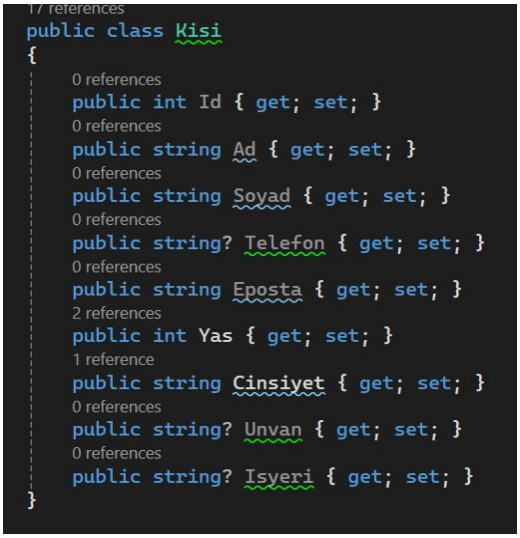
Eposta Gonderim Uygulaması

Bu projede ki kullandığım teknolojiler backend kısmında .Net Core 7.0, frontend kısmında React Js, javascript, Bootstrap, Meterial UI, Veritabanı olarak Postgresql kullandım.

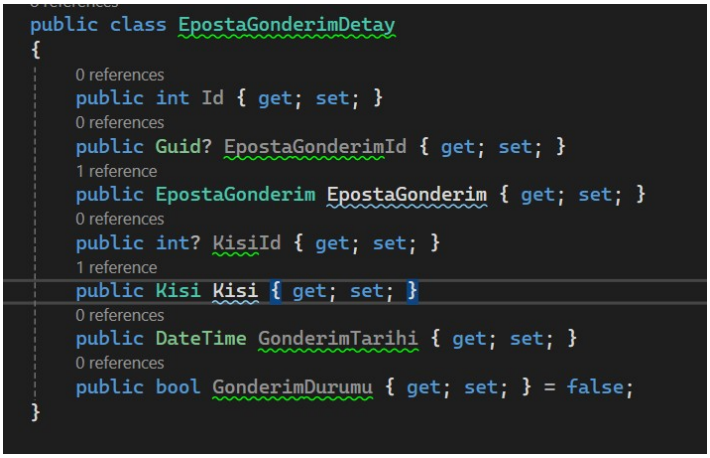
Katmanlı mimari, kurumsal yapı ile solid prensiblere uyarak clean code ile yazdım. Sunum katmanında web api kullandım, iş katmanında automapper, veri erişim katmanında Entity Framework kullandım.

Projemde Kisi, EpostaGonderimDetay, EpostaGonderim, EpostaAdresi entitylerimi oluşturdum bunların kaynak kodları aşağıda ki gibidir.

Kisi Entity:



EpostaGonderimDetay Entity:

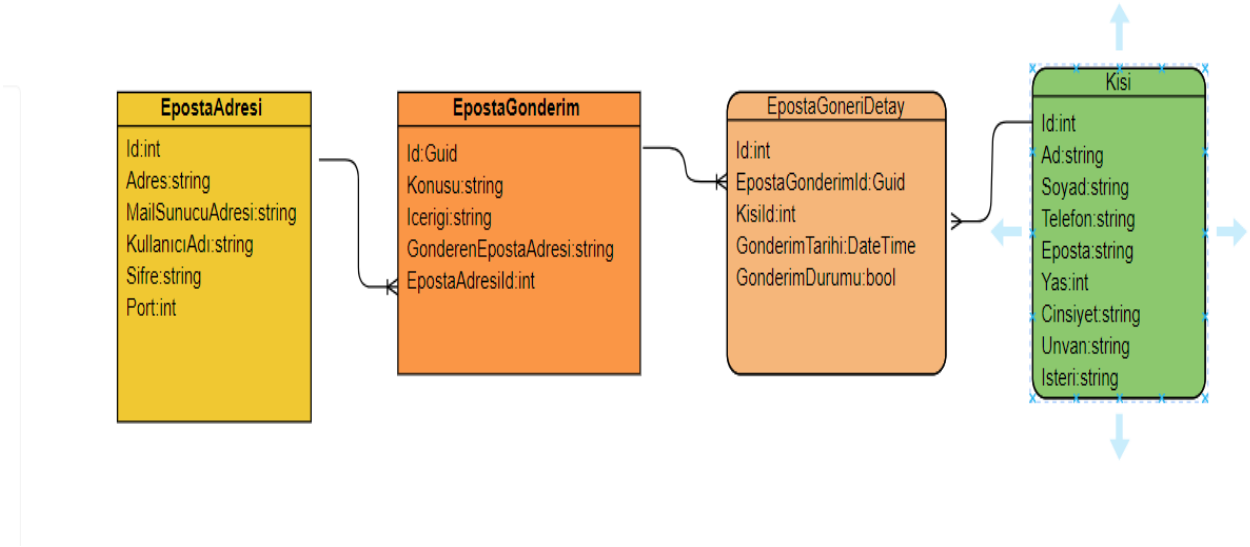


EpostaGonderim Entity:

EpostaAdresi Enttiy:

Entity, Entity Katmanında oluşturulduktan sonra dataaccess katmanında erişilmesi için bağımlılık ayarını yaptım böylece crud işlemler için kullanılacak hale getirdim.

Bu Entitylerin ilişkisi aşağıda ki gibidir:



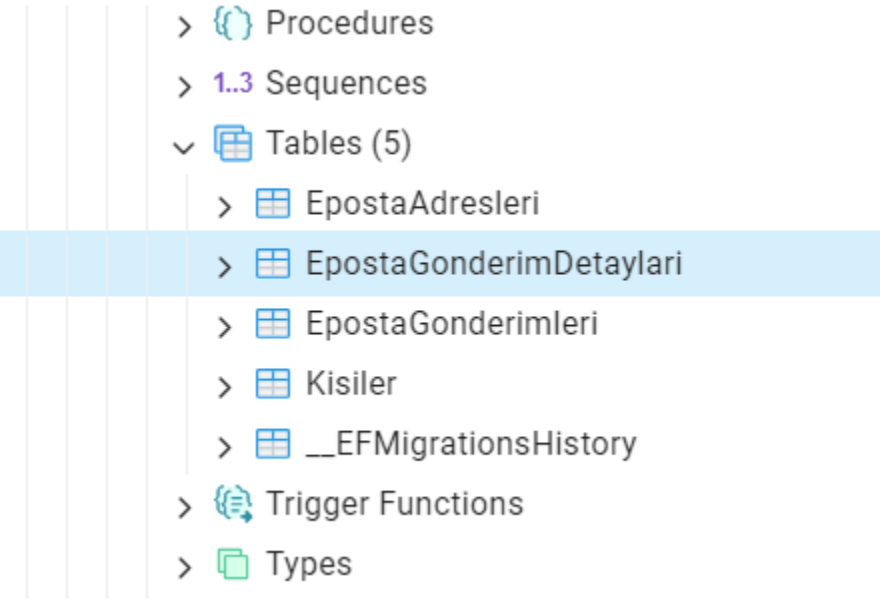
EpostaAdresi ile EpostaGonderim bire çok ilişkidedir. EpostaGonderim, Kisi arasında çoka çok ilişki mevcuttur.

DataAccess Katmanı:

EntityFramework kullandım, veri erişim katmanım şu şekildedir:

```
namespace PiDataEmailApp.DataAccess.Concrete
{
    16 references
    public class PiDataDbContext:DbContext
    {
        0 references
        public PiDataDbContext(DbContextOptions<PiDataDbContext> options):base(options)
        {

        }


        1 reference
        public DbSet<Kisi> Kisiler { get; set; }
        0 references
        public DbSet<EpostaAdresi> EpostaAdresleri { get; set; }
        0 references
        public DbSet<EpostaGonderim> EpostaGonderimleri { get; set; }
        1 reference
        public DbSet<EpostaGonderimDetay> EpostaGonderimDetaylari { get; set; }

    }
```

veritabanımda Kisiler, EpostaAdresleri, EpostaGonderimleri, EpostaGonderimDetayları tabloları bulunmaktadır.



Postgresql'deki görüntüsü bu şekildedir.

Data Erişim katmanında soyut ve somut sınıflar oluşturdum ve interfacelerden yaralandım.

Kaynak kodum aşağıda ki gibidir.

