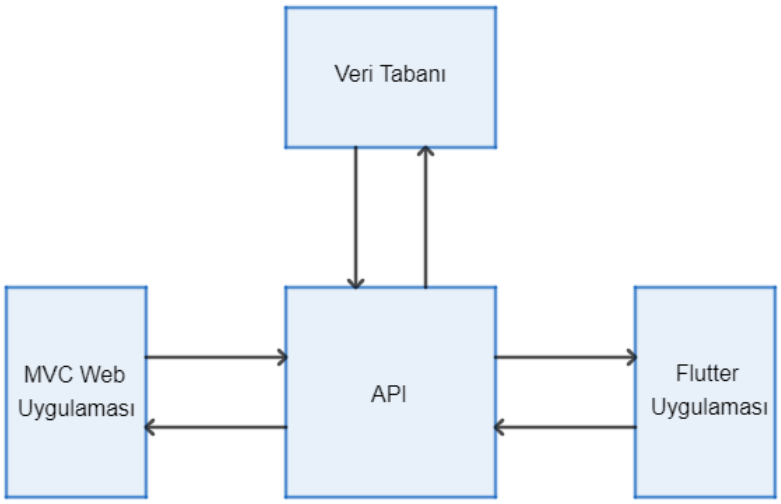
KÜTÜPHANE WEB API

KutuphaneWebAPI, DataLibrary ve KutuphaneDB klasörleri kütüphane projesinin API ve Veri Tabanı kısımlarını oluşturuyor.



Şekil 1.1. Projenin Genel Yapısı

API ile gelen istekler, Web API'nin controllerları tarafından karşılanıp DataLibrary'deki processorlar ile veri tabanı ile aralarındaki bağlantı kurulduktan sonra işlemler yapılmaktadır.

```
public class MembersPuanController : ApiController
{
    // GET: api/MembersPuan
    0 references
    public List<PopularBookModel> Get()
    {
        return GetPopularBooks();
    }

    // GET: api/MembersPuan/5
    0 references
    public List<int> Get(int id)
    {
        return GetMembersPuan(id);
    }

    // POST: api/MembersPuan
    0 references
    public int Post([FromBody] Models.BookMemberModel value)
    {
        return SetMembersPuan(value.MemberId, value.BookId, value.Puan);
    }
```

Şekil 1.2. Controller'daki GET ve POST fonksiyonlarına örnek

Şekil 1.2'de KutuphaneWebAPI'deki membersPuanController gösterilmiştir. Buradaki Post fonksiyonunu ele alacak olursak, fonksiyon, BookMemberModel tipinde (şekil 1.3) bir gövdeye sahip API'yi input olarak almaktadır ve veri tabanına yazmak üzere Şekil 1.4'teki SetMembersPuan fonksiyonunu çağırmaktadır. SetmembersPuan fonksiyonu, Datalibrary kütüphanesinin BusinnessLogic namespace'inin altında BookMemberProcessor.cs içinde barınmaktadır. Fonksiyon,

veri tabanını güncellemek için gerekli olan sorguyu, modelle birlikte SaveData(Şekil 1.5) fonksiyonuna göndermektedir. SaveData fonksiyonu DataLibrary/DatatAccess/SqlDataAccess.cs içinde barınmaktadır. Bu fonksiyon Dapper ile veri tabanına bağlanarak sorguyu gerçekleştirmektedir.

```
public class BookMemberModel
{
    2 references
    public int BookId { get; set; }

    2 references
    public int MemberId { get; set; }

    1 reference
    public int TeslimDurumu { get; set; } //0 => teslim edilmedi, 1 => teslim edildi

    0 references
    public string TeslimTarihi { get; set; } // YYYY-MM-DD

    1 reference
    public int Puan { get; set; }
}
```

Şekil 1.3. BookMemberModel Modeli

```
1 reference
public static int SetMembersPuan(int memberid, int bookid, int puan)
{
    BookMemberModel data = new BookMemberModel
    {
        MemberId = memberid,
        BookId = bookid,
        Puan = puan
    };

    string sql = $@"UPDATE dbo.BookMember
                    SET puan = @puan
                    WHERE MemberId=@MemberId and BookId=@BookId;";

    return SqlDataAccess.SaveData(sql, data);
}
```

Şekil 1.4. SetMembersPuan fonksiyonu

```
17 references
public static int SaveData<T>(string sql, T data)
{
    using (IDbConnection cnn = new SqlConnection(GetConnectionString()))
    {
        try
        {
            return cnn.Execute(sql, data);
        }
        catch
        {
            return 0;
        }
    }
}
```

Şekil 1.5. SaveData fonksiyonu

Kurulum

Kurulum için Github'daki dosyalar indirildikten sonra (KutuphaneWebAPI, DataLibrary ve KutuphaneDB aynı klasörde bulunmalıdır), KutuphaneWebAPI klasörünün içindeki KutuphaneWebAPI.sln açılır. Açıldıktan sonra ilk kurulumun tamamlanmasıyla beraber KutuphaneWebAPI klasörünün içinde .vs klasörü gizli klasör olarak oluşacaktır. Flutter uygulamasının API'ye bağlanması için …\KutuphaneWebAPI\.vs\KutuphaneWebAPI\config yolundaki applicationhost.config dosyası içindeki <binding protocol="http" bindingInformation="*:44383:localhost "/> olan kısım, <binding protocol="http" bindingInformation="*:44383:127.0.0.1"/> olarak güncellenmelidir.

Uygulama çalıştırıldığında …\bin\roslyn\csc.exe yolunun bir parçası bulunamadı hatası alınırsa visual studiodaki package manager consol'a **Update-Package Microsoft.CodeDom.Providers.DotNetCompilerPlatform -r** kodu yazılmalıdır.

Bu adımların ardından API kurulumu tamamlanmış olacaktır.

NOT: DataLibrary.DataAcces.sqlDataAccess.cs dosyasının 90. satırında, parolayı şifrelemek için private-keyin konumuna referans vardır. Bu referansın, yüklenecek makinedeki konumuna göre güncellenmesi gerekmektedir.