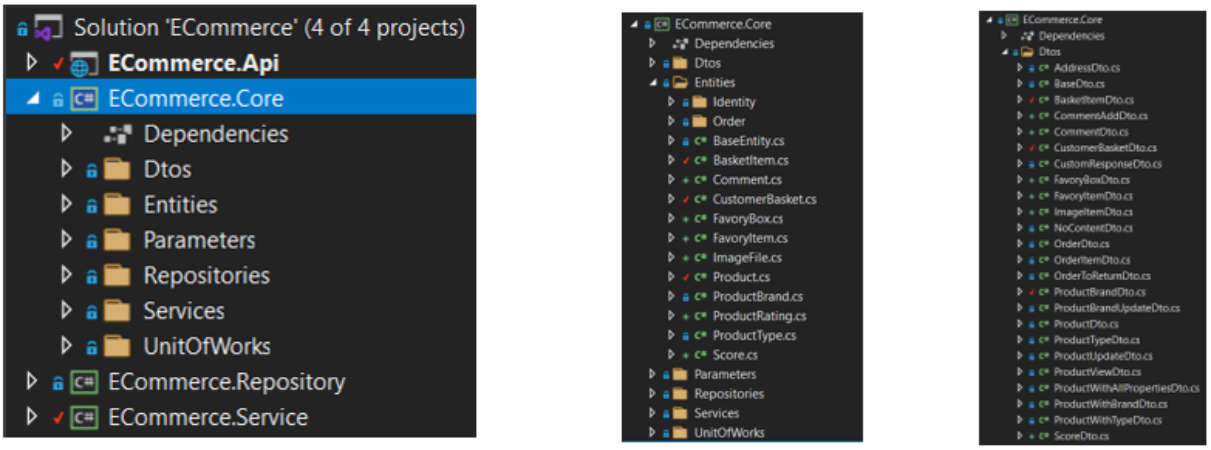
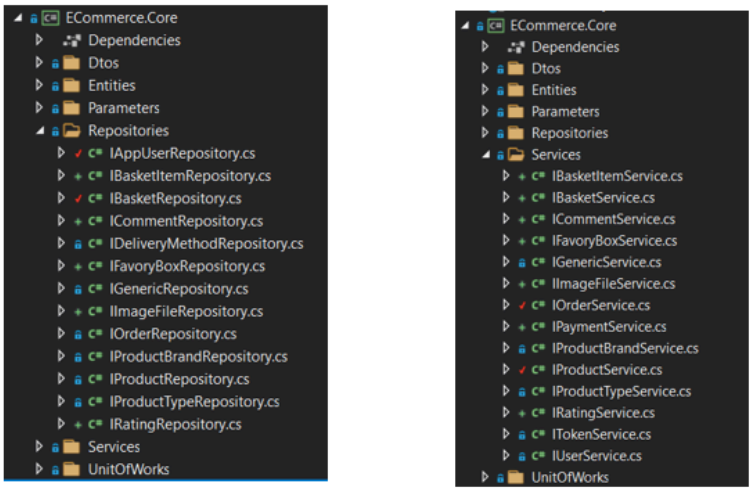
BACKEND .Net Core API ve FRONTEND Angular.js kullandım

- C#, .Net Core 3.1 APİ, Katmanlı mimari, MS Sql Server
- EntityFrameworkCore, Code First yaklaşımı
- Dto, Interface, IUnitOfWork ve Repository kullanımı
- Generic Response Dto nesnesi
- Email Service
- Mapping
- Fluent Validation
- ANGULAR.JS
- HTML, CSS, Bootstrapt
- LINQ
- Identity
- JWT Token
- Postman ve Swaqger kullanımı