Interface:

```
public interface IEntityRepository<T> where T : class, new()
{
    Task<List<T>> GetAll(Expression<Func<T, bool>>? filter = null);
    T? Get(Expression<Func<T, bool>>? filter);
    T? GetById(int id);
    Task Add(T entity);
    void AddList(List<T> entityList);
    void Update(int id, T entity);
    Task Delete(int id);

}
```

Somut class:

```
public class EfEntityRepository<T, TContext> : IEntityRepository<T>
    where T : class, new()
    where TContext : DbContext
{
    private readonly TContext _context;

    public EfEntityRepository(TContext context)
    {
        _context = context;
    }

    public async Task<List<T>> GetAll(Expression<Func<T, bool>>? filter)
    {
        return filter == null
            ? await _context.Set<T>().ToListAsync()
            : await _context.Set<T>().Where(filter).ToListAsync();
    }

    public T? Get(Expression<Func<T, bool>>? filter = null)
    {
        return filter == null
            ? _context.Set<T>().FirstOrDefault()
            : _context.Set<T>().FirstOrDefault(filter);
    }

    public T? GetById(int id)
    {
        return _context.Set<T>().Find(id);
```

```
}

public async Task Add(T entity)
{

   _context.Entry(entity).State = EntityState.Added;
   await _context.SaveChangesAsync();
}

public void AddList(List<T> entityList)
{
   try
   {
      _context.AddRange(entityList);
      _context.SaveChanges();
   }
   catch (Exception ex)
   {
      Console.WriteLine(ex.Message);
   }
}

public async void Update(int id, T entity)
{
   _context.Entry(entity).State = EntityState.Modified;
   await _context.SaveChangesAsync();
}

public async Task Delete(int id)
```

```
    {
        var entity = GetById(id);
        if (entity != null) _context.Set<T>().Remove(entity);
        await _context.SaveChangesAsync();
    }
}
```

Generic Repositor ile kod tekrarında kurtuldum, Bu yapıya diğer Dallar erişim sağladı bunlar sırası ile

IEpostaAdresiDal:

```
public interface IEpostaAdresiDal:IEntityRepository<EpostaAdresi>
{
 }
```

IEpostaGonderimDal:

```
public interface IEpostaGonderimDal:IEntityRepository<EpostaGonderim>
{

}
```

IEpostaGonderimDetayDal:

```
public interface IEpostaGonderimDetayDal:IEntityRepository<EpostaGonderimDetay>
{
    Task<List<EpostaGonderimDetay>> GetAllWithIncludesAsync();
}
```

IKisiDal:

```
public interface IKisiDal:IEntityRepository<Kisi>
{
    Task<List<Kisi>> GetAllByFilter(string? cinsiyet, int? yasMin, int? yasMax);

}
```

somut sınıflarım ile bu interfaceleri kullandım.

EpostaAdresiDal:

```
public class
EpostaAdresiDal:EfEntityRepository<EpostaAdresi,PiDataDbContext>,IEpostaAdresiDal

{

    public EpostaAdresiDal(PiDataDbContext context) : base(context)

    {

    }

}
```

EpostaGonderimDal:

```
public class
EpostaGonderimDal:EfEntityRepository<EpostaGonderim,PiDataDbContext>,IEpostaGonderimDal

{

    public EpostaGonderimDal(PiDataDbContext context) : base(context)

    {

    }

}
```

EpostaGonderimDetayDal:

```
public class
EpostaGonderimDetayDal:EfEntityRepository<EpostaGonderimDetay,PiDataDbContext>,IEposta
GonderimDetayDal

{

    private readonly PiDataDbContext _context;

    public EpostaGonderimDetayDal(PiDataDbContext context) : base(context)

    {

        _context = context;

    }



    public Task<List<EpostaGonderimDetay>> GetAllWithIncludesAsync()

    {

        return _context.EpostaGonderimDetaylari
```

```
            .Include(x => x.EpostaGonderim)
            .Include(x => x.Kisi)
            .ToListAsync();
    }
}
```

KisiDal :

```
public class KisiDal : EfEntityRepository<Kisi, PiDataDbContext>, IKisiDal
{
    private readonly PiDataDbContext _context;
    public KisiDal(PiDataDbContext context) : base(context)
    {
        _context = context;

    }

    public async Task<List<Kisi>> GetAllByFilter(string? cinsiyet, int? yasMin, int? yasMax)
    {
        return await _context.Kisiler.Where(x =>
            (cinsiyet == null || x.Cinsiyet == cinsiyet)
            && (yasMin == null || yasMin == 0 || x.Yas >= yasMin)
            && (yasMax == null || yasMax == 0 || x.Yas <= yasMax))
            .ToListAsync();

    }
}
```

Complext Type kullanarak filtreleme yaptım ve diğer tabloları birleştirerek raporlama saydfasını yaptım.

İş Katmanı:

Aynı şekilde burada da somut ve soyut sınıflardan yararlandım.

Burada ki sınıflarım ve interfacelerim şu şekildedir.

:IEpostaAdresiService

```
public interface IEpostaAdresiService

{

    Task<ResponseModel<List<EpostaAdresiModel>>> GetAll();

    ResponseModel<EpostaAdresiModel> GetById(int id);

    ResponseModel<EpostaAdresiModel> Create(EpostaAdresiModel model);

    ResponseModel<NoContentModel> CreateList(List<EpostaAdresiModel> model);

    ResponseModel<EpostaAdresiModel> Update(int id, EpostaAdresiModel model);

    Task Delete(int id);

}
```

IEpostaGonderimDetayService:

```
 public interface IEpostaGonderimDetayService

 {

     Task<ResponseModel<List<EpostaGonderimDetayModel>>> GetAll();

     Task<ResponseModel<List<EpostaGonderimDetayWithIncludeModel>>>
GetAllWithIncludesAsync();

     ResponseModel<EpostaGonderimDetayModel> GetById(int id);

     Task Delete(int id);

 }
```

IEpostaGonderimService:

```
public interface IEpostaGonderimService

{

    Task<ResponseModel<List<EpostaGonderimModel>>> GetAll();

    ResponseModel<EpostaGonderimModel> GetById(int id);

    Task<ResponseModel<EpostaGonderimModel>> Create(EpostaGonderimModel model);

    ResponseModel<NoContentModel> CreateList(List<EpostaGonderimModel> model);
```

```
    ResponseModel<EpostaGonderimModel> Update(int id, EpostaGonderimModel model);
    Task Delete(int id);
}
```

IKisiService:

```
public interface IKisiService
{
    Task<ResponseModel<List<Kisi>>> GetAll();
    Task<ResponseModel<List<Kisi>>> GetAllByFilter(string? cinsiyet, int? yasMin, int? yasMax);
    ResponseModel<KisiModel> GetById(int id);
    ResponseModel<KisiModel> Create(KisiModel model);
    ResponseModel<NoContentModel> CreateList(List<KisiModel> model);
    ResponseModel<KisiModel> Update(int id, KisiModel model);
    Task Delete(int id);
}
```

somut sınıflarım ise:

EpostaAdresiManager:

```
public class EpostaAdresiManager:IEpostaAdresiService
{
    private readonly IEpostaAdresiDal _epostaAdresiDal;
    private readonly IMapper _mapper;

    public EpostaAdresiManager(IEpostaAdresiDal epostaAdresiDal,
        IMapper mapper)
    {
        _epostaAdresiDal = epostaAdresiDal;
        _mapper = mapper;
    }
```

```
public async Task<ResponseModel<List<EpostaAdresiModel>>> GetAll()
{
    var response=await _epostaAdresiDal.GetAll();
    var result=_mapper.Map<List<EpostaAdresiModel>>(response);
    return await ResponseModel<List<EpostaAdresiModel>>.SuccessAsync(result,200);
}

public ResponseModel<EpostaAdresiModel> GetById(int id)
{
    var response = _epostaAdresiDal.GetById(id);
    var result = _mapper.Map<EpostaAdresiModel>(response);
    return ResponseModel<EpostaAdresiModel>.Success(result, 200);
}

public ResponseModel<EpostaAdresiModel> Create(EpostaAdresiModel model)
{
    var response = _mapper.Map<EpostaAdresi>(model);
    var result=_epostaAdresiDal.Add(response);
    return ResponseModel<EpostaAdresiModel>.Success(model, 200);
}

public ResponseModel<NoContentModel> CreateList(List<EpostaAdresiModel> model)
{
    var response = _mapper.Map<List<EpostaAdresi>>(model);
    _epostaAdresiDal.AddList(response);
    return ResponseModel<NoContentModel>.Success(204);
}

public ResponseModel<EpostaAdresiModel> Update(int id, EpostaAdresiModel model)
```

```
    {
        var response = _mapper.Map<EpostaAdresi>(model);
        _epostaAdresiDal.Update(id, response);
        return ResponseModel<EpostaAdresiModel>.Success(model, 200);
    }

    public Task Delete(int id)
    {
        return _epostaAdresiDal.Delete(id);
    }
}
```

