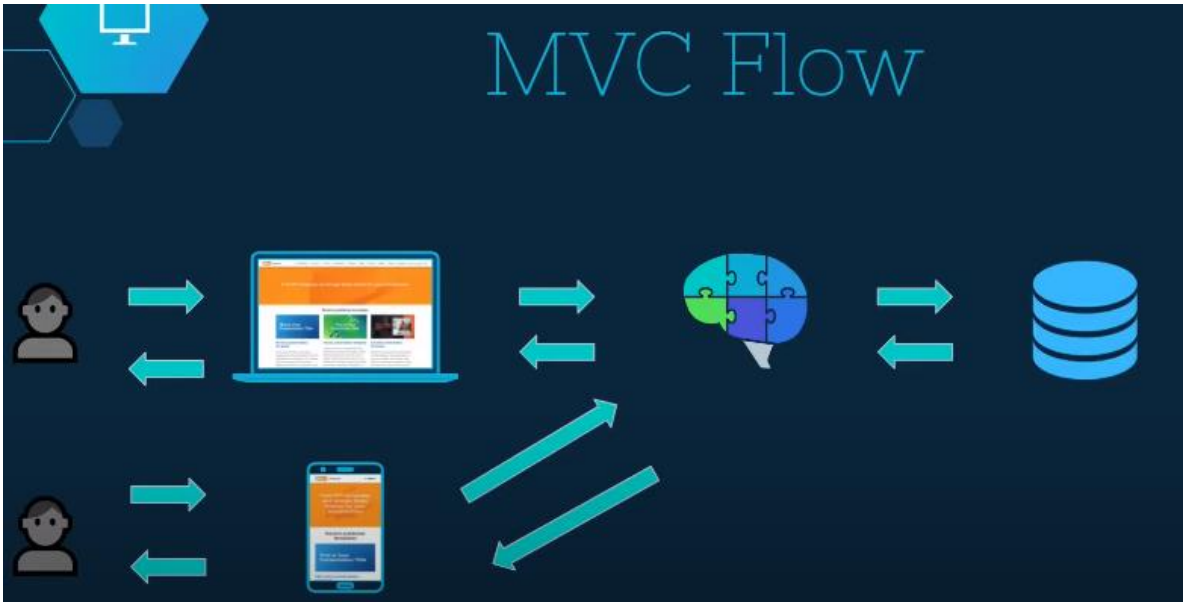
1. **MVC (Modelo-Vista-Controlador):**
    - **Ventajas:**
        - Separación clara de responsabilidades entre el modelo, la vista y el controlador.
        - Facilita la reutilización del código y la escalabilidad de las aplicaciones.
        - Es un patrón ampliamente utilizado y comprendido.
    - **Desventajas:**
        - En aplicaciones grandes y complejas, puede llevar a un número excesivo de controladores y vistas, volviéndose difícil de gestionar.
2. **Arquitectura Hexagonal (Hexagonal Architecture):**
    - **Ventajas:**
        - Centra el diseño en los casos de uso y en las interacciones del sistema con el mundo exterior, lo que facilita las pruebas unitarias y la sustitución de componentes.
        - Desacopla el núcleo de la aplicación de los detalles de implementación como bases de datos y frameworks.
    - **Desventajas:**
        - Puede parecer complejo para aplicaciones pequeñas o sencillas.
        - Requiere un buen entendimiento del dominio y de los casos de uso para ser implementado correctamente.
3. **Clean Architecture:**
    - **Ventajas:**
        - Establece capas claras de abstracción, con el núcleo (core) de la aplicación en el centro y las capas externas (como interfaces de usuario y bases de datos) dependiendo del núcleo.
        - Fomenta la independencia de las herramientas y frameworks.
        - Facilita las pruebas y la modificación del código sin afectar otras partes del sistema.
    - **Desventajas:**
        - Puede parecer complejo y excesivo para aplicaciones pequeñas o proyectos con requisitos sencillos

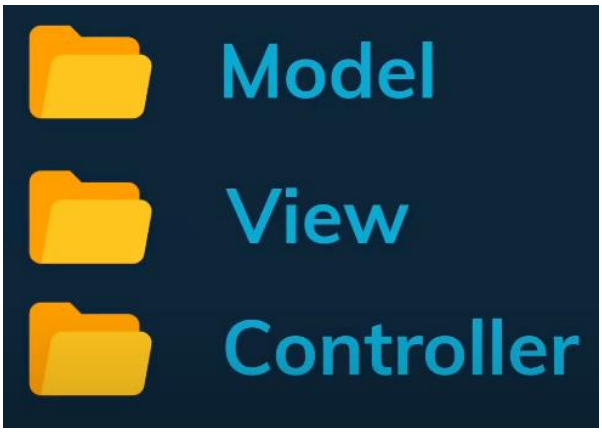
CONCLUSION

MVC (Model View Controller)**:**

Es un patrón de diseño de software, además es el mas usado en el desarrollo webno se limita a un solo lenguaje de programación, ella esta dividida en 3 grandes capas:

- **Model** es la que se encarga de administrar y guardar datos, o sea la base de datos.
- **View** interfaz grafica de usuario, es la que se encarga de la interaccion con el usuario mediante la representación visual
- **Controller** es la que se encarga de traducir lo que se ejecuta en view con el modelo, por ejemplo alguna update de datos





Hace mucho más fácil el entendimiento de código, es una manera estandarizado de organizar nuestros folders