Şekil 3.4. Ecommerce.Core katmanı görünüşü

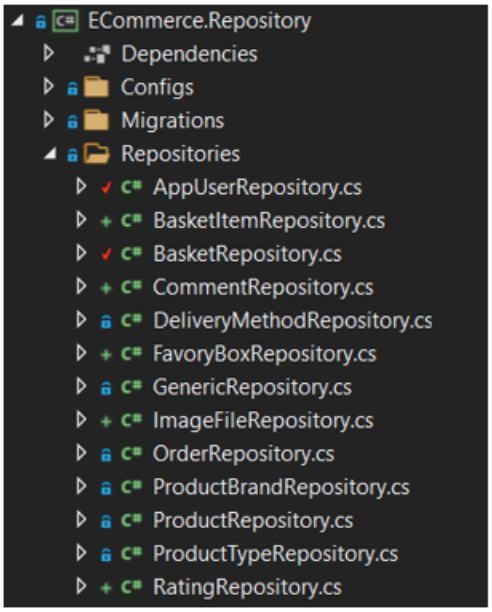


Şekil 3.5. Ecommerce.Core layer repository interfaces

```
public class ECommerceContext : IdentityDbContext<AppUser>
{
    public DbSet<AppUser>  AppUsers { get; set; }
    1 reference
    public DbSet<Core.Entities.Identity.Address>  Addresses { get; set; }
    9 references
    public DbSet<Product> Products { set; get; }
    0 references
    public DbSet<ProductType> ProductTypes { set; get; }
    0 references
    public DbSet<ProductBrand> ProductBrands { set; get; }
    public DbSet<Order> Orders { set; get; }
    0 references
    public DbSet<OrderItem> OrderItems { set; get; }
    0 references
    public DbSet<DeliveryMethod> DeliveryMethods { set; get; }

    public DbSet<Comment> Comments { get; set; }
    0 references
    public DbSet<ImageFile>  ImageFiles { get; set; }
    public DbSet<CustomerBasket> CustomerBaskets { get; set; }
    1 reference
    public DbSet<BasketItem> BasketItem { get; set; }
    public DbSet<FavoryBox>  FavoryBoxes { get; set; }
    public DbSet<ProductRating> ProductRatings { get; set; }
    0 references
    public DbSet<Score> Scores { get; set; }
```

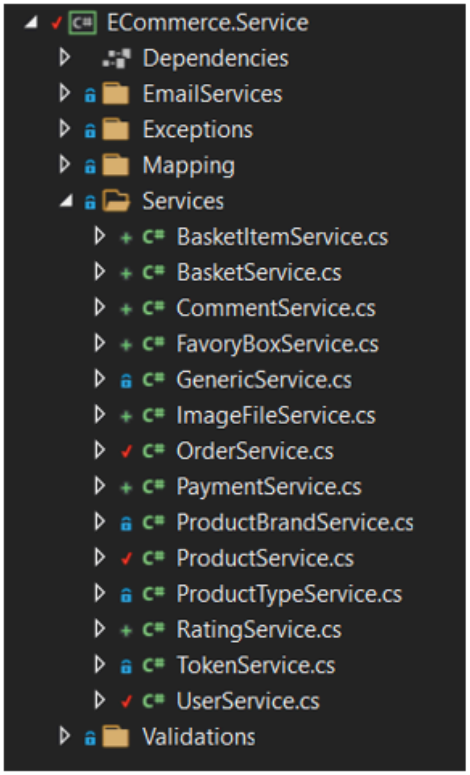
Şekil 3.6. ECommerceContext sınıfı



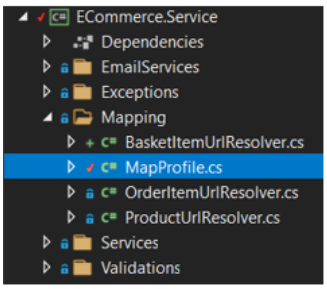
Şekil 3.8. ECommerce.Repository katmanı görünümü

```
public interface IGenericRepository<T> where T : class
{
    5 references
    IQueryable<T> GetList(Expression<Func<T, bool>> filter = null);
    1 reference
    Task<T> GetAsync(Expression<Func<T, bool>> filter);
    3 references
    Task AddAsync(T entity);
    2 references
    void Delete(T entity);
    2 references
    void Update(T entity);
    7 references
    Task<T> GetByIdAsync(int id);
    2 references
    Task<bool> AnyAsync(Expression<Func<T,bool>> expression);
    2 references
    IQueryable<T> Where(Expression<Func<T,bool>> expression);
    2 references
    void RemoveRange(IEnumerable<T> entities);
    2 references
    Task AddRangeAsync(IEnumerable<T> entities);
```

Şekil 3.9. IGenericRepository interface



Şekil 3.10. ECommerce.Service katmanı görünümü




```cs
public class MapProfile  : Profile
{

        0 references
        public MapProfile()   {

        var apiUrl = "https://localhost:44347/";

        CreateMap<Product, ProductDto>().ReverseMap();
        CreateMap<ProductUpdateDto, Product>();
        CreateMap<Product,ProductWithBrandDto>();
        CreateMap<Product,ProductWithTypeDto>();
        CreateMap<Product,ProductWithAllPropertiesDto>();

        CreateMap<ProductBrand, ProductBrandDto>().ReverseMap();
        CreateMap<ProductBrandUpdateDto, ProductBrand>();

        CreateMap<ProductType, ProductTypeDto>().ReverseMap();

        CreateMap<Product, ProductViewDto>()
            .ForMember(des => des.ProductBrand, operation => operation.MapFrom(source => source.ProductBrand.Name))
            .ForMember(des => des.ProductType, operation => operation.MapFrom(source => source.ProductType.Name))
            .ForMember(des => des.PictureUrl, operation => operation.ResolveUsing<ProductUrlResolver>());
```

