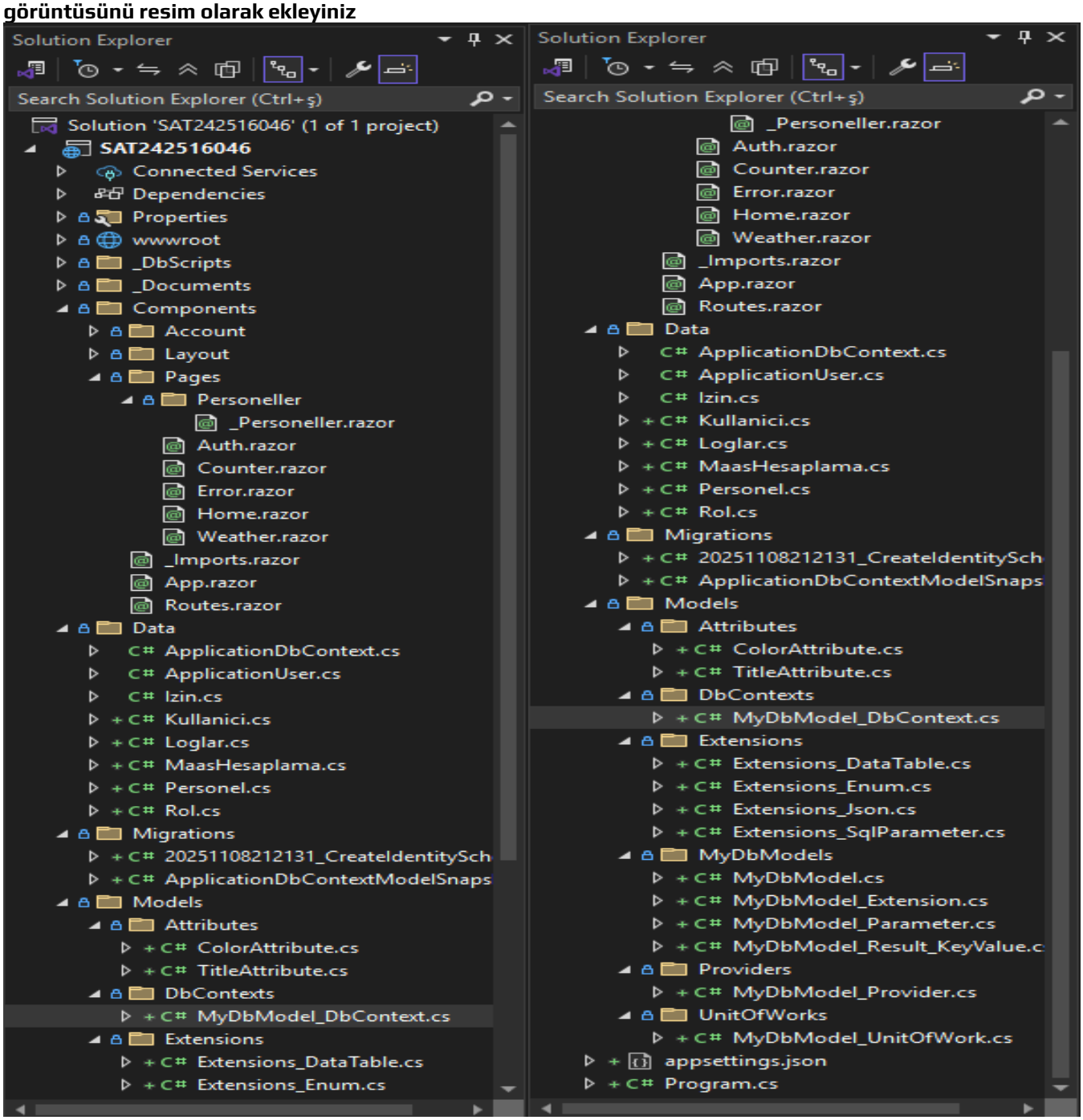
Proje Adı	Personel Maaş Sistemi
Proje Konusu	Personel, Rol, Kullanıcı, İzin ve Maaş Hesaplama İşlemleri
Proje Çözüm Adı	SAT242516046

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=SAT242516046; 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.

