# **مرحله اول: ساختار بندی پوشه ها در سطح اول پروژه قبل از لایه بندی**

* Doc
* Src
* Test

# **مرحله ؟ : نصب پروژه از نوع ASP.NET Core Web Api 5.0**

# مرحله ؟ : ساخت مدل ها یا همان **Entities**

Account // حساب

Transaction // تراکنش

Response // پاسخ

# مرحله ؟ : نصب پکیج های لازم شامل

Microsoft.EntityFrameworkCore 5.0

Microsoft.EntityFrameworkCore.SqlServer 5.0

Microsoft.EntityFrameworkCore.Tools 5.0

AutoMapper.Extensions.Microsoft.DependencyInjection 11

# مرحله ؟ : ساخت **Context** و اقدامات مربوطه شامل:

* ساخت فایل کانتکست و ارث بری از کلاس DbContext و تعیین مسیر در سازنده بصورت
* public CoreBankingDbContext(DbContextOptions<CoreBankingDbContext> options) : base(options)
* ایجاد DbSet ها
* ایجاد کانکشن استرینگ درون فایل appsettings.json جهت داینامیک بودن آن و توانایی تغییر راحت

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=CoreBankingDB;Trusted\_Connection=true;"

},

* اینجکت کانکشن استرینگ ایجاد شده به درون سرویس ها بصورت

services.AddDbContext<CoreBankingDbContext>(q => q.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

* ساخت Migration با دستور Add-Migration init
* ایجاد دیتابیس با دستور Update-Databas

تا این مرحله، ساختار ما ایجاد شده و درون دیتابیس قرار می گیرد.

# مرحله ؟ : ساخت اینترفیس ها و سرویس ها درون پوشه **Services**

* Services
  + Interfaces
  + Implementations

حالا درون تابع سازنده سرویس ها، می توانیم کانتکست را دریافت کنیم و آنرا استفاده کنیم:

private readonly CoreBankingDbContext \_dbContext;

public AccountService(CoreBankingDbContext dbContext)

{

\_dbContext = dbContext;

}

# مرحله ؟ : ساخت پوشه **Profiles** و فایل **AutoMapperProfiles**

در این مرحله برای مدل ها چندین DTO می سازیم و از انها استفاده میکنیم تا برای هر نوع عملیات بخش های مختلفی از مدل اصلی را به کاربر نمایش دهیم و قسمت های حساس و محرمانه را مخفی کنیم

# مرحله ؟ : ساخت **Controllers**