Şekil 3.11. MapProfile class

```cs
1 reference
public class RepoServiceModule : Module
{

    0 references
    protected override void Load(ContainerBuilder builder)
    {

        builder.RegisterGeneric(typeof(GenericRepository<>)).As(typeof(IGenericRepository<>)).InstancePerLifetimeScope();
        builder.RegisterGeneric(typeof(GenericService<>)).As(typeof(IGenericService<>)).InstancePerLifetimeScope();

        builder.RegisterType<UnitOfWork>().As<IUnitOfWork>();



        var apiAssembly = Assembly.GetExecutingAssembly();
        var repoAssembly = Assembly.GetAssembly(typeof(ECommerceContext));    // o katmandaki herhangi bir dosya ismi verebilirsin o anlar hangi katman olduğunu
        var serviceAssembly = Assembly.GetAssembly(typeof(MapProfile));

        builder.RegisterAssemblyTypes(apiAssembly, repoAssembly, serviceAssembly).Where(x => x.Name.EndsWith("Repository")).AsImplementedInterfaces().InstancePerLifetimeScope();


        builder.RegisterAssemblyTypes(apiAssembly, repoAssembly, serviceAssembly).Where(x => x.Name.EndsWith("Service")).AsImplementedInterfaces().InstancePerLifetimeScope();
```

Şekil 3.12. RepoServiceModule class

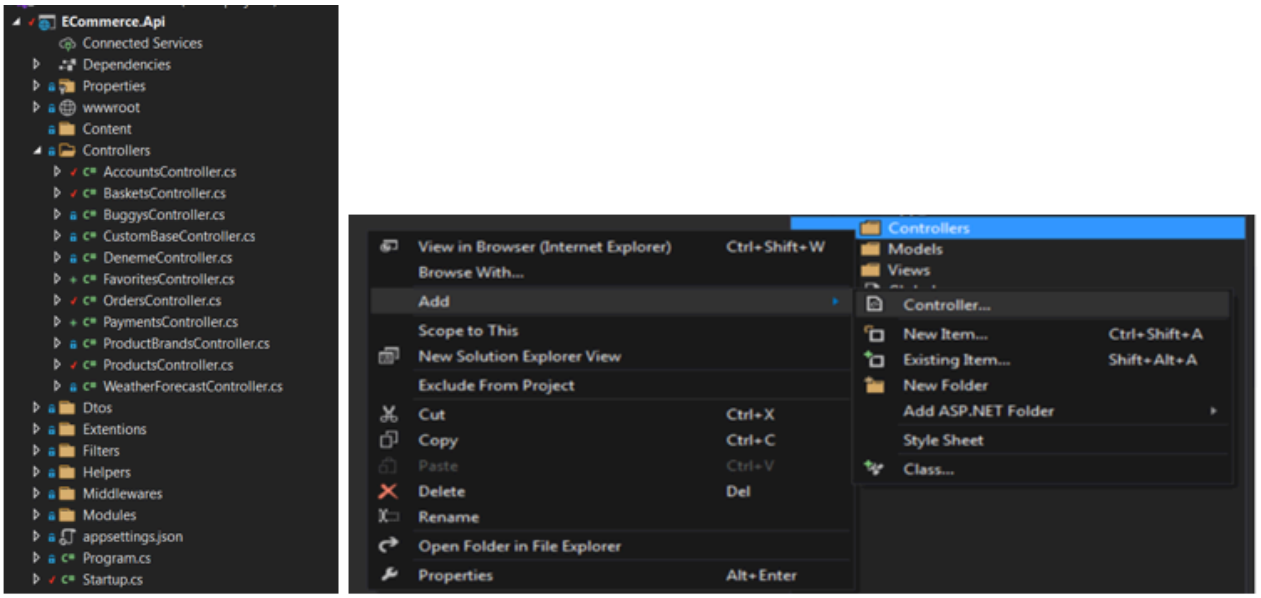
```cs
1 reference
public class ProductService : GenericService<Product>, IProductService
{
    private readonly IProductRepository _productRepository;
    private readonly IMapper _mapper;

    0 references
    public ProductService(IGenericRepository<Product> repository, IUnitOfWork unitOfWork, IMapper mapper, IProductRepository productRepository)
    {
        _mapper = mapper;
        _productRepository = productRepository;
    }
```

