



Curso de **Fundamentos de Entity Framework**

Miguel Teherán



Prerrequisitos

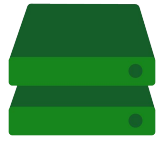


Prerrequisitos

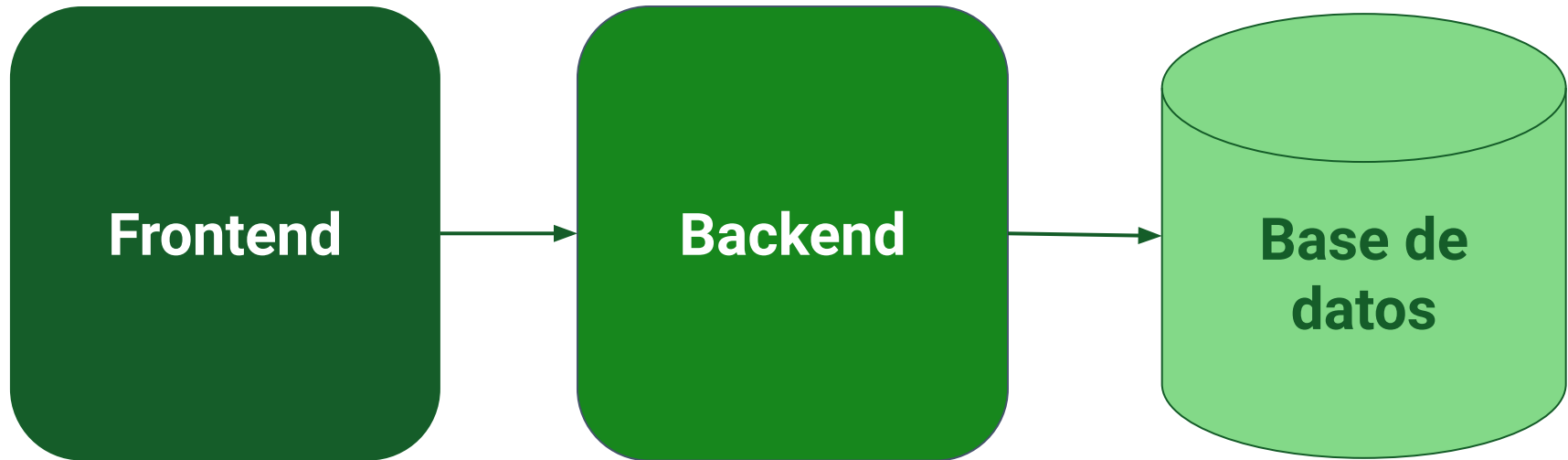
- **Conocimientos y experiencia en C#.**
- **Conocimientos de bases de datos.**
- **SDK de .NET.**
- **Visual Studio Code.**
- **Postman.**
- **SQL server / PostgreSQL.**
- **SQL management.**



