ÖDEVİMİ KİME YOLLAYACAĞIM?

Ödevlerinizi aşağıdaki listede belirtilmiş olan sınıf arkadaşınızla iletişime geçip, birlikte belirleyeceğiniz bir yöntemle ona göndermeniz gerekmektedir. Ödevi gönderdiğiniz arkadaşınız, ödevinizi değerlendirecek ve ödevinizle ilgili değerlendirmelerini sizinle paylaşacaktır. Başarılar :)

Öğrenci	Ödevini Göndereceği Öğrenci
Ali Görkem GELMEZ	Uras YILMAZ
Asikar Furkan ISLAM	Nisanur KOÇ
Aylin YÜKSEL	Samet ECE
Batuhan CELENK	Enes SAÇI
Büsra SÜMBÜL	Yagmur Sultan YAZICIOGLU
Cemre Su SÖYLER	Sedranur GIRIS
Ceyda Nur FILIZ	Talha Emirhan KAYGISUZ
Emir Can UYSAL	Kübra GÖKÇEKE
Emirhan ÇEBIS	Mehmet Harun KAYA
Enes SAÇI	Cemre Su SÖYLER
Halil ibrahim DEMIR	Minel YILMAZ
Hilal Nur ARSLAN	Ömer Tarik AYDOGAN
İkra Selin ARSLAN	Ceyda Nur FILIZ
Kübra GÖKÇEKE	Batuhan CELENK
Mehmet Harun KAYA	Emir Can UYSAL
Mehmet SÖNMEZ	Büsra SÜMBÜL
Mert OKYAY	Mehmet SÖNMEZ
Merve KABAKCI	Mürüvet YOLUK
Mesut AYHAN	Neval SENÖZEN
Minel YILMAZ	Serra TOZLU
Mürüvet YOLUK	Asikar Furkan ISLAM
Neval SENÖZEN	Aylin YÜKSEL
Nisanur KOÇ	Ugur YENIAY
Ömer Tarik AYDOGAN	Hilal Nur ARSLAN
Samet ECE	Sıla KAHYA
Sedranur GIRIS	Sıla ÖNER
Serra TOZLU	Mert OKYAY
Sıla KAHYA	Mesut AYHAN
Sıla ÖNER	Merve KABAKCI
Talha Emirhan KAYGISUZ	Umut Furkan KESKIN
Ugur YENIAY	Halil ibrahim DEMIR
Umut Furkan KESKIN	İkra Selin ARSLAN
Uras YILMAZ	Emirhan ÇEBIS
Yagmur Sultan YAZICIOGLU	Ali Görkem GELMEZ

Ödev 1: Proje İskeleti Kurulumu - "Movie Archive API"

Hedef: `.NET CLI` komutlarını kullanarak sıfırdan, hatasız ve hızlı bir şekilde 3 katmanlı proje mimarisi iskeleti kurma becerisini pekiştirmek.

Senaryo: Kişisel bir film arşivini yönetecek bir API'nin temelini atacaksınız.

Görevler:

1. Bilgisayarınızda `MovieArchiveAPI` adında bir ana klasör oluşturun.
2. Bu klasörün içinde terminali açın.
3. `dotnet new sln -n MovieArchive` komutu ile solution dosyasını oluşturun.
4. Aşağıdaki üç projeyi **CLI komutları ile** oluşturun:
    - `MovieArchive.API` (ASP.NET Core Web API projesi)
    - `MovieArchive.Business` (Class Library projesi)
    - `MovieArchive.Data` (Class Library projesi)
5. Oluşturduğunuz bu üç projeyi `dotnet sln add` komutu ile solution'a ekleyin.
6. Mimarinin en önemli kuralı olan tek yönlü bağımlılığı (`API -> Business -> Data`) **CLI komutları ile** kurun:
    - `dotnet add MovieArchive.API reference MovieArchive.Business`
    - `dotnet add MovieArchive.Business reference MovieArchive.Data`

Ödev 2: Proje İskeleti Kurulumu - "Email Service API"

Hedef: CLI komutlarını tekrar ederek "kas hafızası" oluşturmak. Farklı isimlendirmelerle aynı yapıyı kurabilmek.

Senaryo: E-posta gönderimi yapacak bir mikroservisin altyapısını kuruyorsunuz.

Görevler:

1. `EmailServiceAPI` adında bir ana klasör oluşturun.
2. İçinde `EmailService.sln` adında bir solution oluşturun.
3. Aşağıdaki üç projeyi **CLI komutları ile** oluşturun:
    - `EmailService.API`
    - `EmailService.Business`
    - `EmailService.Data`
4. Tüm projeleri solution'a ekleyin.
5. Gerekli proje referanslarını (`API -> Business -> Data`) **CLI komutları ile** ayarlayın.