Şekil 3.13. Interface kullanarak dependency injection yapma örneği

```
public class ProductDtoValidator : AbstractValidator<ProductDto>
{
    0 references
    public ProductDtoValidator()
    {
        RuleFor(x => x.Price).InclusiveBetween(1, int.MaxValue).WithMessage("{PropertyName} 0'dan büyük olmalı!").NotEmpty().WithMessage("{PropertyName} 0'dan büyü
        RuleFor(x => x.Stock).InclusiveBetween(1, int.MaxValue).WithMessage("{PropertyName} 0'dan büyük olmalı!");
        RuleFor(x => x.Name).NotNull().WithMessage("{PropertyName} zorunlu alan boş bırakılamaz!").NotEmpty().WithMessage("{PropertyName} zorunlu alan boş bırakıla
        RuleFor(x=>x.ProductBrandId).InclusiveBetween(1, int.MaxValue).WithMessage("{PropertyName} 0'dan büyük olmalı!");
        RuleFor(x=>x.ProductTypeId).InclusiveBetween(1, int.MaxValue).WithMessage("{PropertyName} 0'dan büyük olmalı!");
        RuleFor(x => x.Description).NotNull().WithMessage("{PropertyName} zorunlu alan boş bırakılamaz!").NotEmpty().WithMessage("{PropertyName} zorunlu alan boş b
    }
}
```

Şekil 3.14. ProductDto için model doğrulama kuralları



Şekil 3.15. Ecommerce.Api katmanında controller oluşturma

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class ProductsController : CustomBaseController
{
    private readonly IMapper _mapper;
    private readonly IProductService _service;
    private readonly IProductBrandService _productBrandservice;
    private readonly IProductTypeService _productTypeservice;
    private readonly ICommentService _commentService;
    private readonly IImageFileService _imageFileService;
    private readonly IRatingService _ratingService;

    0 references
    public ProductsController(IProductService service, IMapper mapper, IProductBrandService productBrandservice, IProductTypeService productTypeservice,
    {

        _mapper = mapper;
        _service = service;
        _productBrandservice = productBrandservice;
        _productTypeservice = productTypeservice;
        _commentService = commentService;
        _imageFileService = imageFileService;
        _ratingService = ratingService;
    }
```

```
/// GET api/products
[HttpGet]
0 references
public async Task<IActionResult> List()
{
    var products = await _service.GetListAsync();
    var productsDtos = _mapper.Map<List<ProductDto>>(products.ToList());
    return CreateActionResult(CustomResponseDto<List<ProductDto>>.SuccessfulResponse(200, productsDtos));
}

/// GET api/products/2
[HttpGet("{id}")]
0 references
public async Task<IActionResult> GetById(int id)
{
    var product =  await _service.GetByIdAsync(id);
    var productDto = _mapper.Map<ProductDto>(product);
    return CreateActionResult(CustomResponseDto<ProductDto>.SuccessfulResponse(200,productDto));
}

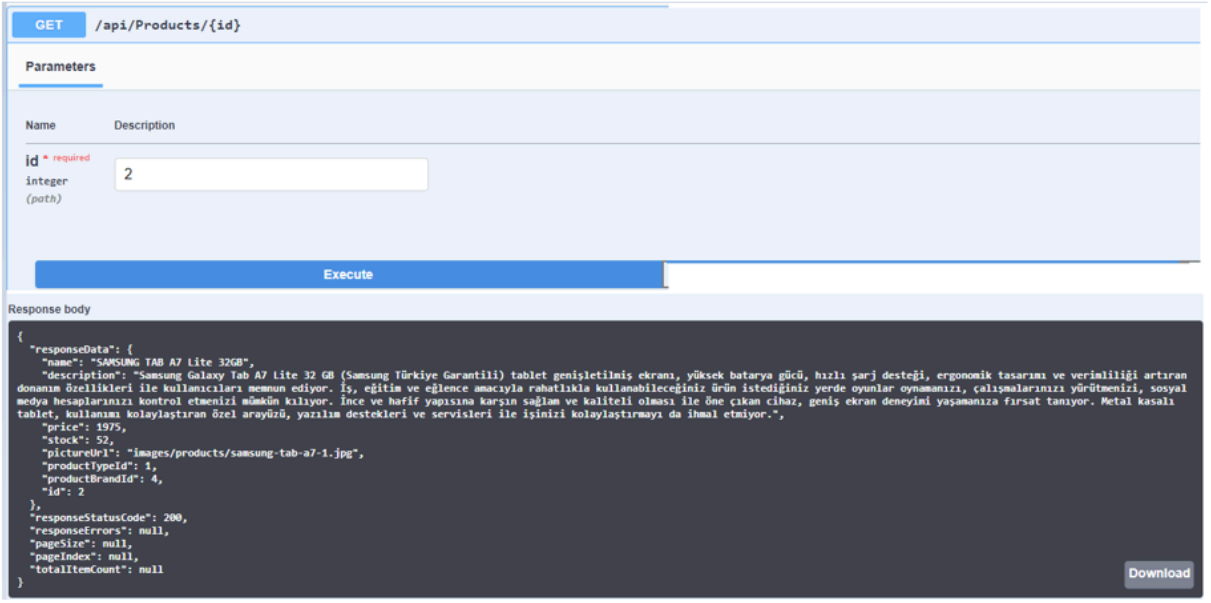
[HttpPost]
0 references
public async Task<IActionResult> Save(ProductDto productDto)
{
    var product =  await _service.AddAsync(_mapper.Map<Product>(productDto));
    var dto = _mapper.Map<ProductDto>(product);
    return CreateActionResult(CustomResponseDto<ProductDto>.SuccessfulResponse(201, dto));

