¡Sí! 🔥 Lo que resumiste ahí es **el patrón de diseño más común en Angular** hoy en día.

Te lo detallo:

**📌 Patrón más usado en Angular**

* **Model** → DTOs, interfaces, tipos → (carpeta models/)
* **View** → plantillas HTML + estilos CSS/SCSS → (carpeta features/\* en cada módulo)
* **ViewModel (Controller adaptado)** →
  + **component.ts** → maneja el estado y la interacción con la vista.
  + **services** → encapsulan lógica y peticiones HTTP.

Eso es exactamente el patrón **MVVM (Model–View–ViewModel)**.  
En la práctica, **Angular no sigue MVC clásico**, pero sí aplica MVVM de forma natural.

**📂 Organización típica en proyectos reales**

La mayoría de proyectos medianos/grandes en Angular usan algo así:

src/app/

core/ <-- servicios globales, interceptores, guards

shared/ <-- componentes, pipes, directivas reutilizables

features/ <-- módulos de negocio (cada módulo = vista + lógica asociada)

animals/

animals.module.ts

animals.component.ts

animals.component.html

animals.service.ts

models/ <-- DTOs, interfaces

**✅ Conclusión**

* Lo que escribiste (Model = models/, View = features/\*, ViewModel = component + service) es **la forma más común de estructurar Angular**.
* El patrón que realmente describe Angular es **MVVM**, no MVC.
* En la industria, esta separación (core/, shared/, features/, models/) es considerada **buena práctica** y está alineada con la guía de estilos oficial de Angular.

¿Quieres que te arme un **mapa visual** (tipo diagrama) de cómo se relacionan core, shared, features y models dentro del patrón MVVM en Angular?