**Add EF Core NPGSQL (Hangisini sorarlarsa o eklenecek)**

.**NET CLI (View > Terminal > cd ./{ProjectName} yapılacak önce)** (Burada hangi projeye eklenecekse önce o seçilmeli)

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL **Program.cs DataLayer proj**

**PackageManager Console**

NuGet\Install-Package Npgsql.EntityFrameworkCore.PostgreSQL **Program.cs DataLayer proj**

**Add EF Core SQLServer (Hangisini sorarlarsa o eklenecek)**

dotnet add package Microsoft.EntityFrameworkCore.SQLServer **Program.cs projda**

NuGet\Install-Package Microsoft.EntityFrameworkCore.SQLServer **Program.cs projda**

**Add EF Core**

dotnet add package Microsoft.EntityFrameworkCore **Program.cs ve DataLayer projda**

dotnet add package Microsoft.EntityFrameworkCore.Design(Migration **DataLayer proj ve program.cs**)

dotnet add package Microsoft.EntityFrameworkCore.Relational

dotnet add package Microsoft.EntityFrameworkCore.Tools (migration update-database **DataLayer projda**)

dotnet add package Microsoft.EntityFrameworkCore.InMemory

dotnet tool install --global dotnet-ef

NuGet\Install-Package Microsoft.EntityFrameworkCore

NuGet\Install-Package Microsoft.EntityFrameworkCore.Design (Migration ve program.cs in oraya)

NuGet\Install-Package Microsoft.EntityFrameworkCore.Relational

NuGet\Install-Package Microsoft.EntityFrameworkCore.Tools

NuGet\Install-Package Microsoft.EntityFrameworkCore.InMemory

**Swagger**

dotnet add package Swashbuckle.AspNetCore **Program.cs in olduğu yere**

NuGet\Install-Package Swashbuckle.AspNetCore **Program.cs in olduğu yere**

**JWT Bearer**

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer **Program.cs in olduğu yere**

NuGet\Install-Package Microsoft.AspNetCore.Authentication.JwtBearer

**AutoMapper**

dotnet add package AutoMapper

NuGet\Install-Package AutoMapper -Version 14.0.0

**Microsoft.Identity**

dotnet add package Microsoft.AspNetCore.Identity **PassswordHash için**

dotnet add package Microsoft.AspNetCore.Identity .EntityFrameworkCore **(IdentityDbContext için)**

**Appsettings.json’a ConnectionString eklenecek**

"JwtSettings": {

"Issuer": "MyAppIssuer",

"Audience": "MyAppAudience",

"SecretKey": "my\_super\_secret\_key\_123!"

},

"ConnectionStrings": {

"PostgresConnection": "Host=localhost;Database=vaccinedb;Username=postgres;Password=1234qqq"

}

"ConnectionStrings": {

"SBDbContext": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=model;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False"

},

**lunchSettings**

Eğer projede otomatik olarak WebAPI’nin swagger’ı açmasını istiyorsan lunchSettings’de aşağıdaki ayarları eklemek gerekir.

Eğer IISExpress’de proje başlatılıyorsa,

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

},

"launchUrl": "swagger"

}

"launchBrowser": true, ve "launchUrl": "swagger" değerlerinin eklendiği kontrol edilmelidir.

**NpgSQL Code First**

**DBContext ve Entity’ler eklenecek. Daha sonra efcore için yukarıdaki seed işleminin çalışması için db migration eklenmesi gerek. PackageManager’da aşağıdaki komutlar sırasıyla yazılır.**

*Eğer CodeFirst appsettings.Development.json üzerinden yapılacaksa ve IDesignTimeDbContextFactory<AppDbContext> ile code first yapılacaksa önce ortam development ortamı olduğu bildirilmeli ve aşağıdaki kod kullanılmalı . Burada migration yapmak için şu paketlerin Data katmanında olması gerekir. Daha sonra migration ekleme komutaları çalışacak. Bu işlemler .sln root dizinden yapılmalı. Bu iş yapılırken* Microsoft.EntityFrameworkCore.Design *nuget paket de webapi.csproj da olmalı*