}
```

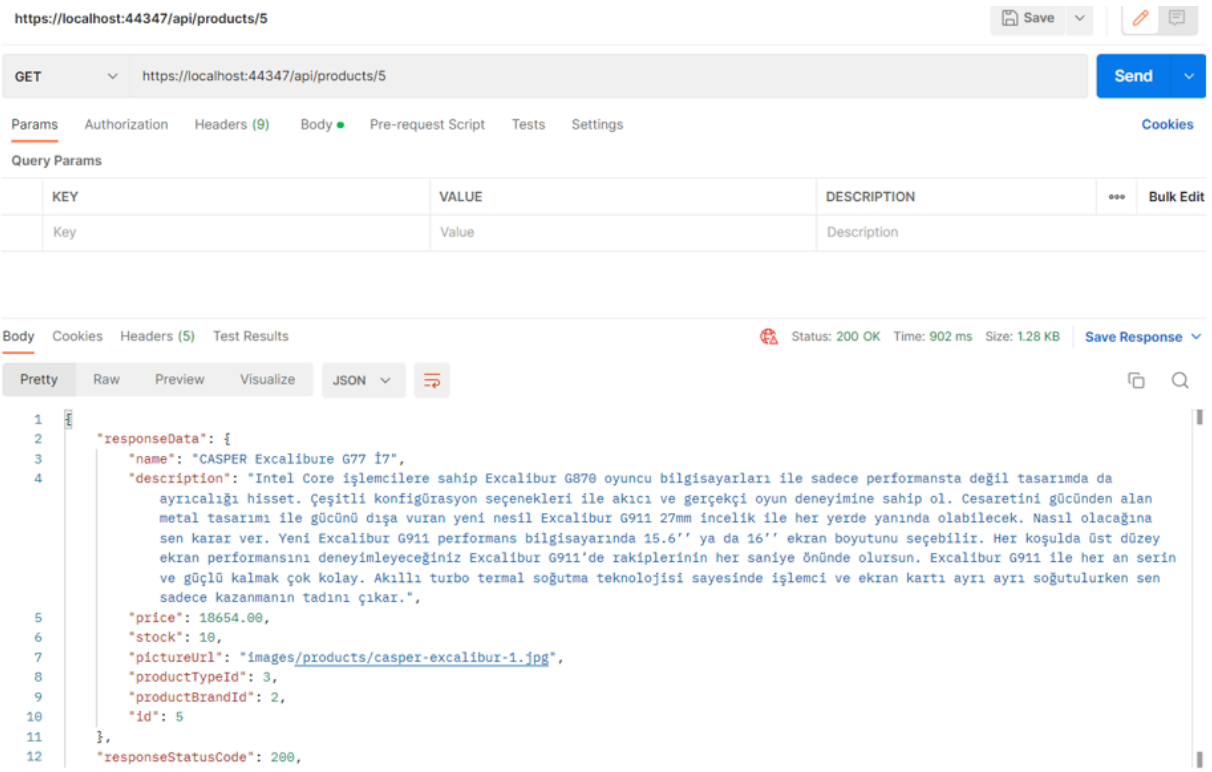
Şekil 3.17. ProductsController görünümü