EpostaGonderimDetayManager:

```
    public class EpostaGonderimDetayManager:IEpostaGonderimDetayService
    {
        private readonly IEpostaGonderimDetayDal _epostaGonderimDetayDal;
        private readonly IMapper _mapper;

        public EpostaGonderimDetayManager(IEpostaGonderimDetayDal epostaGonderimDetayDal,
            IMapper mapper)
        {
            _epostaGonderimDetayDal = epostaGonderimDetayDal;
            _mapper = mapper;
        }
        public async Task<ResponseModel<List<EpostaGonderimDetayModel>>> GetAll()
        {
            var response = await _epostaGonderimDetayDal.GetAll();
            var result = _mapper.Map<List<EpostaGonderimDetayModel>>(response);
            return await ResponseModel<List<EpostaGonderimDetayModel>>.SuccessAsync(result,
200);
```

```
        }


        public async Task<ResponseModel<List<EpostaGonderimDetayWithIncludeModel>>>
GetAllWithIncludesAsync()

        {

            var response = await _epostaGonderimDetayDal.GetAllWithIncludesAsync();

            var result = _mapper.Map<List<EpostaGonderimDetayWithIncludeModel>>(response);

            return await
ResponseModel<List<EpostaGonderimDetayWithIncludeModel>>.SuccessAsync(result, 200);

        }


        public ResponseModel<EpostaGonderimDetayModel> GetById(int id)

        {

            var response = _epostaGonderimDetayDal.GetById(id);

            var result = _mapper.Map<EpostaGonderimDetayModel>(response);

            return ResponseModel<EpostaGonderimDetayModel>.Success(result, 200);

        }


        public Task Delete(int id)

        {

            return _epostaGonderimDetayDal.Delete(id);

        }

    }

}
```

EpostaGonderimManager:

```
public class EpostaGonderimManager : IEpostaGonderimService

{

    private readonly IEpostaGonderimDal _epostaGonderimDal;

    private readonly IEpostaGonderimDetayDal _epostaGonderimDetayDal;
```

```
private readonly IEpostaAdresiDal _epostaAdresiDal;
private readonly IMapper _mapper;

public EpostaGonderimManager(IEpostaGonderimDal epostaGonderimDal,
    IEpostaGonderimDetayDal epostaGonderimDetayDal,
    IEpostaAdresiDal epostaAdresiDal,
    IMapper mapper)
{
    _epostaGonderimDal = epostaGonderimDal;
    _epostaGonderimDetayDal = epostaGonderimDetayDal;
    _epostaAdresiDal = epostaAdresiDal;
    _mapper = mapper;
}
public async Task<ResponseModel<List<EpostaGonderimModel>>> GetAll()
{
    var response = await _epostaGonderimDal.GetAll();
    var result = _mapper.Map<List<EpostaGonderimModel>>(response);
    return await ResponseModel<List<EpostaGonderimModel>>.SuccessAsync(result, 200);

}

public ResponseModel<EpostaGonderimModel> GetById(int id)
{
    var response = _epostaGonderimDal.GetById(id);
    var result = _mapper.Map<EpostaGonderimModel>(response);
    return ResponseModel<EpostaGonderimModel>.Success(result, 200);
}
```

```
    public async Task<ResponseModel<EpostaGonderimModel>> Create(EpostaGonderimModel
model)

    {

        var gonderenEmailId = Guid.NewGuid();

        int epostaAdresiId =_epostaAdresiDal.Get(x => x.Adres == model.GonderenEmail)!.Id;

        EpostaGonderim epostaGonderim = new EpostaGonderim

        {

            Id=gonderenEmailId,

            Konusu = model.Konu,

            Icerigi = model.Icerik,

            GonderenEpostaAdresi = model.GonderenEmail,

            EpostaAdresiId = epostaAdresiId

        };

        await _epostaGonderimDal.Add(epostaGonderim);



        foreach (var id in model.KisiListesiIds)

        {

            EpostaGonderimDetayModel epostaGonderimDetayModel = new
EpostaGonderimDetayModel

            {

                EpostaGonderimId =gonderenEmailId,

                KisiId = id,

                GonderimDurumu = true,

                GonderimTarihi = DateTime.UtcNow



            };

            var epostaGonderimDetay =
_mapper.Map<EpostaGonderimDetay>(epostaGonderimDetayModel);

            await _epostaGonderimDetayDal.Add(epostaGonderimDetay);

        }
```

```
        return await ResponseModel<EpostaGonderimModel>.SuccessAsync(model, 200);
    }

    public ResponseModel<NoContentModel> CreateList(List<EpostaGonderimModel> model)
    {
        var response = _mapper.Map<List<EpostaGonderim>>(model);
        _epostaGonderimDal.AddList(response);
        return ResponseModel<NoContentModel>.Success(204);
    }

    public ResponseModel<EpostaGonderimModel> Update(int id, EpostaGonderimModel model)
    {
        var response = _mapper.Map<EpostaGonderim>(model);
        _epostaGonderimDal.Update(id, response);
        return ResponseModel<EpostaGonderimModel>.Success(model, 200);
    }

    public Task Delete(int id)
    {
        return _epostaGonderimDal.Delete(id);
    }
}
```

KisiManager:

```
public class KisiManager: IKisiService
{
    private readonly IKisiDal _kisiDal;
    private readonly IMapper _mapper;
```

```
    public KisiManager(IKisiDal kisiDal,
       IMapper mapper)
    {
       _kisiDal = kisiDal;
       _mapper = mapper;
    }

    public async Task<ResponseModel<List<Kisi>>> GetAll()
    {
       var response=await _kisiDal.GetAll();
       return await ResponseModel<List<Kisi>>.SuccessAsync(response,200);
    }

    public async Task<ResponseModel<List<Kisi>>> GetAllByFilter(string? cinsiyet, int? yasMin,
int? yasMax)
    {
       var response = await _kisiDal.GetAllByFilter(cinsiyet, yasMin, yasMax);
       return await ResponseModel<List<Kisi>>.SuccessAsync(response, 200);

    }

    public ResponseModel<KisiModel> GetById(int id)
    {
       var response = _kisiDal.GetById(id);
       var result = _mapper.Map<KisiModel>(response);
       return ResponseModel<KisiModel>.Success(result, 200);
    }
```

```
    public ResponseModel<KisiModel> Create(KisiModel model)
    {
        var response = _mapper.Map<Kisi>(model);
        _kisiDal.Add(response);
        return ResponseModel<KisiModel>.Success(model, 200);
    }

    public ResponseModel<NoContentModel> CreateList(List<KisiModel> model)
    {
        var response = _mapper.Map<List<Kisi>>(model);
        _kisiDal.AddList(response);
        return ResponseModel<NoContentModel>.Success(204);
    }

    public ResponseModel<KisiModel> Update(int id, KisiModel model)
    {
        var response = _mapper.Map<Kisi>(model);
        _kisiDal.Update(id, response);
        return ResponseModel<KisiModel>.Success(model, 200);
    }

    public Task Delete(int id)
    {
        return _kisiDal.Delete(id);
    }
}
```

isterleri karşılamak için yazdığım iş katmanı kodlarım bunlardır. Modeller ile bu verilen iş katmanı sınıflarını tamamladım.

