

Definir una nueva clase en la carpeta Data, MiContexto para cada base de datos. Diferentes contextos de datos que hereda de DbContext.

```
public class MiContexto : DbContext
```

Definimos un constructor con parámetro de opciones de configuración y se la paso a la clase superior con :base(options)

```
{
    0 references | 0 changes | 0 authors, 0 changes
    public MiContexto(DbContextOptions options) :base(options)
    {
    }
}
```

Luego nos vamos a la clase Startup.cs y en el método ConfigureServices agregamos el servicio de DbContext

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<MiContexto>(options => options.UseSqlServer(| ));

    services.AddControllersWithViews();
}
```

Debemos definir cual es el servidor de sql (localDb), el tipo de autenticación y nombre para la base de datos.

```
tions.UseSqlServer(Configuration.GetConnectionString("EstacionamientoCS" ));
```

Posteriormente nos vamos a appsettings.json para definir el Connection String

Debemos incluir a qué servidor se va a conectar, localDB en este caso,

```
"ConnectionStrings": {  
  "EstacionamientoCS": "server=(localdb)\\MSSQLLocalDB",  
  "OtroCS": "Otra cosa"  
}
```

definir el nombre...

```
: "server=(localdb)\\MSSQLLocalDB;database=EstacionamientoDB-D;",  
[REDACTED]
```

por ultimo la conexión...

```
D;Trusted_Connection=true;";  
=true;"
```

Completo:

```
"ConnectionStrings": {  
  "EstacionamientoCS": "server=(localdb)\\MSSQLLocalDB;database=EstacionamientoDB-D;Trusted_Connection=true;";  
  "OtroCS": "server=serverprod1.midominio.local;database=RRHADB;Trusted_Connection=true;"  
}
```

Coincide el nombre de la clase Startup.cs con el nombre en appsettings.json