```
[HttpGet("cheap-products")]
0 references
public async Task<IActionResult> CheapProducts()
{
    return CreateActionResult(await _service.GetCheapProducts());
}
```

Şekil 3.18. En ucuz ürünleri getiren action



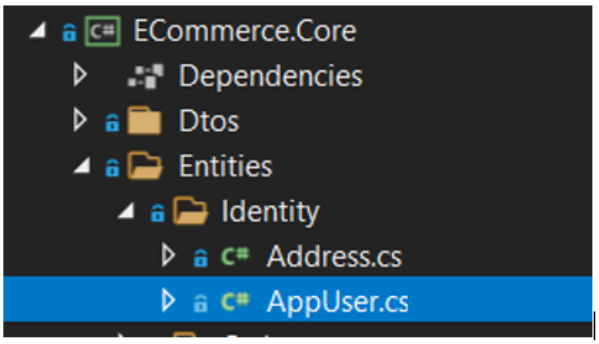
Şekil 3.19. Swagger kullanımı



Şekil 3.20. Postman kullanımı

3.4. Üyelik İşlemleri

ASP.NET Core Identity, ASP.NET Core uygulamalarında kullanıcı hesaplarını yönetmek ve depolamak için bir framework sağlar.

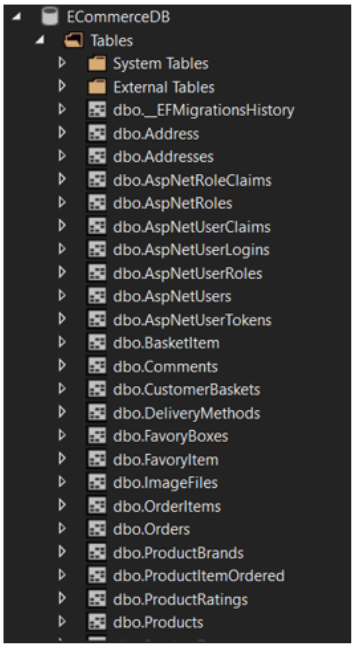


Şekil 3.21. AppUser sınıfı oluşturulması

ECommerce.Core katmanında AppUser class'ı oluşturdum ve IdentityUser interface'inden kalıtım aldım.

```
public class AppUser  : IdentityUser
{
    6 references
    public string DisplayName { get; set; }
    4 references
    public Address Address { get; set; }
}
```

Şekil 3.22. AppUser sınıfı görünümü



Şekil 3.24. ECommerceDB veritabanı tablolar

```
public class UserService : GenericService<AppUser>, IUserService
{

    private readonly IAppUserRepository _userRepository;
    private readonly IMapper _mapper;
    private readonly UserManager<AppUser> _userManager;

    0 references
    public UserService(IGenericRepository<AppUser> repository, IUnitOfWork unitOfWork, IMapper mapper, IAppUserRepository userRepository, UserManager<AppUser> userManager)
    {
        _mapper = mapper;
        _userRepository = userRepository;
        _userManager = userManager;
    }

    2 references
    public AppUser GetUserWithAddress(AppUser appUser)
    {
        var user =  _userRepository.GetUserWithAddress(appUser);
        return user;
    }


    2 references
    public AppUser UpdateUser(AppUser user, AddressDto dto)
    {

        var newAddress = _mapper.Map<AddressDto, Address>(dto);
        var address =  _userRepository.UpdateUserAddress(user,newAddress);
        return user;
    }
```

Şekil 3.25. UserService görünümü

```
public class TokenService : ITokenService
{

    private readonly IConfiguration _config;  //  config deki bilgileri okumak için
    private readonly SymmetricSecurityKey _key;   // şifreleme kütüphanesi

    0 references
    public TokenService(IConfiguration config)
    {
        _config = config;
        _key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Token:Key"]));
    }

    4 references
    public string CreateToken(AppUser user)
    {
        var claim = new List<Claim>  // Token oluşturuken kullanıcıya yetkiler vermek için
        {
            new Claim(JwtRegisteredClaimNames.Email,user.Email),   // email bilgisiyle eşle
            new Claim(JwtRegisteredClaimNames.GivenName,user.DisplayName),   //

        };

        var cred = new SigningCredentials(_key, SecurityAlgorithms.HmacSha512Signature);  // key ve şifreleme algoritmamı seçtim
        var tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(claim),
            Expires = DateTime.Now.AddDays(7),
            SigningCredentials = cred,
            Issuer = _config["Token:Issuer"]
        };  // Tanımlamamı yaptım artık oluşturucam
        var tokenHandler = new JwtSecurityTokenHandler();
        var token =  tokenHandler.CreateToken(tokenDescriptor);

        return tokenHandler.WriteToken(token);

    }
}
```

Şekil 3.27. Token servis görünümü

```
[HttpPost("register")]
0 references
public async Task<ActionResult<UserResponseDto>> Register(RegisterDto registerDto)
{

    if (CheckEmailExistAsync(registerDto.Email).Result.Value)
    {
        return BadRequest( CustomResponseDto<NoContentDto>.FailResponse(400, "Bu email adresi zaten kayıtlı!"));
    }

    var user = new AppUser
    {
        DisplayName = registerDto.DisplayName,
        Email = registerDto.Email,
        UserName = registerDto.UserName

