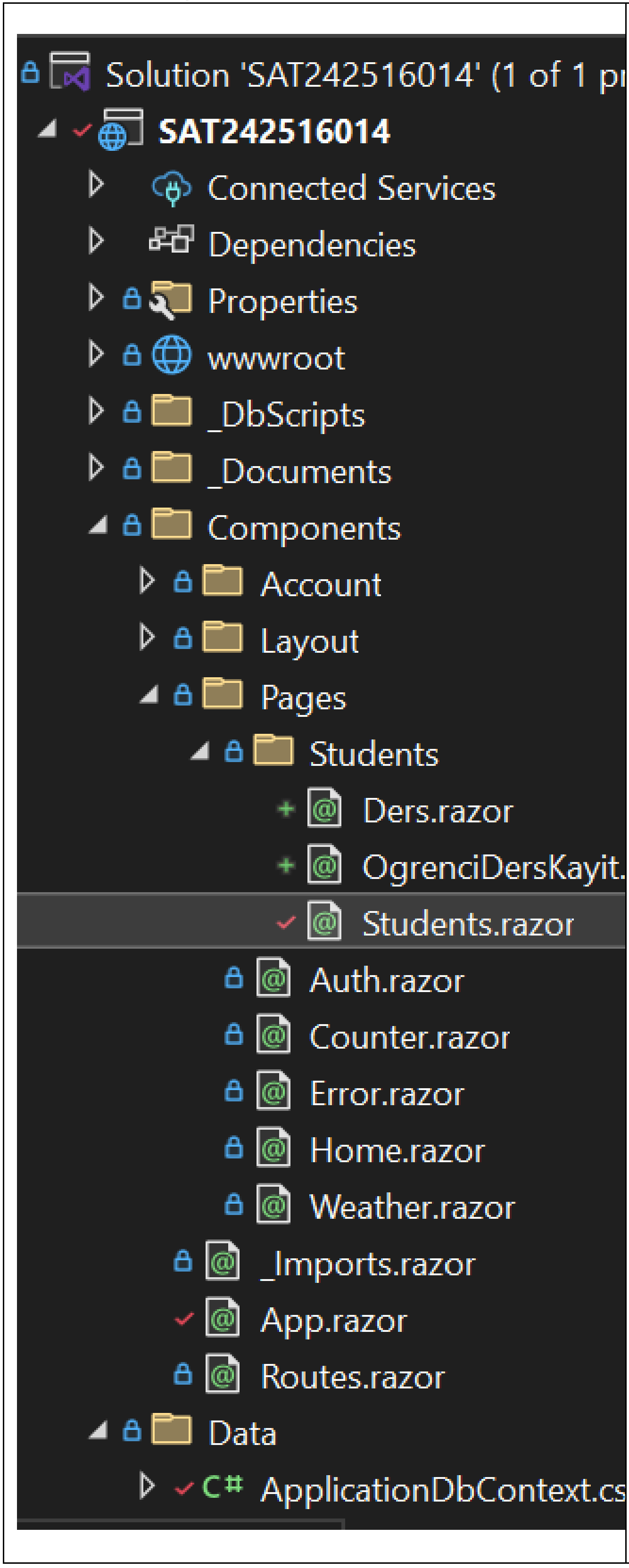
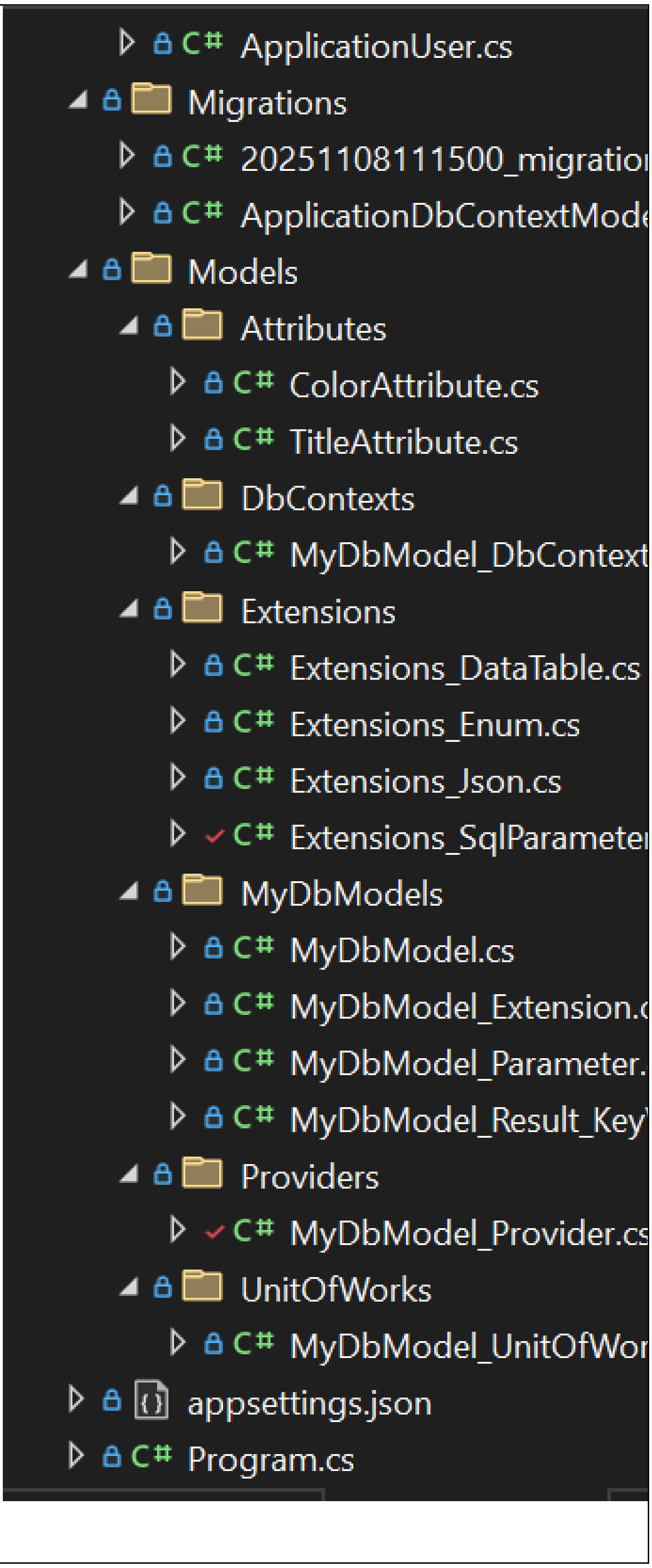
Proje Adı	Eğitim Kurumu Kayıt Sistemi
Proje Konusu	Yeni Öğrenci Başvuru ve Kayıt Süreci
Proje Çözüm Adı	SAT242516014