***dotnet add package Microsoft.Extensions.Configuration***

***dotnet add package Microsoft.Extensions.Configuration.Json***

***dotnet add package*** Microsoft.EntityFrameworkCore.Design

$env:ASPNETCORE\_ENVIRONMENT = "Development"

dotnet ef migrations add InitialCodeFirstMigration --context AppDbContext --project Vaccine.Data/VaccineApp.Data.csproj --startup-project WebAPI/WebAPI.csproj

dotnet ef database update --project Vaccine.Data/VaccineApp.Data.csproj --startup-project WebAPI/WebAPI.csproj

dotnet ef migrations remove --project Vaccine.Data/VaccineApp.Data.csproj --startup-project WebAPI/WebAPI.csproj

dotnet ef migrations add InitialCodeFirstMigration --context AppDbContext

dotnet ef database update --context AppDbContext

Enable-Migrations

add-migration EmptyMigration

update-database

**Daha sonra seed işlemi için uygulama ayağa kaldırılır.**

**NpgSQL DB First**

Eğer hazırda bir db varsa ve buradan entity modeller yaratılacaksa bu komut kullanılır**. Eğer public adında bir şema yok derse db export import yapmak lazım.**

dotnet tool install --global dotnet-ef

dotnet ef dbcontext scaffold "Host=localhost;Database=vaccinedb;Username=postgres;Password=1234qqq" Npgsql.EntityFrameworkCore.PostgreSQL --output-dir Context --context AppDbContext --no-onconfiguring --force –verbose --use-database-names

**RabbitMq Kurulum**

dotnet add package RabbitMQ.Client

docker run -d --hostname rabbitmq --name rabbitmq -e RABBITMQ\_DEFAULT\_USER=admin -e RABBITMQ\_DEFAULT\_PASS=123456 -p 5672:5672 -p 15672:15672 rabbitmq:3-management

**Redis Kurulumu**

~~dotnet add package StackExchange.Redis~~  **Program.cs in olduğu yere**

dotnet add package Microsoft.Extensions.Caching.StackExchangeRedis

docker run -d --name redis-stack -p 6379:6379 -p 8001:8001 redis/redis-stack:latest

**DependencyInjection**

dotnet add package Microsoft.Extensions.DependencyInjection.Abstractions **Program.cs**

builder.Services.AddScoped(typeof(IRepository<>), typeof(Repository<>));

builder.Services.AddScoped<IUnitOfWork, UnitOfWork>(); **Program.cs**

**UnitOfWorkTransactionFilter**

UnioOfWork transaction yönetimini sağlamak için servislerin başına bir filtre koyulur

dotnet add package Microsoft.AspNetCore.Mvc.Abstractions **Business katmanı**

builder.Services.AddScoped<UnitOfWorkTransactionFilter>(); **Program.cs**

builder.Services.AddControllers(options =>

{

options.Filters.Add<UnitOfWorkTransactionFilter>();

});

**ElasticSearch**

docker pull docker.elastic.co/elasticsearch/elasticsearch-wolfi:9.0.0

docker run -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:7.5.2

dotnet add package NEST

dotnet add package Elasticsearch.Net

**React**

* npx create-react-app <my-app-name> (Eğer çalışmazsa aşağıdakiler yapılacak)
  + npm cache clean –force
  + npm install -g npm@latest
  + npm install -g create-react-app
* npm install react-router-dom
* npm install react-bootstrap
* npm i @material-ui/core --legacy-peer-deps
* npm config set legacy-peer-deps true (eğer dependency hatası verirse)

Chrome’da kodun görünmesi için package-json’a eklenir.

* "build": "GENERATE\_SOURCEMAP=false react-scripts build",
* "winBuild": "set \"GENERATE\_SOURCEMAP=false\" && react-scripts build",

**Program.cs**