    };

    var result = await _userManager.CreateAsync(user, registerDto.Password);
    if (!result.Succeeded)
    {
        return BadRequest(CustomResponseDto<NoContentDto>.FailResponse(401, "HATA! Kullanıcı kayıt etme işlemi gerçeklestirilemedi!"));
    }
```

devamı

```
if (result.Succeeded)
{
    // generate token   burda link oluşturduk
    var token = await _userManager.GenerateEmailConfirmationTokenAsync(user);
    var uriBuilder = new UriBuilder(_config["ReturnPaths:ConfirmEmail"]);
    var query = HttpUtility.ParseQueryString(uriBuilder.Query);
    query["userId"] = user.Id;
    query["token"] = token;
    uriBuilder.Query = query.ToString();
    var urlString = uriBuilder.ToString();
    var msg = "Lütfen hesabınızı onaylamak için aşağıdaki linke tıklayınız: \n\n" + urlString;
    var senderEmail = _config["ReturnPaths:SenderEmail"];
    await _emailSender.SendEmailAsync(senderEmail, user.Email, "Confirm your email address", msg);


}

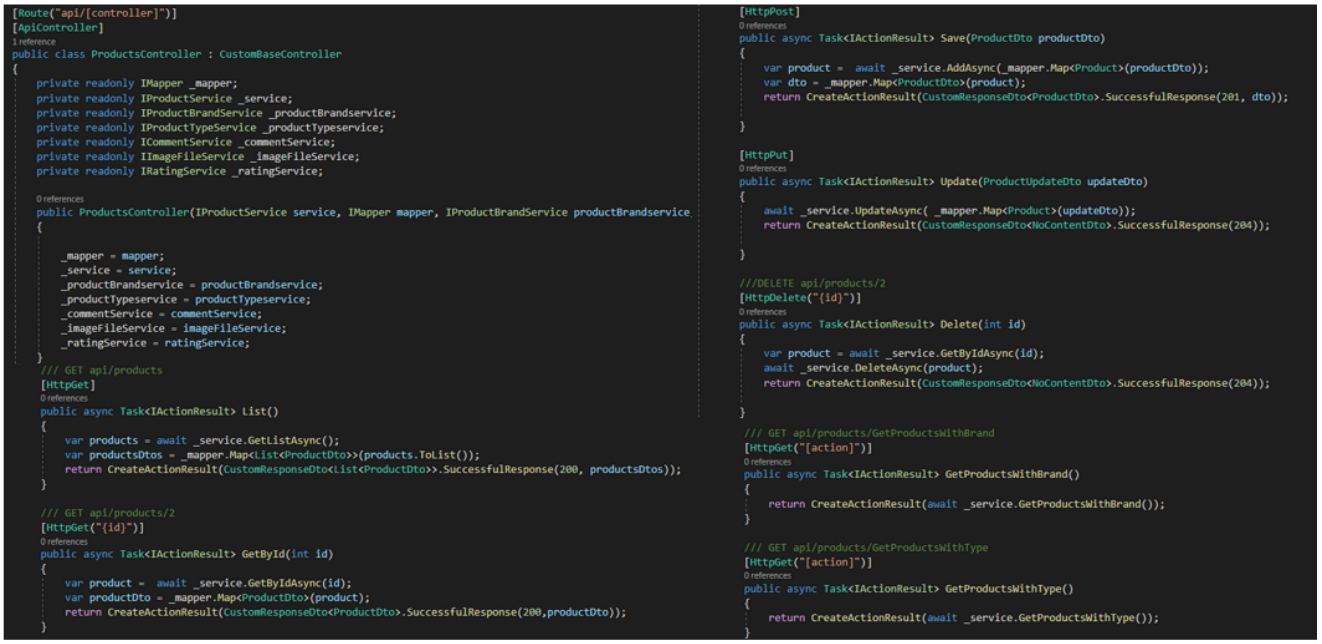
var favoryBox = new FavoryBox();
favoryBox.UserId = user.Id;
var favoryBoxDto = _mapper.Map<FavoryBoxDto>(favoryBox);

_favoryBoxService.UpdateFavoryBox(favoryBoxDto);


var basket = new CustomerBasket();
basket.UserId = user.Id;
await _basketService.CreateOrUpdateBasketAsync(basket);

return Ok(new UserResponseDto
{
    DisplayName = user.DisplayName,
    Email = user.Email,
    Token = _tokenService.CreateToken(user)
});
```

Şekil 3.28. Register(kullanıcı kayıt) işlemi



