INTRODUCCIÓN

ARQUITECTURA CONVENCIONAL

Daniel Blanco Calviño

POCOS REQUISITOS

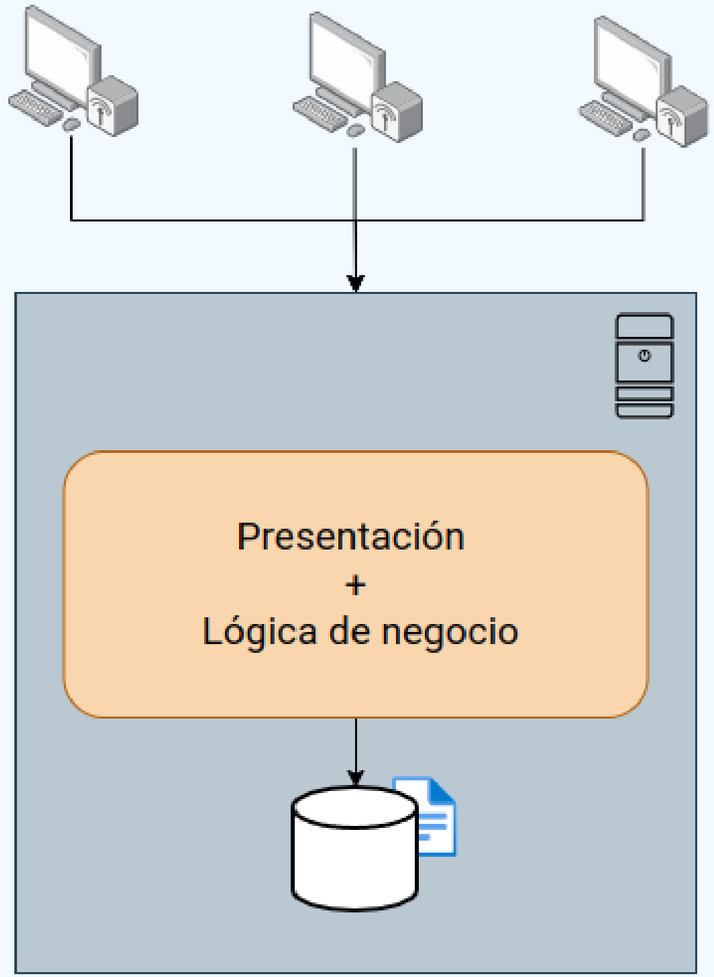
Hace 50 o 60 años la mayoría del software era **mucho más sencillo que hoy en día.**

- Necesidad de automatizar tareas.
- CRUDs simples.
- etc.

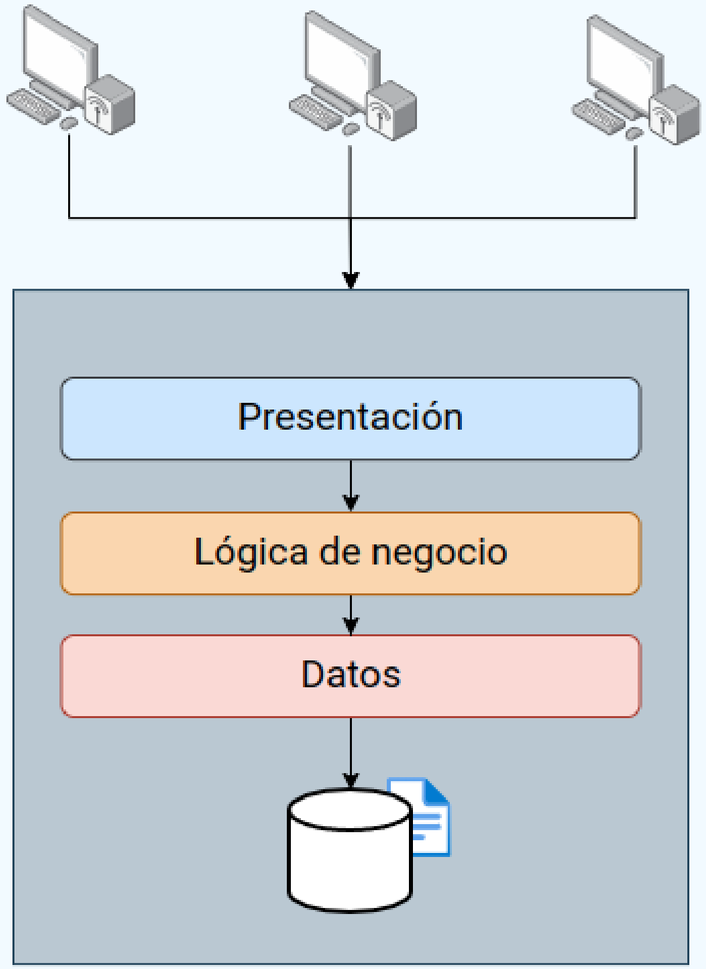
Esto llevó a software del siguiente tipo.

- Pequeños scripts de automatización.
- **Monolitos**.
- Muchas interdependencias entre los componentes internos.
- Código desestructurado y difícil de mantener.

MONOLITOS



MONOLITOS - CAPAS DEL SOFTWARE



MODELO ANÉMICO

- Clases que no almacenan lógica, **únicamente datos.**
- Muy comunes en **CRUDs.**
- Lógica necesaria se implementa en servicios.
- Considerado un **antipatrón de diseño**

```
3   public class AnemicBox {
4     private double width;
5     private double height;
6     private double depth;
7
8     // Getters and Setters
9   }
```

DEPENDENCIAS ENTRE COMPONENTES

En monolitos y arquitecturas convencionales es muy común tener **componentes muy acoplados.** El siguiente escenario es muy común:

- Componente A necesita un componente Z. Lo implementamos.
- Surge una necesidad de ciertas funcionalidades de Z en el componente B.
- Pasa lo mismo en C, D...
- Con el tiempo, **las necesidades evolucionan**, y **se adapta Z** para que abarque todo lo que necesiten A, B, C y D.

 ¡Este tipo de decisiones hacen que nuestra **arquitectura** sea **rígida y difícil de modificar!**

INTRODUCCIÓN

ARQUITECTURA CONVENCIONAL

Daniel Blanco Calviño