Geriye dönüş sağlayan verileri response model kullanarak anlamlı verilere dönüştürdüm yani başarılı ve hata durumlarında uyarı göstermesini sağladım.

```
public class ResponseModel<T>

{

    public T Data { get; set; }

    public List<string> Errors { get; set; }


    [JsonIgnore]

    public int StatusCode { get; set; }


    public static Task<ResponseModel<T>> SuccessAsync(T data, int statusCode)

    {

        return Task.FromResult(new ResponseModel<T> { Data = data, StatusCode =
statusCode });

    }


    public static ResponseModel<T> Success(T data, int statusCode)

    {

        return new ResponseModel<T> { Data = data, StatusCode = statusCode };

    }


    public static ResponseModel<T> Success(int statusCode)

    {

        return new ResponseModel<T> { StatusCode = statusCode };

    }


    public static ResponseModel<T> Fail(int statuscode, List<string> errors)

    {

        return new ResponseModel<T> { StatusCode = statuscode, Errors = errors };
```

```
    }


    public static ResponseModel<T> Fail(int statuscode, string error)

    {

        return new ResponseModel<T> { StatusCode = statuscode, Errors = new List<string>
{ error } };

    }


    public static Task<ResponseModel<T>> SuccessAsync(int statusCode)

    {

        return Task.FromResult(new ResponseModel<T> { StatusCode = statusCode });

    }


    public static Task<ResponseModel<T>> FailAsync(int statuscode, List<string> errors)

    {

        return Task.FromResult(new ResponseModel<T> { StatusCode = statuscode, Errors =
errors });

    }


    public static Task<ResponseModel<T>> FailAsync(int statuscode, string error)

    {

        return Task.FromResult(

            new ResponseModel<T> { StatusCode = statuscode, Errors = new List<string>
{ error } });

    }

}
```

Sunum Katmanı:

Bu katman ile veritabanından alınan bilgileri api ile interface uygulamama gönderdim, interface uygulamamda ise form vasıtası ile aldığım bilgileri veri tabanına kaydettim. Burada ki controllerlarım şu şekilde dir.

EpostaAdresiController:

```
[Route("api/[controller]/[action]")]

[ApiController]

public class EpostaAdresiController : ControllerBase

{

    private readonly IEpostaAdresiService _epostaAdresiService;



    public EpostaAdresiController(IEpostaAdresiService epostaAdresiService)

    {

        _epostaAdresiService = epostaAdresiService;

    }



    [HttpGet]

    public async Task<IActionResult> EpostaAdresleri()

    {

        var response = await _epostaAdresiService.GetAll();

        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };

    }



    [HttpGet("{id}")]

    public IActionResult EpostaAdresi(int id)

    {

        var response = _epostaAdresiService.GetById(id);

        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
```

```
}
[HttpPost]
public IActionResult EpostaAdresiEkle([FromBody] EpostaAdresiModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    var response = _epostaAdresiService.Create(model);
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
}

[HttpPost]
public IActionResult EpostaAdresiEkleList([FromBody] List<EpostaAdresiModel> model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    var response = _epostaAdresiService.CreateList(model);
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
}

[HttpPut("{id}")]
public IActionResult EpostaAdresiGuncelle(int id, [FromBody] EpostaAdresiModel model)
{
    var response = _epostaAdresiService.Update(id, model);
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
```

```
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> EpostaAdresiSil(int id)
    {
        await _epostaAdresiService.Delete(id);
        return new StatusCodeResult(StatusCodes.Status204NoContent);
    }
}
```

EpostaAdresiGonderimController:

```
[Route("api/[controller]/[action]")]
[ApiController]
public class EpostaAdresiGonderimController : ControllerBase
{
    private readonly IEpostaGonderimService _epostaGonderimService;

    public EpostaAdresiGonderimController(IEpostaGonderimService epostaGonderimService)
    {
        _epostaGonderimService = epostaGonderimService;
    }

    [HttpGet]
    public async Task<IActionResult> EpostaAdresiGonderimler()
    {
        var response = await _epostaGonderimService.GetAll();
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
```

```
    [HttpGet("{id}")]
    public IActionResult EpostaAdresiGonderim(int id)
    {
        var response = _epostaGonderimService.GetById(id);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpPost]
    public async Task<IActionResult> Gonder([FromBody] EpostaGonderimModel model)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }
        var response = await _epostaGonderimService.Create(model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpPost]
    public IActionResult EpostaAdresiGonderimEkleList([FromBody] List<EpostaGonderimModel>
model)
    {
        var response = _epostaGonderimService.CreateList(model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpPut("{id}")]
    public IActionResult EpostaAdresiGonderimGuncelle(int id, [FromBody] EpostaGonderimModel
model)
    {
        var response = _epostaGonderimService.Update(id, model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
```

```
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> EpostaAdresiGonderimSil(int id)
    {
        await _epostaGonderimService.Delete(id);
        return new StatusCodeResult(StatusCodes.Status204NoContent);
    }
}
```

EpostagonderimDetayController:

```
 [Route("api/[controller]/[action]")]
 [ApiController]
 public class EpostagonderimDetayController : ControllerBase
 {
     private readonly IEpostaGonderimDetayService _epostagonderimDetayService;

     public EpostagonderimDetayController(IEpostaGonderimDetayService
epostagonderimDetayService)
     {
         _epostagonderimDetayService = epostagonderimDetayService;
     }

     [HttpGet]
     public async Task<IActionResult> EpostagonderimDetaylar()
     {
         var response = await _epostagonderimDetayService.GetAll();
         return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
     }

     [HttpGet]
```

```
    public async Task<IActionResult> EpostagonderimDetaylarWithIncludes()
    {
        var response = await _epostagonderimDetayService.GetAllWithIncludesAsync();
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpGet]
    public IActionResult EpostagonderimDetay(int id)
    {
        var response = _epostagonderimDetayService.GetById(id);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }

    [HttpGet]
    public async Task<IActionResult> EpostagonderimDetaySil(int id)
    {
        await _epostagonderimDetayService.Delete(id);
        return new StatusCodeResult(StatusCodes.Status204NoContent);
    }



}
```

KisiController:

```
[Route("api/[controller]/[action]")]
[ApiController]
public class KisiController : ControllerBase
{
    private readonly IKisiService _kisiService;
```

```
public KisiController(IKisiService kisiService)
{
    _kisiService = kisiService;
}

[HttpGet]
public async Task<IActionResult> Kisiler()
{
    var response = await _kisiService.GetAll();
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
}
[HttpGet]
public async Task<IActionResult> KisilerwithFilter(string? cinsiyet, int? yasMin, int? yasMax)
{
    var response = await _kisiService.GetAllByFilter(cinsiyet, yasMin, yasMax);
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
}

[HttpGet("{id}")]
public IActionResult Kisi(int id)
{
    var response = _kisiService.GetById(id);
    return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
}
[HttpPost]
public IActionResult KisiEkle([FromBody] KisiModel model)
{
```

```
        var response = _kisiService.Create(model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpPost]
    public IActionResult KisiEkleList([FromBody] List<KisiModel> model)
    {
        var response = _kisiService.CreateList(model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpPut("{id}")]
    public IActionResult KisiGuncelle(int id, [FromBody] KisiModel model)
    {
        var response = _kisiService.Update(id, model);
        return new ObjectResult(response.Data) { StatusCode = response.StatusCode };
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> KisiSil(int id)
    {
        await _kisiService.Delete(id);
        return new StatusCodeResult(StatusCodes.Status204NoContent);
    }
}
```

program.cs sayfamda debendecy injectionlardan yararlandım:

```
using Microsoft.EntityFrameworkCore;
using PiDataEmailApp.Business.Abstract;
using PiDataEmailApp.Business.Concrete;
using PiDataEmailApp.Business.Mapping;
using PiDataEmailApp.DataAccess.Abstract;
using PiDataEmailApp.DataAccess.Concrete;
```

```
using PiDataEmailApp.DataAccess.Concrete.Dal;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.



builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<PiDataDbContext>(options =>
{
    options.UseNpgsql(builder.Configuration.GetConnectionString("NpgsqlConnection"));
});
//auto mapper
builder.Services.AddAutoMapper(typeof(MapProfile));

//dependency injection
builder.Services.AddScoped<IKisiDal, KisiDal>();
builder.Services.AddScoped<IEpostaAdresiDal, EpostaAdresiDal>();
builder.Services.AddScoped<IEpostaGonderimDal, EpostaGonderimDal>();
builder.Services.AddScoped<IKisiService, KisiManager>();
builder.Services.AddScoped<IEpostaAdresiService, EpostaAdresiManager>();
builder.Services.AddScoped<IEpostaGonderimService, EpostaGonderimManager>();
builder.Services.AddScoped<IEpostaGonderimDetayDal, EpostaGonderimDetayDal>();
```

```
builder.Services.AddScoped<IEpostaGonderimDetayService, EpostaGonderimDetayManager>();



var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseCors(x => x
    .AllowAnyOrigin()
    .AllowAnyMethod()
    .AllowAnyHeader());



app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

React Uygulamam:

Projenin ön yüzünü javascript kütüphanelerinden olan react ile yazdım meterial ui kullandım ve bootstraplerden yararlandım.

Kisileri kaydetme sayfam:

```
import React, { useState } from 'react'
import {
 TextField,
 Button,
 Typography,
 Container,
 Grid,
 Snackbar,
 MenuItem,
 Select,
 InputLabel,
 FormControl,
} from '@mui/material'

function Kisikayit() {
 const [formData, setFormData] = useState({
  ad: '',
  soyad: '',
  telefon: '',
  eposta: '',
  yas: '',
  cinsiyet: '', // Cinsiyet alanı eklendi
  unvan: '',
  isyeri: '',
 })

 const [openSnackbar, setOpenSnackbar] = useState(false)

 const handleChange = (e) => {
  const { name, value } = e.target
  setFormData((prevState) => ({
   ...prevState,
   [name]: value,
  }))
 }

 const handleSubmit = (e) => {
  e.preventDefault()
  fetch('https://localhost:7012/api/Kisi/KisiEkle', {
```

```
    method: 'POST',
    headers: {
     'Content-Type': 'application/json',
    },
    body: JSON.stringify(formData),
   })
    .then((response) => response.json())
    .then((data) => {
     console.log('API response:', data)
     setOpenSnackbar(true)
     setFormData({
      ad: '',
      soyad: '',
      telefon: '',
      eposta: '',
      yas: '',
      cinsiyet: '',
      unvan: '',
      isyeri: '',
     })
    })
    .catch((error) => {
     console.error('API error:', error)
    })
  }

  const handleCloseSnackbar = () => {
   setOpenSnackbar(false)
  }

  return (
   <Container maxWidth="md">
    <Typography variant="h4" gutterBottom>
     2. Kişi Kayıtları
    </Typography>
    <Typography variant="body1" paragraph>
     Eposta gönderilecek kişilerin kaydedildiği bölümdür. İşletme, buraya
     kişileri kaydedecektir.
    </Typography>
    <form onSubmit={handleSubmit}>
     <Grid container spacing={2}>
      <Grid item xs={6}>
       <TextField
        label="Ad"
        name="ad"
```

```
      value={formData.ad}
      onChange={handleChange}
      fullWidth
      required
     />
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="Soyad"
      name="soyad"
      value={formData.soyad}
      onChange={handleChange}
      fullWidth
      required
     />
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="Telefon"
      name="telefon"
      value={formData.telefon}
      onChange={handleChange}
      fullWidth
      required
     />
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="E-posta"
      name="eposta"
      value={formData.eposta}
      onChange={handleChange}
      fullWidth
      required
     />
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="Yaş"
      name="yas"
      type="number"
      value={formData.yas}
      onChange={handleChange}
      fullWidth
      required
```

```
      />
    </Grid>
    <Grid item xs={6}>
     <FormControl fullWidth>
      <InputLabel id="cinsiyet-label">Cinsiyet</InputLabel>
      <Select
       labelId="cinsiyet-label"
       id="cinsiyet-select"
       value={formData.cinsiyet}
       onChange={handleChange}
       name="cinsiyet"
       required // Cinsiyet alanı gerekli olduğu belirtildi
      >
       <MenuItem value="Erkek">Erkek</MenuItem>
       <MenuItem value="Kadın">Kadın</MenuItem>
      </Select>
     </FormControl>
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="Unvan"
      name="unvan"
      value={formData.unvan}
      onChange={handleChange}
      fullWidth
     />
    </Grid>
    <Grid item xs={6}>
     <TextField
      label="İş Yeri"
      name="isyeri"
      value={formData.isyeri}
      onChange={handleChange}
      fullWidth
     />
    </Grid>
    <Grid item xs={12}>
     <Button type="submit" variant="contained" color="primary" fullWidth>
      Kaydet
     </Button>
    </Grid>
   </Grid>
  </form>
  <Snackbar
   anchorOrigin={{
```

```
    vertical: 'bottom',
    horizontal: 'left',
   }}
   open={openSnackbar}
   autoHideDuration={6000}
   onClose={handleCloseSnackbar}
   message="Kişi kaydedildi."
  />
 </Container>
)
}

export default Kisikayit
```

Eposta tanıtım sayfam:

```
import React, { useState } from 'react'
import {
 TextField,
 Button,
 Typography,
 Container,
 Grid,
 Snackbar,
} from '@mui/material'

export default function EpostaTanitim() {
 const [formData, setFormData] = useState({
  adres: '',
  mailSunucuAdresi: '',
  kullaniciAdi: '',
  sifre: '',
  port: 0,
 })

 const [openSnackbar, setOpenSnackbar] = useState(false)

 const handleChange = (e) => {
  const { name, value } = e.target
  setFormData((prevState) => ({
   ...prevState,
   [name]: value,
  }))
 }
```

```
const handleSubmit = (e) => {
  e.preventDefault()

  fetch('https://localhost:7012/api/EpostaAdresi/EpostaAdresiEkle', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(formData),
  })
    .then((response) => response.json())
    .then((data) => {
      console.log('API response:', data)
      setOpenSnackbar(true)
      setFormData({
        adres: '',
        mailSunucuAdresi: '',
        kullaniciAdi: '',
        sifre: '',
        port: 0,
      })
    })
    .catch((error) => {
      console.error('API error:', error)
    })
}

const handleCloseSnackbar = () => {
  setOpenSnackbar(false)
}

return (
  <Container maxWidth="sm">
    <Typography variant="h4" align="center" gutterBottom>
      Eposta Adres Tanımı
    </Typography>
    <Typography variant="body1" paragraph>
      İşletme IT yöneticisi, sahip oldukları Eposta adreslerini buraya
      kaydedecektir.
    </Typography>

    <form onSubmit={handleSubmit}>
      <Grid container spacing={2}>
        <Grid item xs={12}>
          <TextField
```

```
      fullWidth
      label="E-posta Adresi"
      name="adres"
      value={formData.adres}
      onChange={handleChange}
      required
     />
    </Grid>
    <Grid item xs={12}>
     <TextField
      fullWidth
      label="Mail Sunucu Adresi"
      name="mailSunucuAdresi"
      value={formData.mailSunucuAdresi}
      onChange={handleChange}
      required
     />
    </Grid>
    <Grid item xs={12}>
     <TextField
      fullWidth
      label="Kullanıcı Adı"
      name="kullaniciAdi"
      value={formData.kullaniciAdi}
      onChange={handleChange}
      required
     />
    </Grid>
    <Grid item xs={12}>
     <TextField
      fullWidth
      label="Şifre"
      name="sifre"
      type="password"
      value={formData.sifre}
      onChange={handleChange}
      required
     />
    </Grid>
    <Grid item xs={12}>
     <TextField
      fullWidth
      label="Port"
      name="port"
      type="number"
```

```
        value={formData.port}
        onChange={handleChange}
        required
      />
    </Grid>
    <Grid item xs={12}>
     <Button type="submit" variant="contained" color="primary">
      Kaydet
     </Button>
    </Grid>
   </Grid>
  </form>

  <Snackbar
   anchorOrigin={{
    vertical: 'bottom',
    horizontal: 'left',
   }}
   open={openSnackbar}
   autoHideDuration={6000}
   onClose={handleCloseSnackbar}
   message="Eposta tanıtıldı."
  />
 </Container>
)
}
```

Eposta Gonderim sayfam:

```
import React, { useState, useEffect } from 'react'
import axios from 'axios'
import {
 Table,
 TableBody,
 TableCell,
 TableContainer,
 TableHead,
 TableRow,
 TextField,
 Button,
 Grid,
 FormControl,
 InputLabel,
 Select,
 MenuItem,
```