Conexión a base de datos



Arquitectura básica





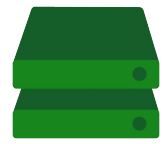
Tipos de conexión

- **ODBC**
- **OLEDB**
- **SQL server**
- **Conexión Azure SQL**

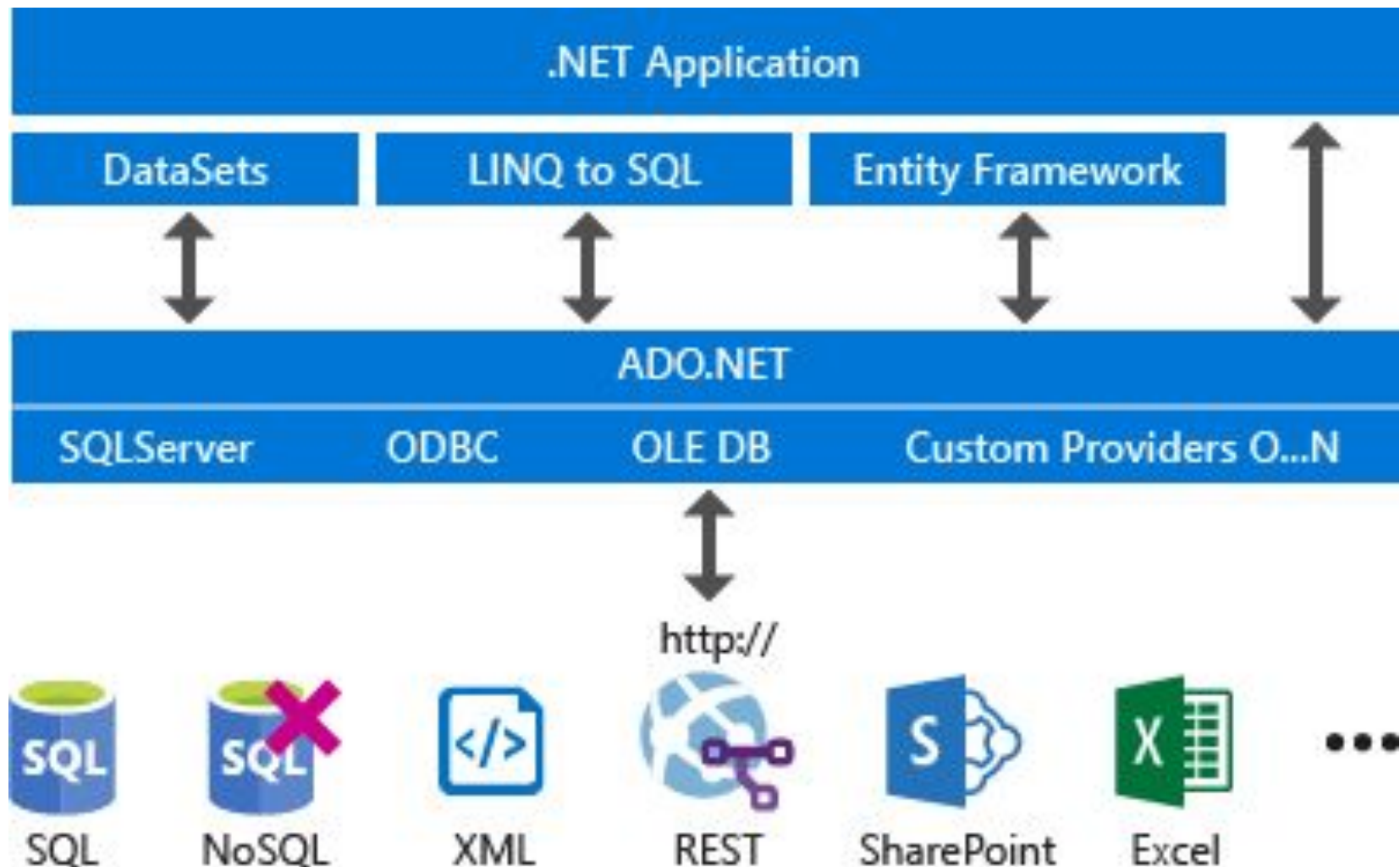


ADO.NET
***Conjunto de librerías
para acceder a datos y
servicios de datos.***





Funcionamiento de ADO.NET





¿Qué es un ORM?



Desafíos en la conexión a una base de datos

- Mantenimiento del esquema y datos.
- Creación de consultas usando SQL.
- Transformación de datos para ser usados en el backend.
- Garantizar la seguridad al manipular datos.