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



```
Solution 'SAT242516014' (1 of 1 pr
  SAT242516014
    Connected Services
    Dependencies
    Properties
    wwwroot
    _DbScripts
    _Documents
    Components
      Account
      Layout
      Pages
        Students
          Ders.razor
          OgrenciDersKayit.
          Students.razor
        Auth.razor
        Counter.razor
        Error.razor
        Home.razor
        Weather.razor
      _Imports.razor
      App.razor
      Routes.razor
    Data
      C# ApplicationDbContext.cs
```



```
      C# ApplicationUser.cs
    Migrations
      C# 20251108111500_migratio
      C# ApplicationDbContextMode
    Models
      Attributes
        C# ColorAttribute.cs
        C# TitleAttribute.cs
      DbContexts
        C# MyDbModel_DbContext
      Extensions
        C# Extensions_DataTable.cs
        C# Extensions_Enum.cs
        C# Extensions_Json.cs
        C# Extensions_SqlParamete
      MyDbModels
        C# MyDbModel.cs
        C# MyDbModel_Extension.c
        C# MyDbModel_Parameter.
        C# MyDbModel_Result_Key
      Providers
        C# MyDbModel_Provider.cs
      UnitOfWorks
        C# MyDbModel_UnitOfWor
    appsettings.json
    C# Program.cs
```

