PATRONES DE ARQUITECTURA DE SOFTWARE

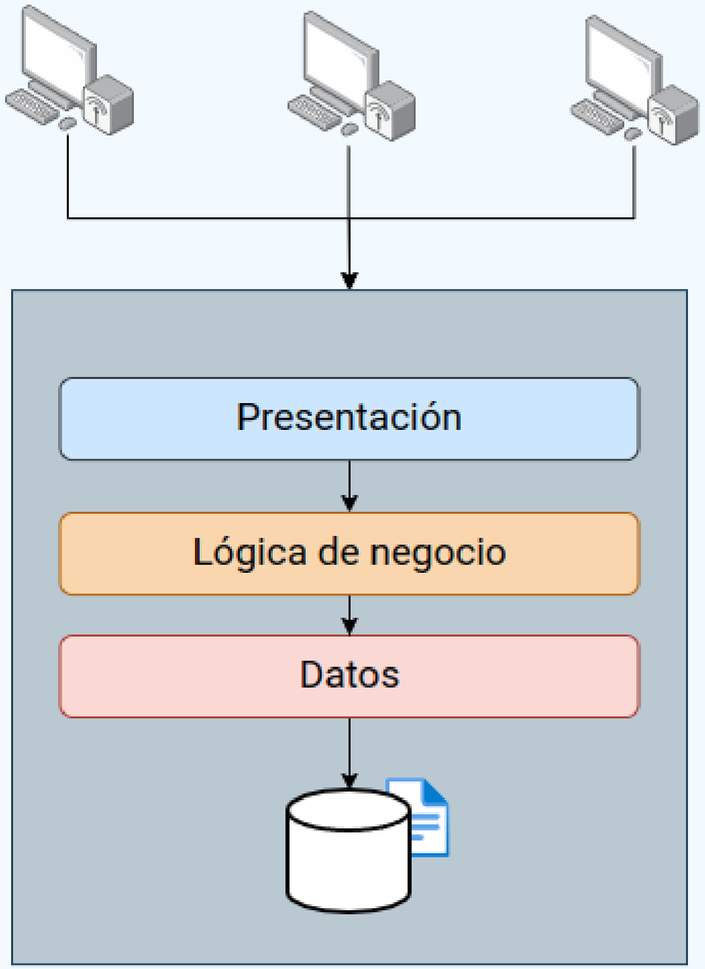
ARQUITECTURAS MULTI-LAYER Y MULTI-TIER

Daniel Blanco Calviño

ARQUITECTURA MULTI-LAYER

- Todo el código del sistema en un **único componente desplegable.**

- **Dividido en capas** lógicas. Habitual tres capas:
  - Capa de **presentación**.
  - Capa de **lógica de negocio.**
  - Capa de **acceso a datos.**



ASPECTOS POSITIVOS

- **Menor complejidad**, arquitectura muy sencilla.
- Mayor velocidad de desarrollo al inicio del proyecto.
- Cualquier desarrollador sin experiencia se puede adaptar rápidamente.
- Puede ser más fácil de testear que otras arquitecturas.
- **Compilación y despliegue** sencillos.

ASPECTOS NEGATIVOS 

- **Difícil de mantener** a largo plazo.
- Tendencia a una **gran dependencia entre los distintos componentes**.
- **Arquitectura rígida** y difícil de modificar en el futuro.
- Mayor dificultad a la hora de **repartir el trabajo**.
- Si queremos actualizar el sistema, debemos desplegarlo de nuevo completamente.

CUÁNDO USAR

- **Proyectos pequeños**, con pocos requisitos y claramente definidos.
- Sistemas con un **corto tiempo de vida.**
- Equipo con muy **poca experiencia.**

ARQUITECTURA MULTI-TIER

- **También está dividido en niveles**. No está limitado, pero lo habitual son los mismos tres niveles.

- Cada nivel representa un **componente desplegable independiente.**
  - Cada uno se puede compilar y desplegar de forma independiente al resto.
  - Mitigamos las desventajas del patrón multi-layer.
  - Podemos **escalar de forma independiente** cada nivel.

- Más fácil de trabajar con este patrón.
  - Respositorios independientes, podemos asignar un equipo a cada nivel.

- Desventaja importante: El nivel de lógica de negocio suele ser mucho más pesado. **Riesgo de monolito** en ese nivel.