```
  Snackbar,
} from '@mui/material'

function Epostagonderim() {
  const [kisiler, setKisiler] = useState([])
  const [filtre, setFiltre] = useState({
    cinsiyet: '',
    yasMin: '',
    yasMax: '',
  })
  const [gonderenEmail, setGonderenEmail] = useState('')
  const [epostaAdresleri, setEpostaAdresleri] = useState([])
  const [epostaIcerik, setEpostaIcerik] = useState('')
  const [epostaKonu, setEpostaKonu] = useState('')
  const [snackbarOpen, setSnackbarOpen] = useState(false)

  useEffect(() => {
    fetchKisiler()
    fetchEpostaAdresleri()
  }, [])

  const fetchKisiler = async () => {
    try {
      const response = await axios.get(
        'https://localhost:7012/api/Kisi/Kisiler',
      )
      setKisiler(response.data)
    } catch (error) {
      console.error('Error fetching kisiler:', error)
    }
  }

  const fetchEpostaAdresleri = async () => {
    try {
      const response = await axios.get(
        'https://localhost:7012/api/EpostaAdresi/EpostaAdresleri',
      )
      setEpostaAdresleri(response.data)
    } catch (error) {
      console.error('Error fetching eposta adresleri:', error)
    }
  }

  const handleFiltrele = async () => {
    try {
```

```
    const response = await axios.get(
      `https://localhost:7012/api/Kisi/KisilerwithFilter?cinsiyet=${filtre.cinsiyet}&yasMin=${filtre.yasMin}&yasMax=${filtre.yasMax}`,
    )
    setKisiler(response.data)
  } catch (error) {
    console.error('Error filtering kisiler:', error)
  }
}

const handleEpostaGonder = async () => {
  try {
    const epostaBilgileri = {
      konu: 'E-posta Konusu',
      icerik: epostaIcerik,
      gonderenEmail: gonderenEmail,
      kisiListesiIds: kisiler.map((kisi) => kisi.id),
    }
    console.log('E-posta bilgileri:', epostaBilgileri)
    const response = await axios.post(
      'https://localhost:7012/api/EpostaAdresiGonderim/Gonder',
      epostaBilgileri,
    )

    console.log('E-posta gönderme başarılı:', response.data)

    // Form alanlarını sıfırla
    setEpostaKonu('')
    setGonderenEmail('')
    setEpostaIcerik('')
    setKisiler([])

    // Snackbar'ı aç
    setSnackbarOpen(true)
  } catch (error) {
    console.error('Error sending e-posta:', error)
  }
}

const handleChange = (e) => {
  setFiltre({ ...filtre, [e.target.name]: e.target.value })
}

const handleGonderenEmailChange = (e) => {
  setGonderenEmail(e.target.value)
```

```
}

const handleIcerikChange = (e) => {
 setEpostaIcerik(e.target.value)
}

const handleKonuChange = (e) => {
 setEpostaKonu(e.target.value)
}

const handleCloseSnackbar = () => {
 setSnackbarOpen(false)
}

return (
 <div>
  <h1>E-posta Gönderimi</h1>
  <div>
   <h2>Kişiler</h2>
   <TableContainer>
    <Table>
     <TableHead>
      <TableRow>
       <TableCell>Ad</TableCell>
       <TableCell>Soyad</TableCell>
       <TableCell>Yaş</TableCell>
       <TableCell>Cinsiyet</TableCell>
      </TableRow>
     </TableHead>
     <TableBody>
      {kisiler.map((kisi) => (
       <TableRow key={kisi.id}>
        <TableCell>{kisi.ad}</TableCell>
        <TableCell>{kisi.soyad}</TableCell>
        <TableCell>{kisi.yas}</TableCell>
        <TableCell>{kisi.cinsiyet}</TableCell>
       </TableRow>
      ))}
     </TableBody>
    </Table>
   </TableContainer>
  </div>
  <div>
   <h2>Filtreleme</h2>
   <Grid container spacing={2}>
```

```
    <Grid item xs={4}>
     <TextField
      label="Cinsiyet"
      name="cinsiyet"
      value={filtre.cinsiyet}
      onChange={handleChange}
      fullWidth
     />
    </Grid>
    <Grid item xs={4}>
     <TextField
      label="Yaş Min"
      name="yasMin"
      type="number"
      value={filtre.yasMin}
      onChange={handleChange}
      fullWidth
     />
    </Grid>
    <Grid item xs={4}>
     <TextField
      label="Yaş Max"
      name="yasMax"
      type="number"
      value={filtre.yasMax}
      onChange={handleChange}
      fullWidth
     />
    </Grid>
    <Grid item xs={12}>
     <Button
      variant="contained"
      color="primary"
      onClick={handleFiltrele}
     >
      Filtrele
     </Button>
    </Grid>
   </Grid>
   <div>
    <h2>E-posta Gönderimi</h2>
    {/* E-posta gönderme formu */}
    <Grid container spacing={2}>
     <Grid item xs={6}>
      <TextField
```

```
        label="Konu"
        name="konu"
        value={epostaKonu}
        onChange={handleKonuChange}
        fullWidth
      />
    </Grid>
    <Grid item xs={6}>
     <FormControl fullWidth>
      <InputLabel id="gonderen-email-label">
       Gönderen E-posta Adresi
      </InputLabel>
      <Select
       labelId="gonderen-email-label"
       id="gonderen-email-select"
       value={gonderenEmail}
       onChange={handleGonderenEmailChange}
      >
       {epostaAdresleri.map((epostaAdres) => (
        <MenuItem key={epostaAdres.id} value={epostaAdres.adres}>
         {epostaAdres.adres}
        </MenuItem>
       ))}
      </Select>
     </FormControl>
    </Grid>
    <Grid item xs={12}>
     <TextField
      label="İçerik"
      name="icerik"
      value={epostaIcerik}
      onChange={handleIcerikChange}
      fullWidth
      multiline
      rows={4}
     />
    </Grid>
    <Grid item xs={12}>
     <Button
      variant="contained"
      color="primary"
      onClick={handleEpostaGonder}
     >
      E-posta Gönder
     </Button>
```

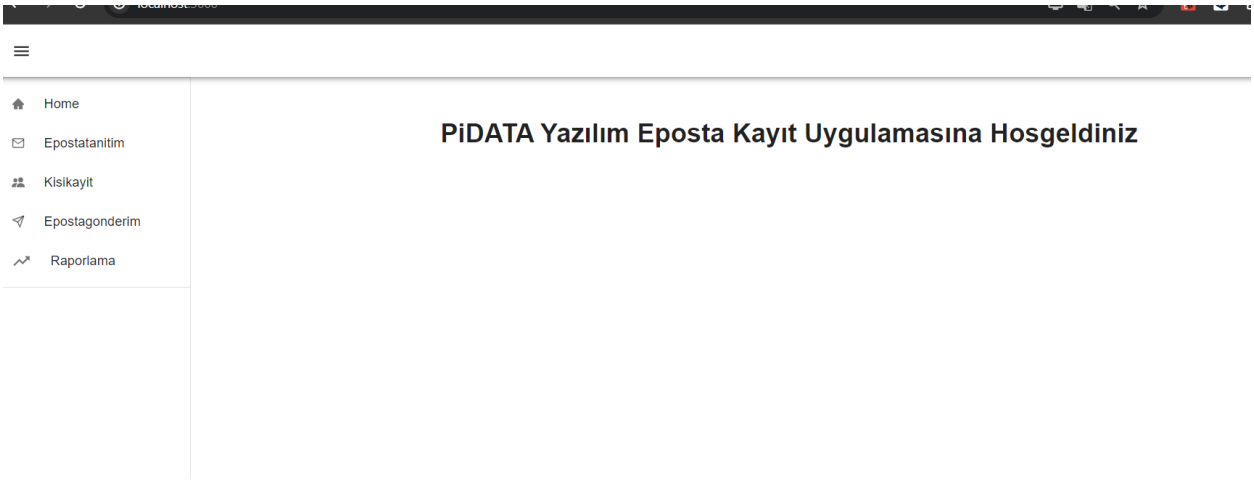
```
      </Grid>
    </Grid>
   </div>
  </div>
  <Snackbar
   open={snackbarOpen}
   autoHideDuration={6000}
   onClose={handleCloseSnackbar}
   message="E-posta başarıyla gönderildi."
  />
 </div>
)
}

