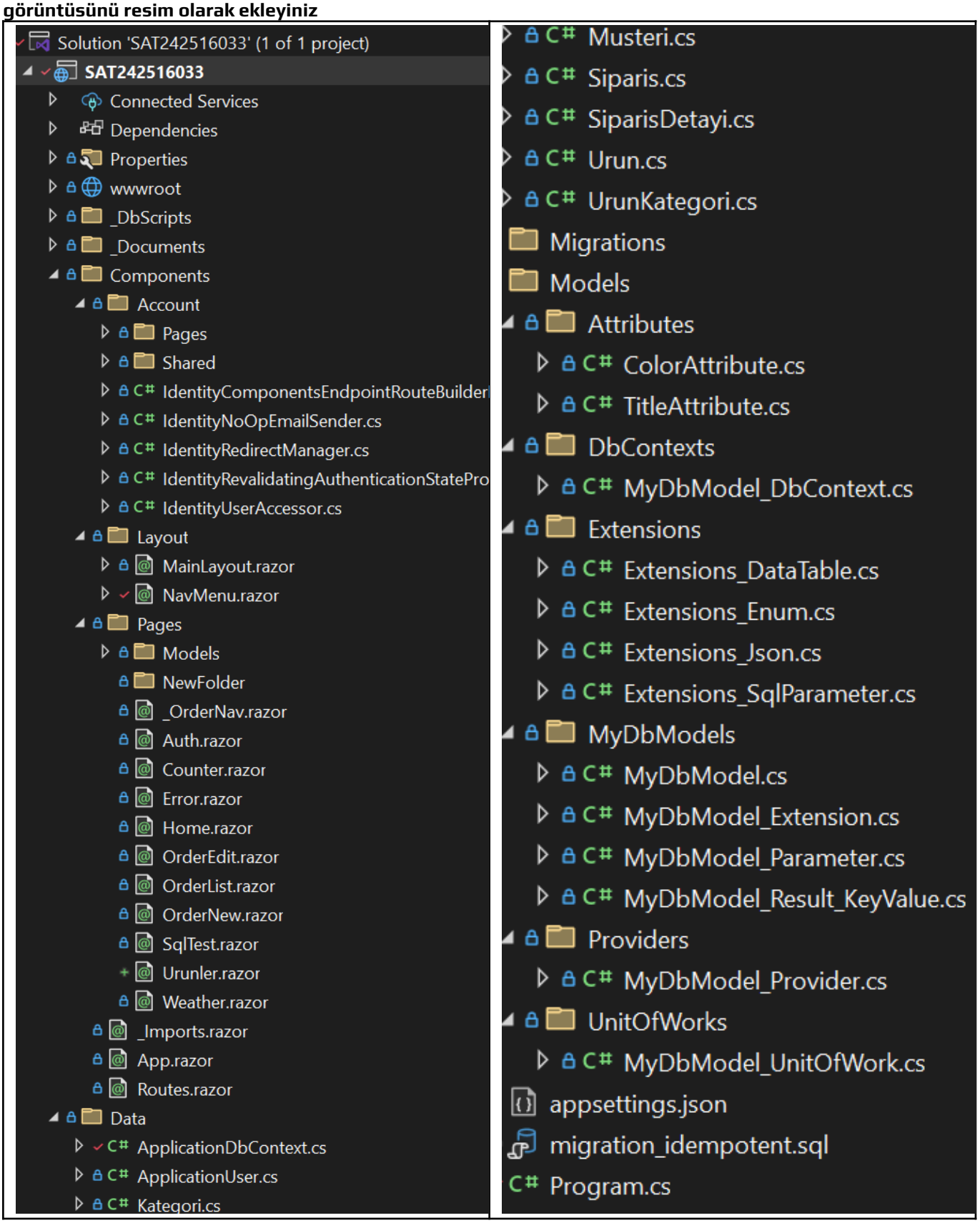
Proje Adı	Ürün Sipariş ve Takip Sistemi
Proje Konusu	Müşteri, ürün, kategori, sipariş ve sipariş detaylarının yönetildiği; stok takibi ve sipariş süreçlerinin yönetildiği Blazor Server tabanlı sistem.
Proje Çözüm Adı	SAT242516033

Proje 'Çözüm Gezgini (Solution Explorer)' penceresinin (alt klasörler ve dosyalar dahil olacak şekilde) ekran görüntüsünü resim olarak ekleyiniz