ORM

Object-Relational

Mapping

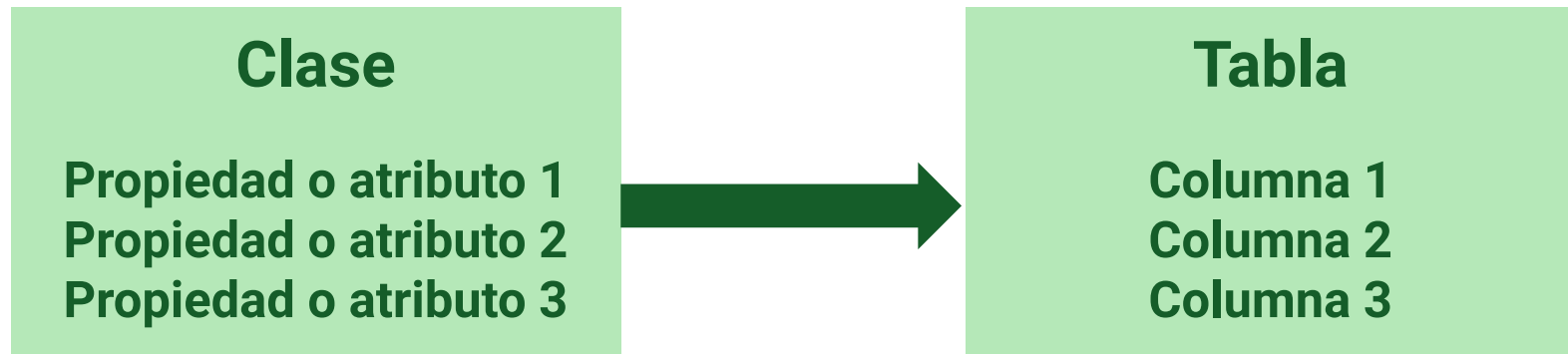




ORM

Modelado de bases de datos relacional usando programación orientada a objetos.

Reemplaza a SQL como lenguaje de consultas utilizando funciones.





ORM populares

- Hibernate
- Dapper
- NHibernate
- Django ORM



Introducción a **Entity Framework**



Entity Framework (EF)

- ORM de código abierto para .NET
- Utiliza ADO.NET para facilitar la conexión a bases de datos y la manipulación de datos.

EF es compatible con: SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, y otras bases de datos utilizando un API plugin.



Entity Framework **vs.** ***Entity Framework Core***





Ventajas de EF

- Mejora la velocidad de desarrollo.
- Permite manejar un solo repositorio para backend y base de datos.
- Mejora la seguridad.
- Permite programar de manera más amigable y fácil.
- Nos permite controlar el historial de cambios de la base de datos de manera muy sencilla.



Proyecto en .NET con Entity Framework



Proyecto .NET usando Minimal API

Modelo - Tarea

Tareald
Categoriald
Titulo
Descripcion
Prioridad
Fecha_Creacion

Modelo - Categoria

Categoriald
Nombre
Descripcion



Creación de modelos



Configuración de Entity Framework



Mapeo de modelos usando atributos



Utilizando base de
datos en memoria



Conectado a base de
datos SQL server



Agregando conexión al
archivo appsettings



Introducción a Fluent API



Fluent API

Es una forma avanzada de configurar los modelos de Entity Framework sin utilizar atributos o data-annotations, permitiendo diseñar la base de datos considerando aspectos avanzados.


Se usan funciones de extensión anidadas para configurar tablas, columnas y especificar el mapeo de los datos.



Ejemplo

```
builder.Entity<Client>(entity =>
{
    entity.ToTable("Client");
    entity.HasKey(e => e.PersonId);
    entity.Property(e => e.ContactName).IsRequired(false).HasMaxLength(300);
    entity.Property(e => e.ContactPhone).IsRequired(false).HasMaxLength(50);
    entity.Property(e => e.Phone2).IsRequired(false).HasMaxLength(50);

    entity.HasOne(e => e.Person)
        .WithOne(e => e.Client)
        .HasForeignKey<Client>(b => b.PersonId)
        .HasConstraintName("FK_Client_Person")
        .OnDelete(DeleteBehavior.SetNull);
});
```




Creando modelo de categoría con Fluent API



Creando modelo de tarea con Fluent API



Conectado a base de
datos con **Fluent API**



¿Qué son las
migraciones?



Migraciones

Es una funcionalidad de Entity Framework que nos permite guardar de manera incremental los cambios realizados en la base de datos.

Nos permite construir un versionamiento de la base de datos.



Comandos básicos

- `dotnet ef migrations add InitialCreate`
- `dotnet ef migrations add MyMigration`
- `dotnet ef database update`





**Inicializar las
migraciones**



Creando una migración



**Agregando datos
semilla**



Obteniendo datos con **Entity Framework**



Guardando datos
con **Entity**
framework



Actualizando datos con **Entity Framework**



Eliminando datos con **Entity Framework**



Continúa tu camino
desarrollando