export default Epostagonderim
```

Raporlama sayfam :

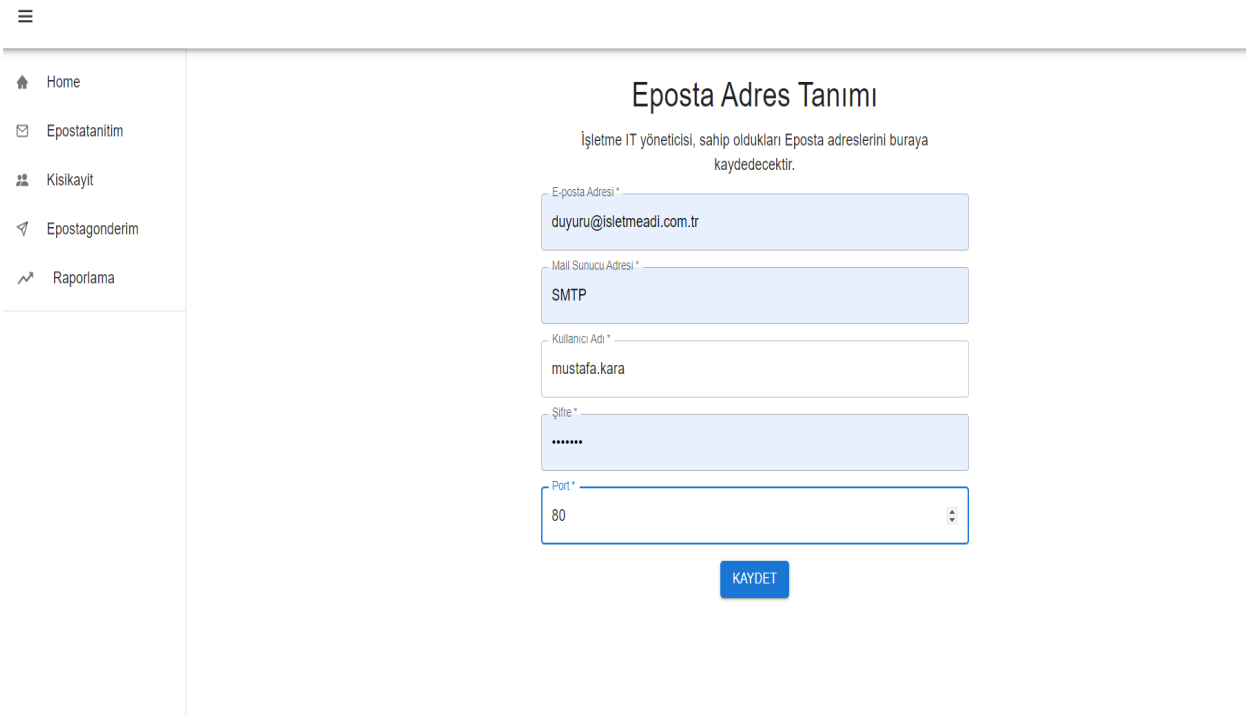
```
import React, { useState, useEffect } from 'react'
import axios from 'axios'
import {
 Table,
 TableBody,
 TableCell,
 TableContainer,
 TableHead,
 TableRow,
 Paper,
} from '@mui/material'

function Raporlama() {
 const [epostaDetaylar, setEpostaDetaylar] = useState([])

 useEffect(() => {
  fetchEpostaDetaylar()
 }, [])

 const fetchEpostaDetaylar = async () => {
  try {
   const response = await axios.get(
'https://localhost:7012/api/EpostagonderimDetay/EpostagonderimDetaylarWit
hIncludes',
   )
   setEpostaDetaylar(response.data)
```

```
  } catch (error) {
   console.error('Error fetching eposta detaylar:', error)
  }
 }

 return (
  <div>
   <h1>E-posta Gönderim Detayları</h1>
   <TableContainer component={Paper}>
    <Table>
     <TableHead>
      <TableRow>
       <TableCell>Tarih</TableCell>
       <TableCell>Gönderen E-posta Adresi</TableCell>
       <TableCell>Konu</TableCell>
       <TableCell>İçerik</TableCell>
       <TableCell>İletim Durumu</TableCell>
       <TableCell>Kişi</TableCell>
      </TableRow>
     </TableHead>
     <TableBody>
      {epostaDetaylar.map((detay) => (
       <TableRow key={detay.epostaGonderimId}>
        <TableCell>{detay.gonderimTarihi}</TableCell>
        <TableCell>
         {detay.epostaGonderim.gonderenEpostaAdresi}
        </TableCell>
        <TableCell>{detay.epostaGonderim.konusu}</TableCell>
        <TableCell>{detay.epostaGonderim.icerigi}</TableCell>
        <TableCell>
         {detay.gonderimDurumu ? 'Başarılı' : 'Başarısız'}
        </TableCell>
        <TableCell>
         {detay.kisi.ad} {detay.kisi.soyad} ({detay.kisi.eposta})
        </TableCell>
       </TableRow>
      ))}
     </TableBody>
    </Table>
   </TableContainer>
  </div>
 )
}

