Arquitectura de N capas

La arquitectura en capas es un modelo de diseño de software, cuya base es la separación de las diferentes funcionalidades del sistema en capas o niveles, donde cada capa se encarga de un conjunto de tareas específicas y se comunica con los niveles adyacentes mediante interfaces bien definidas.

Cada capa se construye sobre la inferior inmediata, la cual proporciona los servicios y la funcionalidad necesarios para que la capa superior pueda funcionar correctamente. Este modelo permite una**mejor organización y modularidad del sistema**; además facilita su mantenimiento y evolución a largo plazo.

Utilizamos para este sistema las siguientes capas:

Acceso Datos: Esta capa se encarga de la interacción con la base de datos. Realiza operaciones como:

* Consultas (Select)
* Inserciones (Insert)
* Actualizaciones (Update)
* Eliminaciones (Delete)

API: Define los endpoints y métodos que permiten a otros sistemas (como aplicaciones web o móviles) interactuar con tu aplicación. Actúa como un intermediario entre el cliente y la lógica de negocio.

Controladora: Recibe las peticiones HTTP y las traducen a acciones en la lógica de negocio. Son responsables de la gestión de la respuesta HTTP al cliente.

Entidades: Las entidades representan las tablas y los datos que se almacenan en la base de datos.

Lógica Negocio: La lógica de negocio contiene las reglas de negocio, validaciones y procesos específicos de la aplicación. Realiza la mayor parte del trabajo relacionado con la lógica de la aplicación y coordina la interacción con otras capas.

**Ventajas**

Modularidad: facilita el desarrollo, la prueba y el mantenimiento de cada capa de forma independiente.

Separación de responsabilidades: permite la separación de responsabilidades y una mayor claridad en la arquitectura, ya que cada capa tiene una función específica.

Reutilización: posibilita que las capas se reutilicen en diferentes proyectos, lo que reduce el tiempo y los costos de desarrollo.

Escalabilidad: significa que se pueden agregar o quitar capas según las necesidades del sistema.

Flexibilidad: es adaptable a diferentes entornos y requisitos del sistema.

Es recomendada cuando el equipo no cuenta con mucha experiencia.

**Desventajas**

Complejidad: aumenta la complejidad del sistema debido a la necesidad de interfaces y comunicación entre las capas.

Tiempo y esfuerzo de desarrollo: requiere más tiempo y esfuerzo de desarrollo debido a la necesidad de diseñar, desarrollar y probar cada capa por separado.

Consumo de recursos: aumenta el consumo de recursos del sistema, como el uso de memoria y CPU, debido a la necesidad de comunicación entre las capas.

Costos: eleva los costos del desarrollo debido a la necesidad de diseñar y desarrollar varias capas separadas.

Rendimiento: disminuye el rendimiento del sistema debido a la necesidad de comunicación entre las capas.

**Dependencias:**

**Acceso Datos <= Entidades**

**Lógica Negocio <= Acceso Datos**

**Lógica Negocio <= Entidades**

**Controladora <= Acceso Datos**

**Controladora <= Lógica Negocio**

**API <= Lógica Negocio**

**API <= Entidades**

**API <= Controladora**