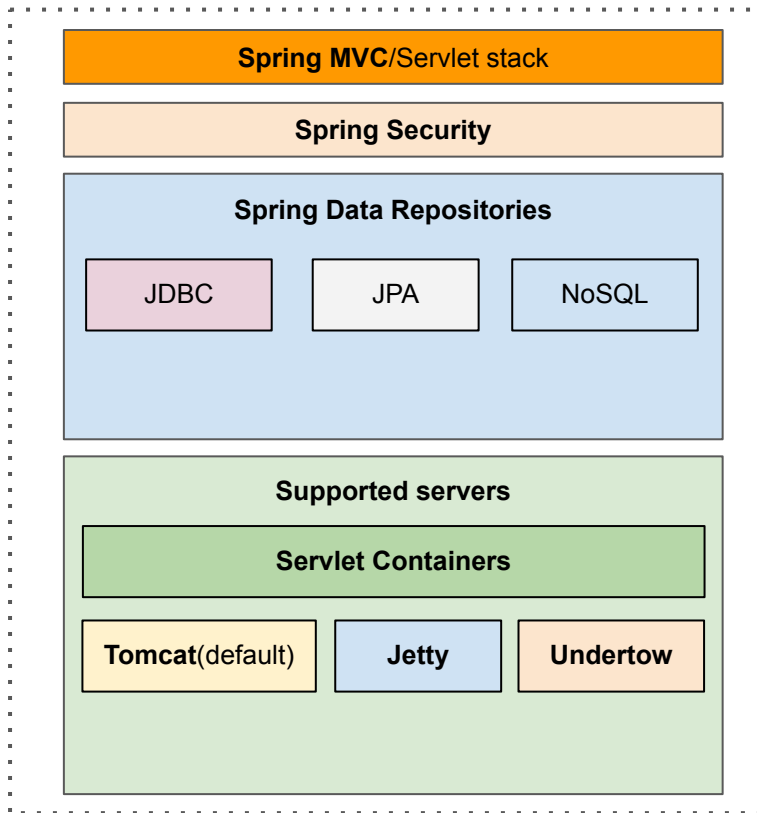
- Client (Mobile app):** Interacts with the **Netty Non Blocking** component.
- Netty Non Blocking:** A yellow box representing the non-blocking network layer.
- Controller:** Receives requests from the Netty layer and forwards them to the **Service** layer.
- Service:** The core business logic layer, which interacts with the **DAO** layer.
- DAO (Data Access Object):** Manages data persistence, interacting with the **Db**.
- Db (Database):** The data storage component.

**Albums Microservice:** A separate service (enclosed in a dashed green box) that interacts with its own **Db**. It is connected to the **Service** layer of the Photos Microservice via a **Non-blocking** interface.

**Non-blocking Nature:** The diagram emphasizes non-blocking communication throughout the system. Green boxes labeled **Non-blocking** point to the Netty layer, the Service layer, the DAO layer, and the external Albums Microservice. Green labels **Non-blocking** are placed on the communication links between the Netty layer, Controller, Service, and DAO, as well as between the Service layer and the Albums Microservice.

- Asynchronous data streams,
- Non-blocking I/O operations,
- Event-driven architecture.

## Traditional(Blocking) application



## Reactive(Non-blocking) application