Ödev 3: Proje İskeleti Kurulumu - "Survey App API"

Hedef: Proje kurulum adımlarını ezberden ve hatasız yapabilme seviyesine gelmek.

Senaryo: Kullanıcıların anket oluşturup cevaplayabileceği bir uygulamanın backend'ini hazırlıyorsunuz.

Görevler:

1. `SurveyAppAPI` adında bir ana klasör oluşturun.
2. İçinde `SurveyApp.sln` adında bir solution oluşturun.
3. Aşağıdaki üç projeyi **CLI komutları ile** oluşturun:
    - `SurveyApp.API`
    - `SurveyApp.Business`
    - `SurveyApp.Data`
4. Tüm projeleri solution'a ekleyin.
5. Gerekli proje referanslarını (`API -> Business -> Data`) **CLI komutları ile** ayarlayın.

Ödev 4: Proje İskeleti Kurulumu - "Forum Platform API"

Hedef: `.NET CLI` ile proje iskeleti oluşturma işlemini bir standart haline getirmek ve bu konuda tam yetkinlik kazanmak.

Senaryo: Kullanıcıların başlıklar açıp mesajlar yazabileceği bir forum platformunun API altyapısını kuruyorsunuz.

Görevler:

1. `ForumPlatformAPI` adında bir ana klasör oluşturun.
2. İçinde `ForumPlatform.sln` adında bir solution oluşturun.
3. Aşağıdaki üç projeyi **CLI komutları ile** oluşturun:
    - `ForumPlatform.API`
    - `ForumPlatform.Business`
    - `ForumPlatform.Data`
4. Tüm projeleri solution'a ekleyin.
5. Gerekli proje referanslarını (`API -> Business -> Data`) **CLI komutları ile** ayarlayın.

Ödev 5: Pratik Uygulama - Özel Doğrulama Kuralları ("Event Manager API")

Hedef: Basit bir Web API projesi içinde, farklı senaryolara yönelik özel doğrulama `Attribute`'leri yazmak, bunları modele uygulamak ve Swagger üzerinden test ederek çalıştığını görmek.

Senaryo: Yeni etkinlikler oluşturan basit bir API yazacaksınız.

Görevler:

1. **Proje Kurulumu:**
   - `EventManagerAPI` adında **tek bir** ASP.NET Core Web API projesi oluşturun. (Bu ödevde katmanlı mimari gerekmiyor).
2. **Model Oluşturma:**
   - Proje içinde `Event.cs` adında bir model sınıfı oluşturun.
   - İçinde şu property'ler olsun: `Id` (int), `Name` (string), `Description` (string), `EventDate` (DateTime).
3. **Controller Oluşturma:**
   - `EventsController.cs` adında bir API controller oluşturun.
   - İçerisinde sadece yeni bir etkinlik eklemeyi sağlayan bir `POST` metodu olsun. (Veritabanı bağlantısı gerekmez, metot boş kalabilir veya sadece `Ok()` dönebilir).
4. **Özel Doğrulama Kuralları Yazma:** Proje içinde `ValidationAttributes` adında bir klasör oluşturun ve içine şu 3 `Attribute` sınıfını yazın:
   - **`[NoPastDate]`**: `EventDate` property'sine uygulanacak. Bu kural, girilen tarihin `DateTime.Now`'dan daha eski bir tarih olmasını engellemeli.
   - **`[MustStartWithUppercase]`**: `Name` property'sine uygulanacak. Bu kural, etkinlik adının ilk harfinin büyük harf olup olmadığını kontrol etmeli. (İpucu: `value as string` ile değeri alıp `char.IsUpper(str[0])` kullanabilirsiniz).
   - **`[DescriptionMustNotContainName]`**: `Description` property'sine uygulanacak. Bu kural, `validationContext.ObjectInstance` kullanarak `Event` nesnesinin `Name` property'sindeki değeri almalı ve bu değerin `Description` içinde geçip geçmediğini kontrol etmeli. Bu, "cross-property validation" alıştırmasıdır.
5. **Uygulama ve Test:**
   - Oluşturduğunuz bu 3 `Attribute`'ü `Event.cs` modelindeki ilgili property'lerin üzerine ekleyin.
   - Projeyi çalıştırın, Swagger'ı açın ve POST endpoint'ini kullanarak her bir kuralı ihlal eden 3 farklı istek göndererek kurallarınızın doğru çalışıp çalışmadığını test edin.