Ejemplo en código:

MODEL

```
public class Person
{
    public int PersonId { get; set; }

    [Required]
    [MinLength(2)]
    public string Name { get; set; }

    [Phone]
    public string PhoneNumber { get; set; }

    [EmailAddress]
    public string Email { get; set; }
}
```

VIEW

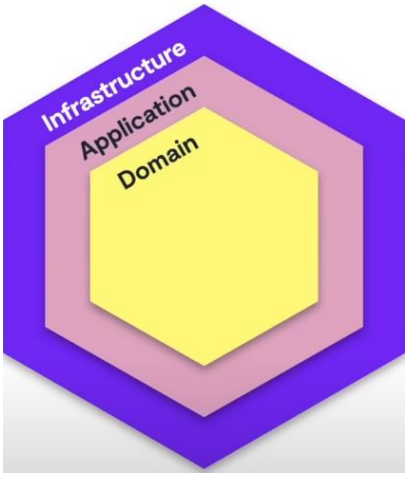
CONTROLLER

```
public class PeopleController : Controller
{
    private readonly AddressBookContext _context;

    public PeopleController(AddressBookContext context)
    {
        _context = context;
    }

    // GET: /people
    public async Task Index()
    {
        return View(await _context.People.ToListAsync());
    }

    // GET: /people/details/5
    public async Task Details(int id)
    {
        var person = await _context.People.Find(id);

        if (person == null)
        {
            return NotFound();
        }

        return View(person);
    }
}
```

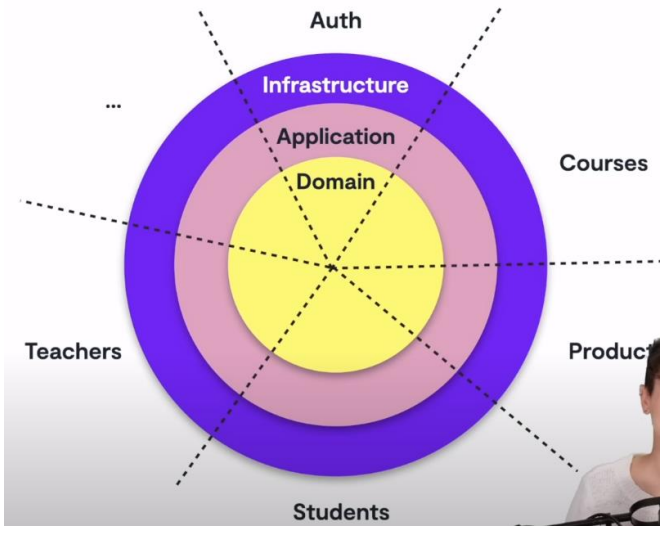
ARCHITECTURE HEXAGONAL

Nos ayuda a tener un código mantenible escalable y testable, esta en globado como una clean arquitectura, cuenta con 3 capas, la arquitectura va con una regla de dependencia que va de afuera hacia dentro:

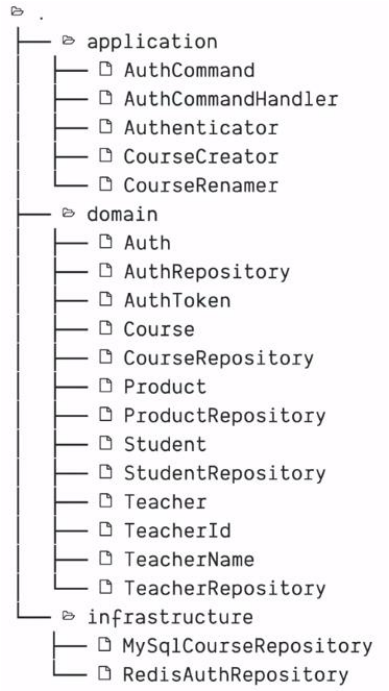
- Capa de infraestructura: Aquí es donde va la conexión de base de datos y cualquier cosa de entrada y salida, puede conocer detalles de application y domain
- Capa application: Aquí dentro están los casos de usos login y log out, solo puede conocer detalles de domain pero no de infraestructure
- Capa de domain: La mas importante conceptos como usuario, userId, cualquier valor que puede haber, aca están las interfaces de los repositorios, solo puede conocer detalles de el mismo, o sea no tiene detalles de de infraestructura ni de application



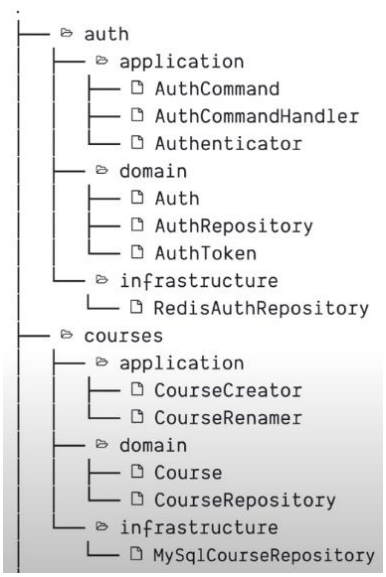
Se puede aplicar vertical slicing, para un mejor manejo de la arquitectura



Folders sin vertical slicing



Folders con vertical slicing, en si este método sirve para organizar de mejor manera los archivos y que sea más entendible



LINK AYUDA DE UNA ARQUITECTURA HEXAGONAL: https://carlos.lat/blog/hexagonal-architecture-using-go-fiber/