Projede oluşturulan arabirimlerin (interface) C# kodlarını yazınız.

```
public interface IMyDbModel
{
    string Message { get; set; }
    IMyDbModel_Parameter Parameters { get; set; }
    IDictionary<object, object> OrderByItems { get; set; }
}
```

```
public interface IMyDbModel<T> : IMyDbModel where T : class, new()
{
    IEnumerable<T> Items { get; set; }
}
```

```
public interface IMyDbModel_Parameter
{
    string OrderBy { get; set; }
    int PageNumber { get; set; }
    int PageSize { get; set; }
    int TotalPageCount { get; }
    int TotalRecordCount { get; set; }
    IDictionary<string, object> Params { get; set; }
    IDictionary<string, string> Where { get; set; }
}
```

```
public interface IMyDbModel_Result_KeyValue<TKey, TValue>
{
 TKey Key { get; set; }
TValue Value { get; set; }
}
```

```
public interface IMyDbModel_Provider
{
    ValueTask<IMyDbModel<TResult>> Execute<TResult>(IMyDbModel<TResult>
myResultModel,
        string spName = "",
        bool isPagination = true) where TResult : class, new();

    ValueTask<IMyDbModel<TResult>> Execute<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();

    ValueTask<IEnumerable<TResult>> GetItems<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();

    ValueTask<IEnumerable<TResult>> SetItems<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();
}
```

```
public interface IMyDbModel_UnitOfWork
{
    Task Execute<T>(IMyDbModel<T> myDbModel, string spName = "", bool isPagination =
true)
        where T : class, new();
}
```

Projede oluşturulan sınıfların (class) C# kodlarını yazınız.

Uyarı : Sınıf içerisinde metotların sadece isim ve parametreleri yazılacak.
Örnek : public async Task MethodAsync(parameters…) { return null; }

```
public MyDbModel() { }
public MyDbModel(int pageNumber, int pageSize, string orderBy) { }
```

```
public static MyDbModel_Parameter Create(int pageNumber, int pageSize, string
orderBy) { }
private MyDbModel_Parameter(int pageNumber, int pageSize, string orderBy) { }
```

```
public class MyDbModel_Result_KeyValue<TKey, TValue> { }
```

```
public static IDictionary<object, object> GetOrderByItems<E>(this MyDbModel<E>
myDbModel) where E : class, new() { }
```

```
public MyDbModel_Provider(IMyDbModel_UnitOfWork myDbModel_UnitOfWork) { }
public async ValueTask<IMyDbModel<TResult>> Execute<TResult>(IMyDbModel<TResult>
myResultModel, string spName = "", bool isPagination = true) { }
public async ValueTask<IMyDbModel<TResult>> Execute<TResult>(string spName = "",
params (string Key, object Value)[] parameters) { }
public async ValueTask<IEnumerable<TResult>> GetItems<TResult>(string spName = "",
params (string Key, object Value)[] parameters) { }
public async ValueTask<IEnumerable<TResult>> SetItems<TResult>(string spName = "",
params (string Key, object Value)[] parameters) { }
```

```
public MyDbModel_UnitOfWork(TDbContext context) { }
public async Task Execute<T>(IMyDbModel<T> myDbModel, string spName = "", bool
isPagination = true) where T : class, new() { }
```

```
public MyDbModel_DbContext(DbContextOptions<MyDbModel_DbContext> options) :
base(options) { }
protected override void OnModelCreating(ModelBuilder modelBuilder) { }
```

```
public static IEnumerable<T> DataTableToList<T>(this DataTable table) where T : class
{ }
public static T GetObject<T>(this DataRow row, List<string> columnsName) where T :
class { }
```

```
public static string Color<T>(this T value) { }
public static string Title<T>(this T value) { }
```

```
public static T JsonToItem<T>(this string jsonItem) { }
public static string ItemToJson<T>(this T item) { }
public static List<T> JsonToList<T>(this string json) { }
public static string ListToJson<T>(this List<T> list) { }
```

```
public static SqlParameter ToSqlParameter_Table_Type_Dictionary<TKey, TValue>(
    this IDictionary<TKey, TValue> dictionary,
    string parameterName,
    string parameterTypeName = "",
    int length = 0,
    SqlDbType sqlDbType = SqlDbType.Structured,
    ParameterDirection direction = ParameterDirection.Input) { }

public static SqlParameter ToSqlParameter_Data_Type<T>(
    this T value,
    string parameterName,
    ParameterDirection direction = ParameterDirection.Input,
    SqlDbType sqlDbType = SqlDbType.NVarChar) { }
```

```
public ColorAttribute(string color) { }
public string Color { get; set; }
```

```
public TitleAttribute(string title) { }
public string Title { get; set; }
```

