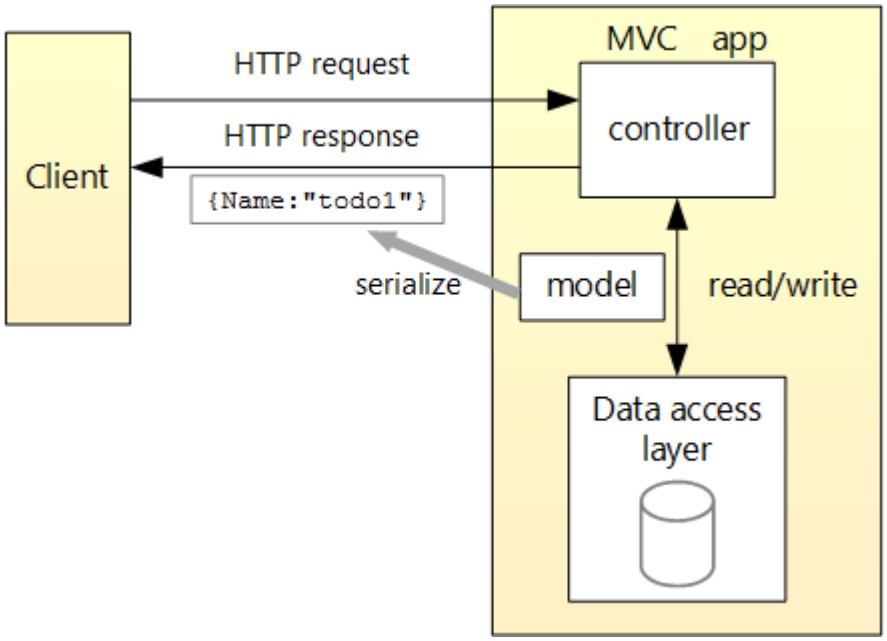
# Informe

### a. Estructura del Proyecto Back

Se creó un proyecto de API en ASP.NET Core Web API utilizando la plantilla básica.

Utilice una arquitectura en capas, separando:

> **Capa de datos (models)**: Aquí definimos las entidades de turnos y sucursales, además del contexto de la base de datos usando Entity Framework Core.
>
> **Capa de servicios** : Aquí se manejará la lógica de negocio, como la creación, gestión y validación de turnos.
>
> **Capa de controladores)**: Aquí se exponen los endpoints REST para interactuar con los turnos.




### b. Estructura del Proyecto Front

Se creó un proyecto en Angular en el cual utilice una arquitectura por componentes

### c. Estructura de base de datos

Cree las siguientes 3 tablas con sus respectivas relaciones

```
CREATE TABLE [User] (
    IdUser INT PRIMARY KEY IDENTITY(1,1),
        DocumentNumber NVARCHAR(50) NOT NULL,
```

```
    Username NVARCHAR(50) NOT NULL,
    Password NVARCHAR(50) NOT NULL
);

CREATE TABLE Branch (
    IdBranch INT PRIMARY KEY IDENTITY(1,1),
        Name NVARCHAR(50) NOT NULL,
    Address NVARCHAR(50) NOT NULL
);

CREATE TABLE [Shift] (
    IdShift INT PRIMARY KEY IDENTITY(1,1),
    IdUser INT NULL,
    IdBranch INT NOT NULL,
    ScheduledDateTime DATETIME NOT NULL,
    ExpirationTime DATETIME NOT NULL,
    IsActive BIT NOT NULL,
        DateAssociation DATETIME NULL,
    FOREIGN KEY (IdUser) REFERENCES [User](IdUser),
    FOREIGN KEY (IdBranch) REFERENCES Branch(IdBranch)
);
GO
```

**d. Decisiones**

primero se ingresa a través de un login para verificar que el usuario está autorizado, posterior a eso se encuentra con una vista compuesta por dos tablas: una que le muestra los turnos disponibles y debajo una tabla con los turnos que ya tiene asignados, la primera tabla cuenta con opción de ‘agendar’ y la segunda tabla cuenta con opción de ‘activar’ entre tanto se tuvieron en cuenta cada una de las reglas descritas en la prueba técnica.