🧩 MODELO DE CAPAS – Explicación Completa (Markdown)

```
# 🏛️ MODELO DE CAPAS (Layered Architecture)

El **Modelo de Capas** es una arquitectura donde la aplicación se divide en módulos ordenados llamados **capas**, donde cada capa tiene una responsabilidad específica y solo se comunica con la capa inmediatamente inferior.

Es la arquitectura más usada en:
- aplicaciones web
- aplicaciones empresariales
- APIs REST
- Android
- Java, .NET, PHP, Node

---

# 🧱 ESTRUCTURA TÍPICA (4 capas)

1️⃣ **Capa de Presentación (UI / Frontend)**
2️⃣ **Capa de Lógica de Negocio (Servicios / Controllers)**
3️⃣ **Capa de Acceso a Datos (DAO / Repository)**
4️⃣ **Capa de Datos (Base de Datos)**

La comunicación es así:



UI → Lógica → Acceso a Datos → BD


# 1️⃣ CAPA DE PRESENTACIÓN (UI)

Es la interfaz que usa el usuario.
Muestra datos y recibe acciones.

Incluye:
- XML en Android
- HTML / CSS / JS
- React, Vue, Angular
- Views de MVC

### ❌ No debe contener lógica de negocio.
### ✔ Solo muestra y envía datos.

### Ejemplo:
- Pantalla “Agregar Película”
```

```
- Layout de login
- Página del carrito de compras

---

# 2️⃣ CAPA DE LÓGICA DE NEGOCIO (Servicios / Controllers)

Aquí vive la **inteligencia del sistema**:

- Reglas de negocio
- Validaciones complejas
- Coordinación de procesos
- Llamadas a la capa de datos

Ejemplos:
- “Si el usuario no está logeado, no puede agregar a favoritos”
- “Calcular total de compra”
- “Verificar stock antes de crear pedido”

---

# 3️⃣ CAPA DE ACCESO A DATOS (DAO / Repository)

Es la capa que habla DIRECTO con la base de datos.

Responsabilidades:
- consultas SQL
- inserciones / updates
- traer documentos Firestore
- manipular colecciones

Ejemplos:
- `PeliculaRepository.getAll()`
- `PedidoDAO.insert(pedido)`
- Firestore: `collection("peliculas").document(id)`

---

# 4️⃣ CAPA DE DATOS (BD)

Es donde realmente se almacena la información:

- Firebase Firestore
- MySQL / PostgreSQL
- Oracle
- MongoDB
- SQL Server

---

# 🧠 EJEMPLO REAL 1: E-commerce (pregunta clásica)

## Capa 1 – Presentación
```

```
- HTML: catálogo, carrito, checkout

## Capa 2 – Lógica
- PedidoService
- CarritoService

## Capa 3 – Acceso a Datos
- ProductoDAO
- ClienteRepository

## Capa 4 – Base de Datos
- Tabla productos
- Tabla carrito
- Tabla pedidos

---

# 🧠 EJEMPLO REAL 2: Tu App WatchMe (Android)

## Capa 1 – Presentación
- XML del Home
- XML del Detalle de Película

## Capa 2 – Lógica
- GalleryViewModel
- FavoriteManager
- Validaciones

## Capa 3 – Acceso a Datos
- Firestore queries
- `peliculasCollection.get()`

## Capa 4 – BD
- Firestore (colecciones)
- Documentos: usuarios, favoritos, vistos

---

# 🧠 EJEMPLO REAL 3: API REST (Node, PHP, Java)

## Capa 1 – Presentación
- JSON que ve el cliente (la “vista” en una API)

## Capa 2 – Lógica
- Controllers
- Services
- Validaciones

## Capa 3 – Acceso a Datos
- Query SQL
- Repositorios

## Capa 4 – BD
```

- MySQL / Mongo

🧱 BENEFICIOS DEL MODELO DE CAPAS

✔ Código ordenado
✔ Fácil de mantener
✔ Permite reemplazar una capa sin afectar las otras
✔ Separación clara de responsabilidades
✔ Escalable
✔ Reciclaje de código

❌ DESVENTAJAS

- Puede volverse lento si hay muchas capas
- A veces hay demasiada abstracción
- No sirve bien para sistemas en tiempo real
- No es ideal para microservicios

📌 RESUMEN PARA LA PRUEBA

Capa 1 (Presentación): muestra datos
Capa 2 (Lógica): reglas del negocio
Capa 3 (Acceso a Datos): consultas BD
Capa 4 (BD): almacenamiento

Comunicación siempre es:

UI → Lógica → Datos → BD

Nunca UI → BD directo.