appsettings.json dosyasında "DefaultConnection" ifadesini projenize göre yazınız

```
"DefaultConnection": "Server=localhost,1445; Database=SAT242516033; User Id=sa;
Password=Oo_454545; TrustServerCertificate=True;"
```

Program.cs dosyasında, gerekli servis kayıtlarını yapan C# kodlarını yazınız.

```
//// DBCONTEXTS
builder.Services.AddDbContext<MyDbModel_DbContext>(options =>
options.UseSqlServer(connectionString));

//// UNITOFWORKS
builder.Services.AddScoped<IMyDbModel_UnitOfWork,
MyDbModel_UnitOfWork<MyDbModel_DbContext>>();

//// MODELS
builder.Services.AddScoped(typeof(IMyDbModel<>), typeof(MyDbModel<>));

//// PROVIDERS
builder.Services.AddScoped<IMyDbModel_Provider, MyDbModel_Provider>();
```

App.razor bileşeninde, gerekli C# kodlarını yazınız

```
<Routes @rendermode="@RenderModeForPage" />
@code {

        [CascadingParameter] private HttpContext HttpContext { get; set; } =
default!;

        private IComponentRenderMode? RenderModeForPage =>
                        HttpContext.Request.Path
                                        .StartsWithSegments("/Account")
                                        ? null
                                        : InteractiveServer;

}
```

Projenizde oluşturmuş olduğunuz bileşenlerin tasarım ve C# kodlarını yazınız (Blazor Component)

NOT: daha fazla bileşen için, aşağıdaki tabloyu çoğaltınız.