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 class MyDbModel_Result_KeyValue<TKey, TValue> :
IMyDbModel_Result_KeyValue<TKey, TValue>
{
    public TKey Key { get; set; }
    public 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 sealed class MyDbModel<T> : IMyDbModel<T> where T : class, new()
{
    public MyDbModel() : this(1, 10, "")
    {
    }

    public MyDbModel(int pageNumber, int pageSize, string orderBy)
    {
        Parameters = MyDbModel_Parameter.Create(pageNumber, pageSize, orderBy);
        OrderByItems = this.GetOrderByItems();
        Items = new List<T>();
    }

    public IMyDbModel_Parameter Parameters { get; set; }
    public IDictionary<object, object> OrderByItems { get; set; }
    public IEnumerable<T> Items { get; set; }
    public string Message { get; set; }
}
```

```
internal sealed class MyDbModel_Parameter : IMyDbModel_Parameter
{
    public static MyDbModel_Parameter Create(int pageNumber, int pageSize, string
orderBy) => new(pageNumber, pageSize, orderBy);
    private MyDbModel_Parameter(int pageNumber, int pageSize, string orderBy)
    {
        PageNumber = pageNumber;
        PageSize = pageSize;
        OrderBy = orderBy;

        if (Params == null) Params = new Dictionary<string, object>();
        if (Where == null) Where = new Dictionary<string, string>();
    }

    public int PageNumber { get; set; }
    public int PageSize { get; set; }
    public int TotalRecordCount { get; set; }
    public int TotalPageCount => (int)Math.Ceiling(TotalRecordCount /
(double)(PageSize <= 0 ? 1 : PageSize));
    public string OrderBy { get; set; }
    public IDictionary<string, object> Params { get; set; }
    public IDictionary<string, string> Where { get; set; }
}
```

```
public class MyDbModel_Result_KeyValue<TKey, TValue> :
IMyDbModel_Result_KeyValue<TKey, TValue>
{
    public TKey Key { get; set; }
    public TValue Value { get; set; }
}
```

```
public static class MyDbModel_Extension
{
    public static IDictionary<object, object> GetOrderByItems<E>(this MyDbModel<E>
myDbModel) where E : class, new()
    {
        var sortByItems = new Dictionary<object, object>();

        return sortByItems;
    }

}
```

```
public class MyDbModel_Result_KeyValue<TKey, TValue> :
IMyDbModel_Result_KeyValue<TKey, TValue>
{
    public TKey Key { get; set; }
    public TValue Value { get; set; }
}
```

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

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


    public async ValueTask<IEnumerable<TResult>> GetItems<TResult>(string spName =
"",
        params (string Key, object Value)[] parameters)
        where TResult : class, new() =>
        (await this.Execute<TResult>(spName, parameters)).Items;


    public async ValueTask<IEnumerable<TResult>> SetItems<TResult>(string spName =
"",
        params (string Key, object Value)[] parameters)
        where TResult : class, new() =>
        (await this.Execute<TResult>(spName, parameters)).Items;
```

```
public sealed class MyDbModel_UnitOfWork<TDbContext>(TDbContext context) :
IMyDbModel_UnitOfWork where TDbContext : DbContext
{
    private readonly DbContext _context = context;

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

```
public class MyDbModel_DbContext(DbContextOptions<MyDbModel_DbContext> options) :
DbContext(options)
{
}
```

```
public static class Extensions_DataTable
{
    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 class Extensions_Enum
{
    public static string Color<T>(this T value)
    {
    }

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

```
public static class Extensions_Json
{
    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 class Extensions_SqlParameter
{

    #region ToSqlParameter_Table_Type_Dictionary
    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 class ColorAttribute(string color) : Attribute
{
    public string Color { get; set; } = color;
}
```

```
public class TitleAttribute(string title) : Attribute
{
    public string Title { get; set; } = title;
}
```

appsettings.json dosyasında “DefaultConnection” ifadesini projenize göre yazınız

```
"ConnectionStrings": {
  "DefaultConnection": "Server=localhost,1445;Database=SAT242516014;User
Id=sa;Password=Oo_454545;MultipleActiveResultSets=true;PersistSecurityInfo =
true;TrustServerCertificate = true;Encrypt = true;"
```

