TRİTTER (Twitter Clone)

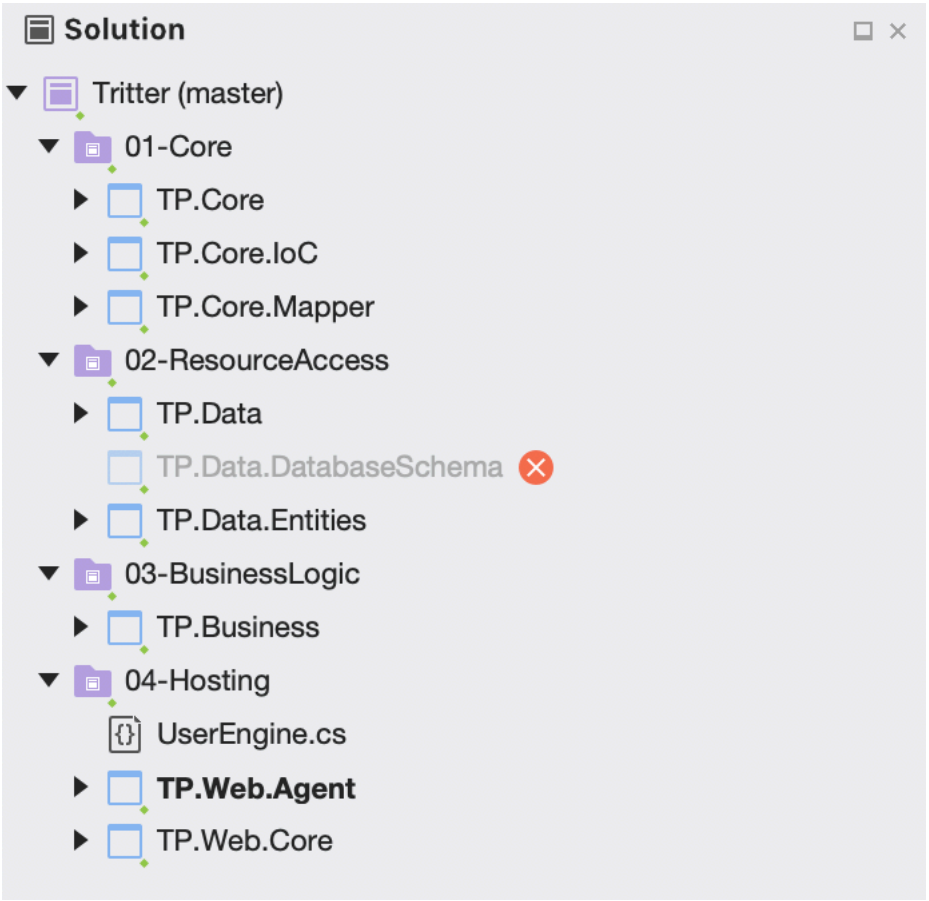
Arda ÖZDEMİR
Kerem DÜNDAR
Mustafa DAĞDELEN

INF353 Web Programlamaya Giriş Proje Raporu

Projeye başlarken ilk olarak, proje grubu içerisinde görev dağılımını gerçekleştirdik. Daha sonra projenin geliştirilmesi sırasında ekip içerisinde geliştirmeyi kolaylaştıracak ortamların kurulumunu yaptık. Bu kurulumlar AWS içinde bir Windows Server Instance kurulumunu ve bu instance içerisinde bir MS SQL Server 2017 kurulumunu gerçekleştirdik. Proje ekibi içerisinde görevlerin takibi için Trello uygulamasını kullandık. Uygulamanın versiyon kontrolü için ise GitHub üstünden bir repository oluşturduk. Proje geliştirme ortamının kurulmasından sonra geliştirme sürecine başladık. Projeyi .Net Core platformu, MVC design pattern'ını baz alarak geliştirdik.

PROJE DOSYA YAPISI

Proje dosya yapısı Şekil - 1'de görüldüğü şekildedir. Şekilde görüldüğü üzere projemiz 4 katmandan oluşmaktadır.



Şekil - 1

KATMANLAR

01-Core = Projenin her bölgesinde kullanılan session ve keywordlerin oluşturulduğu katmandır.

02-ResourceAccess = Entityler ve Modellerin oluşturulduğu, aynı zamanda database bağlantılarının sağlandığı katmandır.

03-BussinessLogic = Oluşturulan Modeller ve Entitylerin kullanıldığı, Post ve Get işlemlerinin yapıldığı ara katmandır.

04-Hosting = Uygulamanın Controller ve View yapılarının bulunduğu katmandır.

CONTROLLER

Controller, ara katman olan Business Logic ile View'ler arasında bağlantıyı kuran bu bağlantının sonuçlarını tekrar View'lere gönderilmesini sağlayan yapıdır.

```
public IActionResult Profile(string userId)
{
    var userResult = _userEngine.GetByUserId(userId);
    var tritResult = _tritEngine.GetTritsByUserId(userId);
    ProfilePageUserModel profilePageUserModel = new ProfilePageUserModel();
    ViewBag.UserId = userId;

    if (userResult.IsSuccess)
    {
        profilePageUserModel.UserCreateModel = userResult.Data;
    }

    if (userResult.IsSuccess)
    {
        profilePageUserModel.TritListModel = tritResult.Data;
    }
    else
    {
        ViewBag.ListIsExist = 0;
    }

    ViewBag.FullName = profilePageUserModel.UserCreateModel.user_name + " " + profilePageUserModel.UserCreateModel.user_surname;
    ViewBag.Email = profilePageUserModel.UserCreateModel.user_email;
    return View(profilePageUserModel);
}
```

Şekil - 2

Şekil - 2'de proje içerisinden örnek bir Controller nesnesi gösterilmiştir. Bu nesne ile proje içerisinde kullanıcının kendi profilene yada her hangi bir profile gitmek istemesi aksiyonunun yönetilmesi yapılmaktadır.

VİEWS

View'lerin kullanıcıya sunulması kısmında ASP.Net platformunda master page yapısına denk gelen Layout(Shared) kavramını kullandık. Bu sayede her sayfada tekrar eden ögelerin arayüz tarafında tekrar kodlanmasının önüne geçmiş olduk.

ENTİTİES

Database erişimini sağladığımız yapılardır. Örnek bir Entity yapısı Şekil - 3'de gösterilmiştir.

```
namespace TP.Data.Entities
{
    [Table(name: "TB_Trit")]
    public class Trit : BaseEntity
    {
        [Key]
        [Required]
        [StringLength(255)]
        [Column(Order = 1)]
        public int trit_id { get; set; }

        [Required]
        [StringLength(255)]
        [Column(Order = 1)]
        public string trit_user_id { get; set; }

        [Required]
        [StringLength(100)]
        [Column(Order = 2)]
        public DateTime trit_time { get; set; }

        [Required]
        [StringLength(255)]
        [Column(Order = 3)]
        public string trit_text { get; set; }

    }
}
```

Şekil - 3

MODELS

Modeller Database'den aldığımız datanın nesne yapısında tutulmasını sağlayan classlarımızdır. Örneğin Şekil - 3'de belirtilen entity ile erişilen database tablosundan gelen bir data Şekil - 4'de gösterilen şekilde class yapısında tutulur.

```
namespace TP.Data.Entities.PageModels.TritModel
{
    public class TritListModel
    {
        public int trit_id { get; set; }

        public string trit_user_id { get; set; }

        public DateTime trit_time { get; set; }

        public string trit_text { get; set; }
    }
}
```

Şekil - 4