Bileşen Adı	Urunler

Tasarım Kodları

```
@page "/urunler"
@using SAT242516033.Data
@using UrunSiparisTakip.Models
@using Microsoft.EntityFrameworkCore
@using Microsoft.AspNetCore.Components.Forms
@inject ApplicationDbContext Db

<h2 class="mt-3">Ürünler</h2>
<h5 class="text-primary">Liste</h5>

@if (list is null)
{
    <p>Veriler yükleniyor...</p>
}
else
{
    <div class="table-responsive">
        <table class="table table-striped table-sm align-middle">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Ad</th>
                    <th>SKU</th>
                    <th>Fiyat</th>
                    <th>Stok</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var u in list)
                {
                    <tr>
                        <td>@u.UrunId</td>
                        <td>@u.UrunAdi</td>
                        <td>@u.SKU</td>
                        <td>@u.BirimFiyat</td>
                        <td>@u.StokAdet</td>
                    </tr>
                }
            </tbody>
        </table>
    </div>

    <div class="row g-3 mt-3">
        <!-- Ürün Ekle -->
        <div class="col-md-4">
            <div class="card border-success">
                <div class="card-header fw-bold">Ürün Ekle</div>
                <div class="card-body">
                    <EditForm Model="@newUrun"
OnValidSubmit="@AddAsync">
                        <DataAnnotationsValidator />
                        <div class="mb-2">
                            <label class="form-label">Ürün Adı</label>
                            <InputText class="form-control
form-control-sm" @bind-Value="newUrun.UrunAdi" />
                        </div>
                        <div class="mb-2">
                            <label class="form-label">SKU</label>
                            <InputText class="form-control
form-control-sm" @bind-Value="newUrun.SKU" />
                        </div>
                        <div class="mb-2">
                            <label class="form-label">Fiyat</label>
```

```
                                <InputNumber class="form-control
form-control-sm" TValue="decimal" @bind-Value="newUrun.BirimFiyat" />
                            </div>
                            <div class="mb-3">
                                <label class="form-label">Stok</label>
                                <InputNumber class="form-control
form-control-sm" TValue="int" @bind-Value="newUrun.StokAdet" />
                            </div>
                            <div class="d-flex gap-2">
                                <button type="reset" class="btn
btn-outline-secondary btn-sm">Temizle</button>
                                <button class="btn btn-success btn-sm"
type="submit">Ekle</button>
                            </div>
                        </EditForm>
                    </div>
                </div>
            </div>

            <!-- Ürün Güncelle -->
            <div class="col-md-4">
                <div class="card border-warning">
                    <div class="card-header fw-bold">Ürün Güncelle</div>
                    <div class="card-body">
                        <EditForm Model="@updateUrun"
OnValidSubmit="@UpdateAsync">
                            <DataAnnotationsValidator />
                            <div class="mb-2">
                                <label class="form-label">Id</label>
                                <InputNumber class="form-control
form-control-sm" TValue="int" @bind-Value="updateUrun.UrunId" />
                            </div>
                            <div class="mb-2">
                                <label class="form-label">Ürün Adı</label>
                                <InputText class="form-control
form-control-sm" @bind-Value="updateUrun.UrunAdi" />
                            </div>
                            <div class="mb-2">
                                <label class="form-label">SKU</label>
                                <InputText class="form-control
form-control-sm" @bind-Value="updateUrun.SKU" />
                            </div>
                            <div class="mb-2">
                                <label class="form-label">Fiyat</label>
                                <InputNumber class="form-control
form-control-sm" TValue="decimal" @bind-Value="updateUrun.BirimFiyat"
/>
                            </div>
                            <div class="mb-3">
                                <label class="form-label">Stok</label>
                                <InputNumber class="form-control
form-control-sm" TValue="int" @bind-Value="updateUrun.StokAdet" />
                            </div>
                            <div class="d-flex gap-2">
                                <button type="reset" class="btn
btn-outline-secondary btn-sm">Temizle</button>
                                <button class="btn btn-warning btn-sm"
type="submit">Güncelle</button>
                            </div>
                        </EditForm>
                    </div>
                </div>
            </div>
```

```
        <!-- Ürün Sil -->
        <div class="col-md-4">
            <div class="card border-danger">
                <div class="card-header fw-bold">Ürün Sil</div>
                <div class="card-body">
                    <EditForm Model="@removeModel"
OnValidSubmit="@RemoveAsync">
                        <div class="mb-3">
                            <label class="form-label">Id</label>
                            <InputNumber class="form-control
form-control-sm" TValue="int?" @bind-Value="removeModel.Id" />
                        </div>
                        <div class="d-flex gap-2">
                            <button type="reset" class="btn
btn-outline-secondary btn-sm">Temizle</button>
                            <button class="btn btn-danger btn-sm"
type="submit">Sil</button>
                        </div>
                    </EditForm>
                </div>
            </div>
        </div>
    </div>
}

@code {
    List<Urun>? list;

    Urun newUrun = new();
    Urun updateUrun = new();

    class RemoveDto { public int? Id { get; set; } }
    RemoveDto removeModel = new();

    protected override async Task OnInitializedAsync()
    {
        list = await Db.Urunler.OrderBy(x => x.UrunId).ToListAsync();
    }

    async Task RefreshAsync()
    {
        list = await Db.Urunler.OrderBy(x => x.UrunId).ToListAsync();
        StateHasChanged();
    }

    async Task AddAsync()
    {
        Db.Urunler.Add(newUrun);
        await Db.SaveChangesAsync();
        newUrun = new();
        await RefreshAsync();
    }

    async Task UpdateAsync()
    {
        var ent = await Db.Urunler.FindAsync(updateUrun.UrunId);
        if (ent is not null)
        {
            ent.UrunAdi = updateUrun.UrunAdi;
            ent.SKU = updateUrun.SKU;
            ent.BirimFiyat = updateUrun.BirimFiyat;
            ent.StokAdet = updateUrun.StokAdet;
            await Db.SaveChangesAsync();
            updateUrun = new();
```

```
                await RefreshAsync();
            }
        }

        async Task RemoveAsync()
        {
            if (removeModel.Id is null) return;
            var ent = await Db.Urunler.FindAsync(removeModel.Id.Value);
            if (ent is not null)
            {
                Db.Urunler.Remove(ent);
                await Db.SaveChangesAsync();
                removeModel = new();
                await RefreshAsync();
            }
        }
}
```

Ekran Görüntüleri

UYARI: Resim boyutlarının çok büyük olmamasına dikkat ediniz