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

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

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

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

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

App.razor bileşeninde, gerekli C# kodlarını yazınız

```
<body>
    <Routes @rendermode="@RenderModeForPage"/>
    <script src="_framework/blazor.web.js"></script>
</body>


@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	Students

Tasarım Kodları

```
@page "/students"

<h1>Students</h1>

<hr />

@if (_myDbModel != null)
{
    <h4 class="text-@_operation.Color()">
        @_operation.Title()
    </h4>

    <hr />

    @if (!string.IsNullOrEmpty(_myDbModel.Message))
    {
        <h1 class="text-warning">
            @_myDbModel.Message
        </h1>
    }

    @if (_operation == Operations.List)
    {
        <table class="table table-striped">

            <thead>
                <tr>
                    <th>
                        <button class="btn btn-outline-
@Operations.Add.Color()"
                                @onclick="() =>
OnClick_Operation(Operations.Add)">
                            @Operations.Add.Title()
                        </button>
                    </th>
                    @foreach (var prop in GetProperties<Model>())
                    {
                        <th>@prop.Name</th>
                    }

                </tr>
            </thead>
            <tbody>
                @foreach (var item in _myDbModel.Items)
                {
                    <tr>
                        <td>
                            <div class="btn-group">
                                @foreach (var operation in new[] {
Operations.Update, Operations.Remove })
                                {
                                    <button class="btn btn-outline-
@operation.Color()"
                                            @onclick="() =>
OnClick_Operation(operation, item.Id)">
                                        @operation.Title()
                                    </button>
                                }
                            </div>
                        </td>

                        @foreach (var prop in GetProperties<Model>())
                        {
                            <td>
```

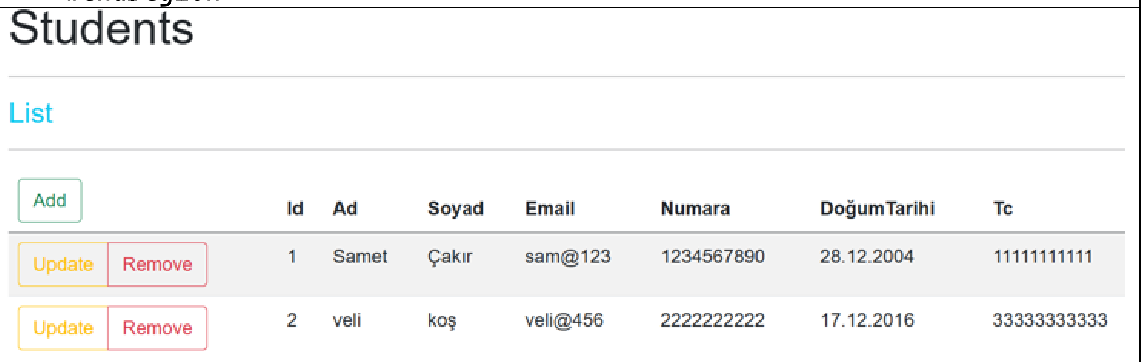
```
                                @if (prop.PropertyType ==
typeof(DateTime?))
                                {

@(((DateTime?)prop.GetValue(item))?.ToString("dd.MM.yyyy"))
                                }
                                else
                                {
                                    @(prop.GetValue(item))
                                }
                            </td>
                        }
                    </tr>
                }
            </tbody>
        </table>
    }

    @if (new[] { Operations.Add, Operations.Update, Operations.Remove
}.Contains(_operation))
    {
        @foreach (var prop in GetProperties<Model>().Where(p => p.Name
!= "Id"))
        {
            <div class="input-group mb-1" style="width:
300px!important;">
                <div class="input-group-text" style="width:
100px!important;">@prop.Name</div>