```
@page "/personeller"
@using SAT242516046.Data
@using SAT242516046.Models.MyDbModels
@using SAT242516046.Models.Providers
```

```
@using SAT242516046.Models.Extensions
@using System.ComponentModel.DataAnnotations

<h3 class="mb-4">Personel Yönetim Sistemi</h3>

@if (!string.IsNullOrEmpty(Model.Message))
{
    <div class="alert alert-danger">SQL Hatası: @Model.Message</div>
}
@if (IsSuccess)
{
    <div class="alert alert-success">İşlem Başarılı! Liste güncellendi.</div>
}




@if (OperationMode != Operation.None)
{    <div class="card p-4 mb-4">        <h4 class="text-@(GetColor(OperationMode))">
            @(OperationMode == Operation.Add ? "Yeni Personel Ekle" :
              OperationMode == Operation.Update ? "Personel Güncelle" : "Personel Sil")
        </h4>

        <EditForm Model="@CurrentPersonel" OnValidSubmit="@HandleSubmit">
            <DataAnnotationsValidator />

            <div class="form-group mb-3">
                <label>Ad</label>
                <InputText class="form-control" @bind-Value="CurrentPersonel.Ad" />
            </div>

            <div class="form-group mb-3">
                <label>Soyad</label>
                <InputText class="form-control" @bind-Value="CurrentPersonel.Soyad" />
            </div>

            <div class="d-flex justify-content-end">
                <button type="button" class="btn btn-secondary me-2" @onclick="() => OperationMode =
Operation.None">İptal</button>
                <button type="submit" class="btn btn-@(GetColor(OperationMode))">
                    @(OperationMode == Operation.Add ? "Ekle" : OperationMode == Operation.Update ?
"Güncelle" : "Sil")
                </button>
            </div>
        </EditForm>
    </div>
}

<div class="d-flex justify-content-between align-items-center mb-3">
    <h4>Personel Listesi</h4>
    <button class="btn btn-success" @onclick="() => StartOperation(Operation.Add,
null)">Ekle</button>
</div>

@if (Model.Items != null && Model.Items.Any())
{
    <table class="table table-striped table-bordered">
        <thead>
            <tr>
                <th>ID</th>
                <th>Ad Soyad</th>
                <th>Maaş</th>
                <th>İşlem</th>
            </tr>
        </thead>
        <tbody>
```

```
            @foreach (var personel in Model.Items)
            {
                <tr>
                    <td>@personel.PersonelID</td>
                    <td>@personel.Ad @personel.Soyad</td>
<td>@personel.BrutMaas.ToString("C")</td>                    <td>
<button class="btn btn-sm btn-warning me-2" @onclick="() => StartOperation(Operation.Update,
personel)">Güncelle</button>                    <button class="btn btn-sm btn-danger" @onclick="() =>
StartOperation(Operation.Remove, personel)">Sil</button>
                    </td>
                </tr>
            }
        </tbody>
    </table>
}
else if (OperationMode == Operation.None)
{
    <p><em>Gösterilecek personel bulunamadı.</em></p>
}
```

```
@code {
    public enum Operation { None, Add, Update, Remove }

    public IMyDbModel<Personel> Model { get; set; } = new
    MyDbModel<Personel>();
    public Personel CurrentPersonel { get; set; } = new Personel();
    public Operation OperationMode { get; set; } = Operation.None;
    public bool IsSuccess { get; set; }

    protected override async Task OnInitializedAsync()
    {
        await LoadPersoneller();
    }

    private async Task LoadPersoneller()
    {
        IsSuccess = false;
        Model = await DbProvider.Execute(Model, "sp_Personel_Listele");

        if (!string.IsNullOrEmpty(Model.Message))
        {
            Model.Items = new List<Personel>();
        }
    }

    private void StartOperation(Operation mode, Personel? personel)
    {
        OperationMode = mode;
        IsSuccess = false;

        if (personel != null)
        {
            CurrentPersonel = new Personel
            {
                PersonelID = personel.PersonelID,
                Ad = personel.Ad,
                Soyad = personel.Soyad,
                Departman = personel.Departman,
                BrutMaas = personel.BrutMaas
            };
        }
        else
        {
            CurrentPersonel = new Personel();
        }
    }

    private async Task HandleSubmit()
    {
        string operationName = OperationMode.ToString().ToLower();
        var jsonString = new List<Personel> { CurrentPersonel }.ListToJson();

        var resultModel = await DbProvider.Execute<MyDbModel_Result_KeyValue<string, int>>(
            "sp_Personel_EkleGuncelleSil",
            ("operation", operationName),
            ("jsonvalues", jsonString));

        if (resultModel.Items != null && resultModel.Items.FirstOrDefault()?.Value == 1)
        {
            IsSuccess = true;
        }
        else
        {
            Model.Message = resultModel.Message ?? "İşlem başarısız oldu.";
        }
```

```
        OperationMode = Operation.None;
        await LoadPersoneller();
    }

    private string GetColor(Operation mode) => mode switch
    {
        Operation.Add => "success",
        Operation.Update => "warning",
        Operation.Remove => "danger",
        _ => "primary"
    }; }
```