export default Raporlama
```

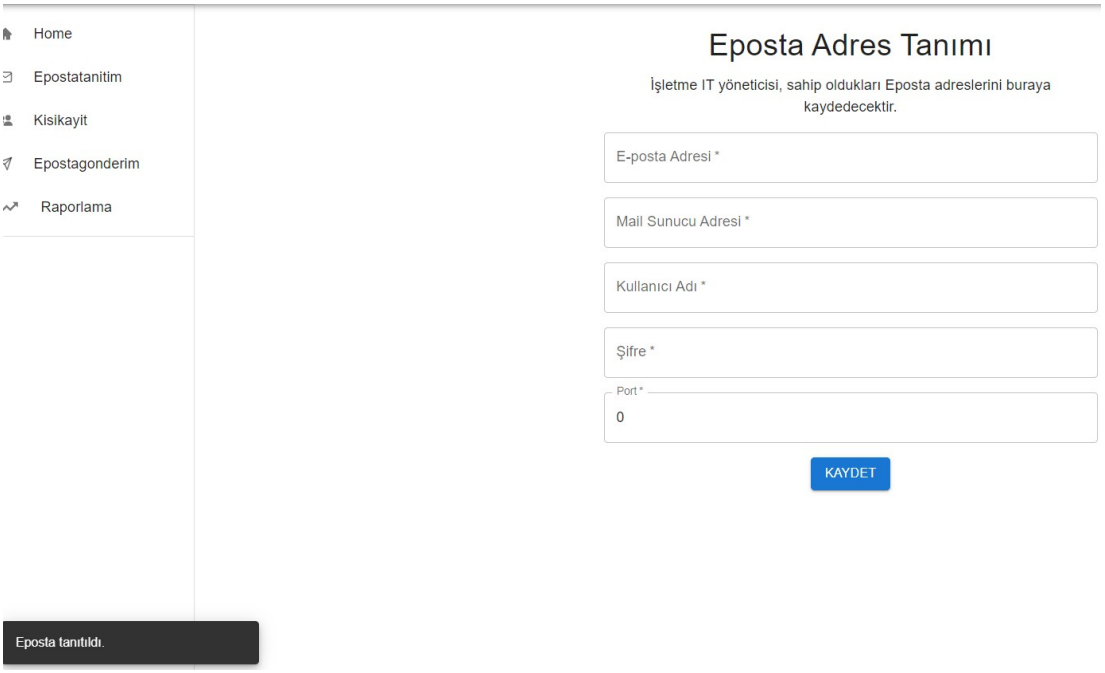
Projenin Ekran Görüntüleri



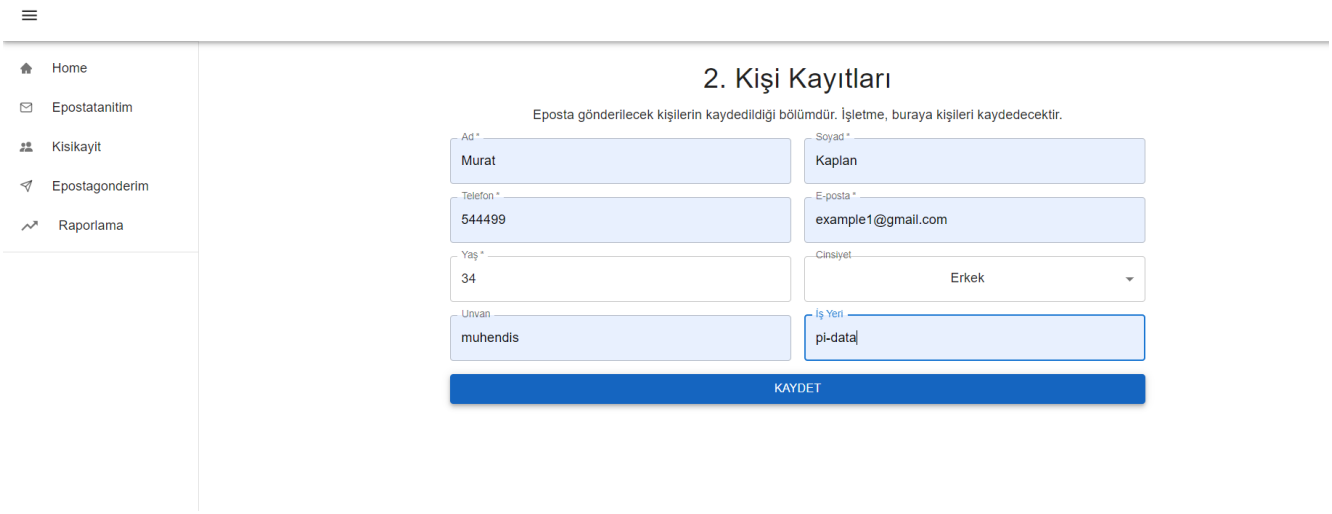
sayfanın ana görüntüsü bu şekilde



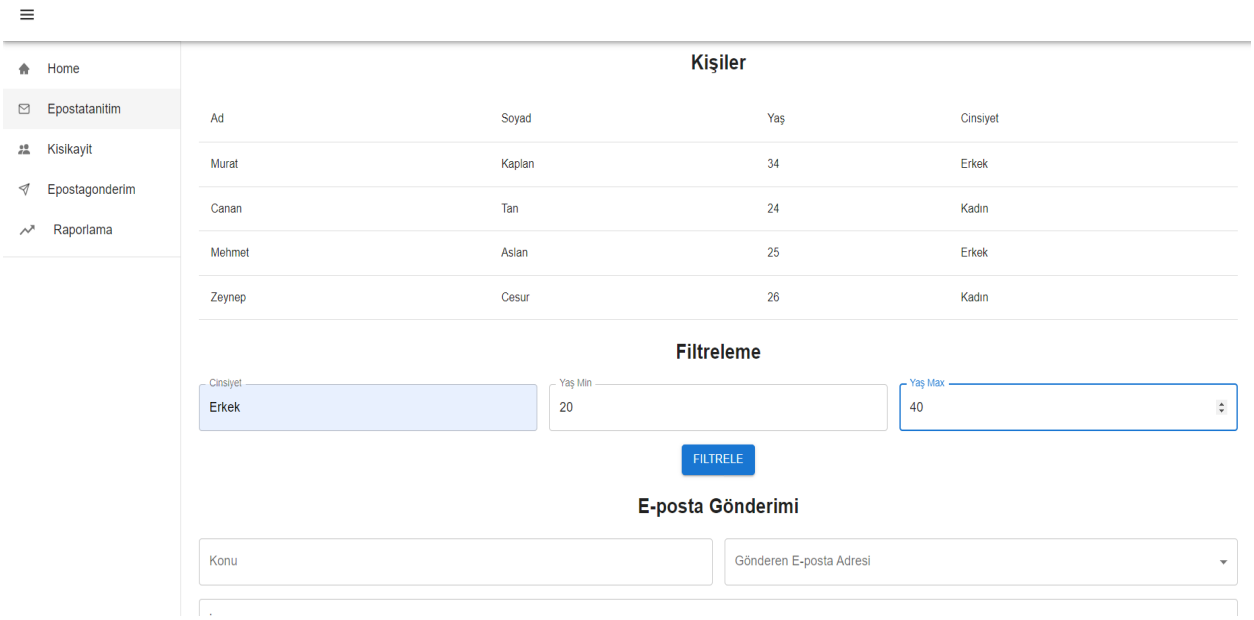
Form vasıtası ile Eposta tanıtım yapıyorum.



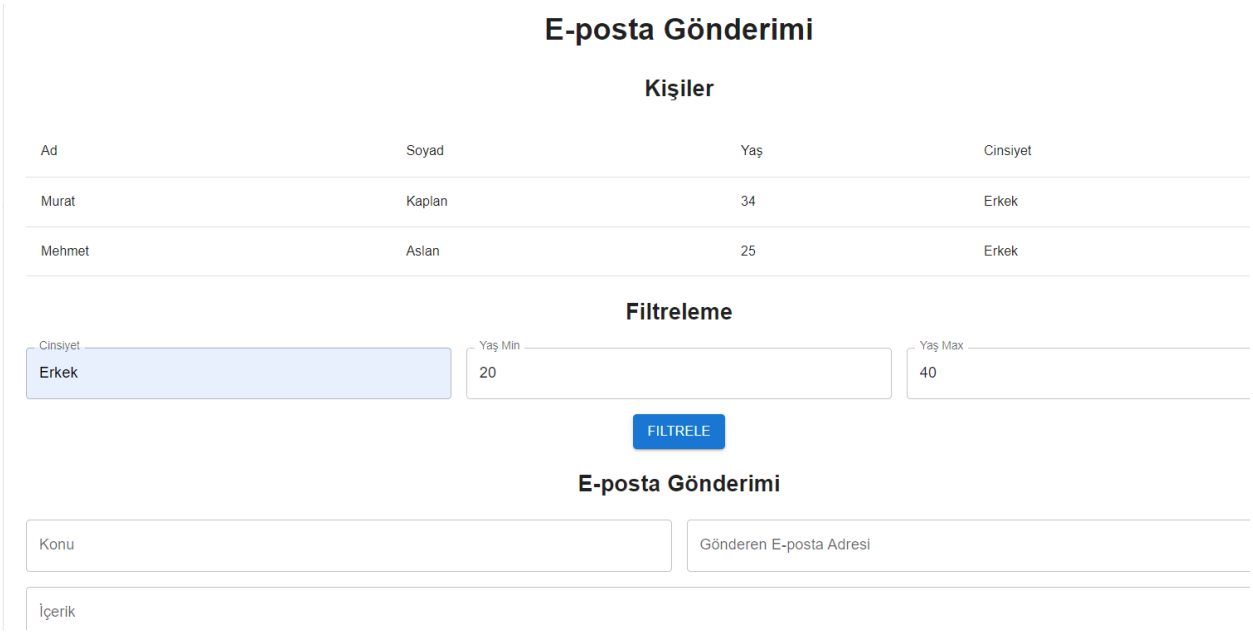
sayfa tanıtıldı snackbar sol alta gözükmektedir.



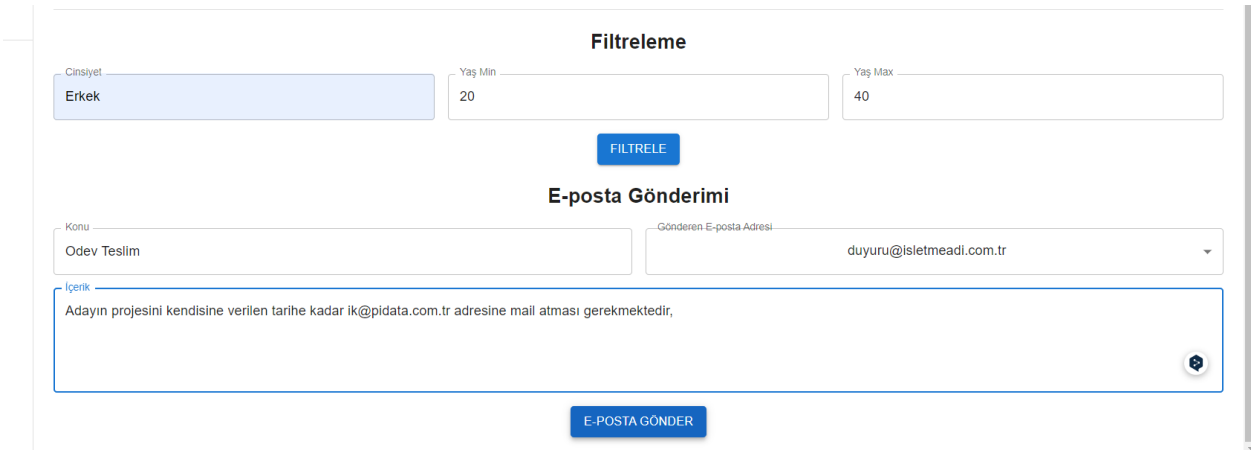
Kisi kayıtları sayfası ile kişileri kaydediyorum.



Eposta gonderim sayfasında ise önce filtreleme yapıyorum ve eposta gönderimini sağğlıyorum.



Filtreleme sonucu geldi.



Konu, Gonderen eposta alanı ve içerik girildikten sonra veri tabanına kayıt olacak ve bir servis ie kisiler ad soyad alanı eklenip eposta gönderilecek.

Raporlama sayfasında gönderilen epostalar takip edilebilecektir.