                @if (prop.PropertyType == typeof(DateTime?))
                {
                    <input type="date" class="form-control"
                           value="@(
((DateTime?)_model[prop.Name])?.ToString("yyyy-MM-dd") )"
                           @onchange="@((e) =>
OnChange_Input(prop.Name, e.Value))" />
                }
                else
                {
                    @switch (prop.Name)
                    {
                        case "Numara":
                            <input type="tel" class="form-control"
                                   value="@(_model[prop.Name])"
                                   maxlength="10"
                                   @oninput="@((e) =>
OnChange_Input(prop.Name, e.Value))" />
                            break;
                        case "Tc":
                            <input type="tel" class="form-control"
                                   value="@(_model[prop.Name])"
                                   maxlength="11"
                                   @oninput="@((e) =>
OnChange_Input(prop.Name, e.Value))" />
                            break;
                        default:
                            <input type="text" class="form-control"
                                   value="@(_model[prop.Name])"
                                   @onchange="@((e) =>
OnChange_Input(prop.Name, e.Value))" />
                            break;
                    }
                }

            </div>
        }

        <div class="btn-group" style="width: 300px!important;">

            <button class="btn btn-outline-@Operations.Cancel.Color()"
                    style="width: 100px!important;"
                    @onclick="() => OnClick_Cancel()">
                @Operations.Cancel.Title()
            </button>

            <button class="btn btn-@_operation.Color()"
                    style="width: 200px!important;"
                    @onclick="() => OnClick_Save()">
                @_operation.Title()
            </button>

        </div>
    }
}
```

C# Kodları

```
@using System.Reflection
@using Microsoft.JSInterop
@using System.Linq

@using Attributes
@using Extensions
@using MyDbModels
@using Providers


@inject IMyDbModel<Model> _myDbModel
@inject IMyDbModel_Provider _myDbModel_Provider
@inject IJSRuntime JS

@code {
    #region Fields

    Operations _operation = Operations.List;

    Model _model = new();

    #endregion

    #region OnAfterRenderAsync

    protected override async Task OnAfterRenderAsync(bool firstRender)
    {
        if (firstRender)
        {
            await GetItems();
        }
    }

    #endregion

    #region GetItems

    async Task GetItems()
    {
        _operation = Operations.List;

        await _myDbModel_Provider
            .Execute<Model>(_myDbModel, "sp_Students");

        StateHasChanged();
    }

    #endregion

    #region OnClick_ Input, Operation, Save, Cancel

    // GÜNCELLEME 2. BÖLÜM: 'Number' ve 'Tc' için sayısal filtreleme eklendi
    private void OnChange_Input(string name, object value)
    {
        var prop = typeof(Model).GetProperty(name);
        var stringValue = value?.ToString() ?? string.Empty; // Gelen değeri string'e çevir

        // Gelen değer bir tarih alanı içinse...
        if (prop.PropertyType == typeof(DateTime?))
        {
            if (DateTime.TryParse(stringValue, out DateTime dt))
            {
                prop.SetValue(_model, dt);
            }
            else
            {
                prop.SetValue(_model, null);
            }
        }
        // Number ve Tc alanları için sayısal filtreleme
        else if (name == "Numara" || name == "Tc")
        {
            // LINQ kullanarak string içindeki sadece rakamları al
            var filteredValue = new
string(stringValue.Where(char.IsDigit).ToArray());

            // HTML 'maxlength' zaten kısıtlıyor ama C# tarafında da
garantileyim
            if (name == "Numara" && filteredValue.Length > 10)
            {
                filteredValue = filteredValue.Substring(0, 10);
            }
            else if (name == "Tc" && filteredValue.Length > 11)
            {
                filteredValue = filteredValue.Substring(0, 11);
            }

            prop.SetValue(_model, filteredValue);
        }
        else
        {
            // Diğer tüm alanlar (Name, Surname, Email vb.)
            prop.SetValue(_model, stringValue);
        }
    }


    private void OnClick_Operation(Operations operation, int? id = 0)
    {
        _operation = operation;

        if (_operation == Operations.Add)
            _model = new Model();
        else
            _model = _myDbModel
                .Items
                .FirstOrDefault(f => f.Id == id) ?? new Model();
    }

    private async void OnClick_Save()
    {
        var jsonvalues = _model.ItemToJson();

        var results = await _myDbModel_Provider
            .SetItems<MyDbModel_Result_KeyValue<string,
bool>>("sp_Student_Add_Update_Remove"
                , ("operation", _operation.ToString().ToLower())
                , ("jsonvalues", jsonvalues)
            );

        var result = results.FirstOrDefault();

        await JS.InvokeVoidAsync("alert", $"{result.Key} :
{result.Value}");

        await GetItems();
    }
    private void OnClick_Cancel() => _operation = Operations.List;

    #endregion

    #region Models

    private IEnumerable<PropertyInfo> GetProperties<T>() where T :
class, new() =>
        typeof(T)
            .GetProperties()
            .Where(p => p.GetIndexParameters().Length == 0);

    // 1. ADIM: Model sınıfı GÜNCELLENDİ
    class Model
    {
        public int Id { get; set; }
        public string Ad { get; set; }
        public string Soyad { get; set; }
        public string Email { get; set; }
        public string Numara { get; set; }
        public DateTime? DoğumTarihi { get; set; } // string ->
DateTime?
        public string Tc { get; set; }

        public object this[string name]
        {
            get => this.GetType().GetProperty(name).GetValue(this);
            set => this.GetType().GetProperty(name).SetValue(this,
value);
        }
    }

    #endregion

    #region Enums

    enum Operations
    {
        //attribute
        [Color("info"), Title("List")] List,
        [Color("success"), Title("Add")] Add,
        [Color("warning"), Title("Update")] Update,
        [Color("danger"), Title("Remove")] Remove,
        [Color("dark"), Title("Cancel")] Cancel
    }

    #endregion
```

Ekran Görüntüleri

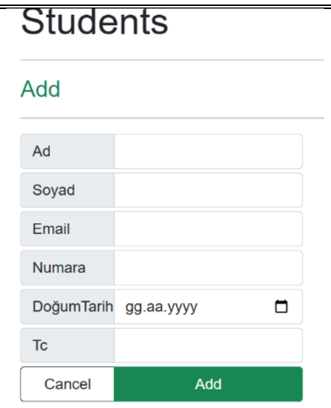
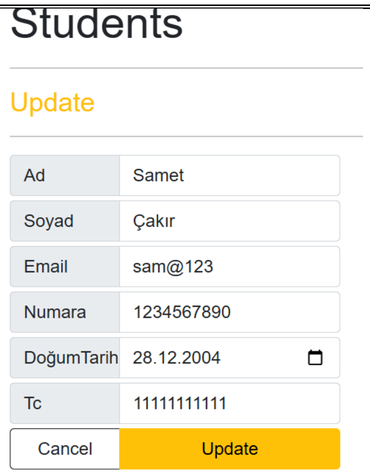
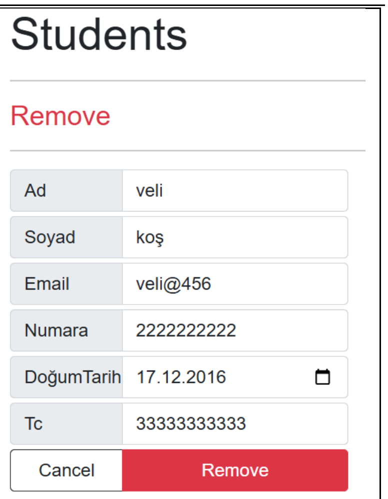
UYARI: Resim boyutlarının çok büyük olmamasına dikkat ediniz



Students

List

Add	Id	Ad	Soyad	Email	Numara	DoğumTarihi	Tc
Update Remove	1	Samet	Çakır	sam@123	1234567890	28.12.2004	11111111111
Update Remove	2	veli	koş	veli@456	2222222222	17.12.2016	33333333333

	 Students Add Ad Soyad Email Numara DoğumTarih gg.aa.yyyy Tc 	 Students Update Ad Samet Soyad Çakır Email sam@123 Numara 1234567890 DoğumTarih 28.12.2004 Tc 11111111111 	 Students Remove Ad veli Soyad koş Email veli@456 Numara 2222222222 DoğumTarih 17.12.2016 Tc 33333333333