```
[Route("api/[controller]")]
[ApiController]
1 reference
public class ProductsController : CustomBaseController
{
    private readonly IMapper _mapper;
    private readonly IProductService _service;
    private readonly IProductBrandService _productBrandservice;
    private readonly IProductTypeService _productTypeservice;
    private readonly ICommentService _commentService;
    private readonly IImageFileService _imageFileService;
    private readonly IRatingService _ratingService;

    0 references
    public ProductsController(IProductService service, IMapper mapper, IProductBrandService productBrandservice
    {

        _mapper = mapper;
        _service = service;
        _productBrandservice = productBrandservice;
        _productTypeservice = productTypeservice;
        _commentService = commentService;
        _imageFileService = imageFileService;
        _ratingService = ratingService;
    }
    /// GET api/products
    [HttpGet]
    0 references
    public async Task<IActionResult> List()
    {
        var products = await _service.GetListAsync();
        var productsDtos = _mapper.Map<List<ProductDto>>(products.ToList());
        return CreateActionResult(CustomResponseDto<List<ProductDto>>.SuccessfulResponse(200, productsDtos));
    }

    /// GET api/products/2
    [HttpGet("{id}")]
    0 references
    public async Task<IActionResult> GetById(int id)
    {
        var product =  await _service.GetByIdAsync(id);
        var productDto = _mapper.Map<ProductDto>(product);
        return CreateActionResult(CustomResponseDto<ProductDto>.SuccessfulResponse(200,productDto));
    }
```

```
    [HttpPost]
    0 references
    public async Task<IActionResult> Save(ProductDto productDto)
    {
        var product =  await _service.AddAsync(_mapper.Map<Product>(productDto));
        var dto = _mapper.Map<ProductDto>(product);
        return CreateActionResult(CustomResponseDto<ProductDto>.SuccessfulResponse(201, dto));

    }

    [HttpPut]
    0 references
    public async Task<IActionResult> Update(ProductUpdateDto updateDto)
    {
        await _service.UpdateAsync( _mapper.Map<Product>(updateDto));
        return CreateActionResult(CustomResponseDto<NoContentDto>.SuccessfulResponse(204));

    }

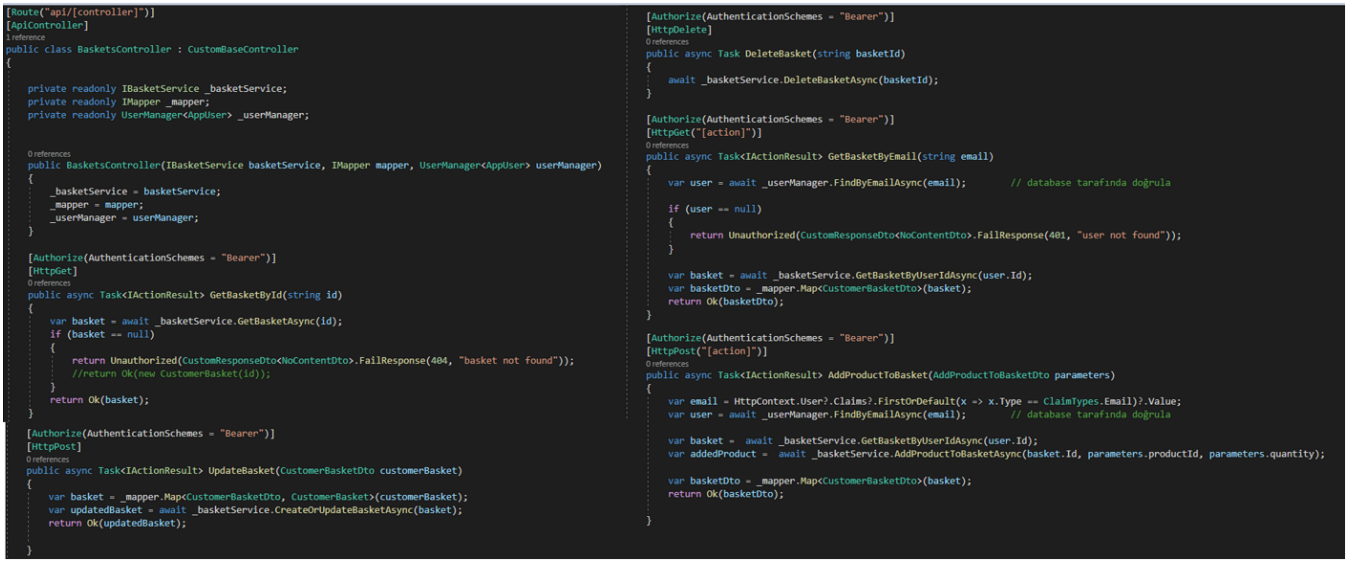
    ///DELETE api/products/2
    [HttpDelete("{id}")]
    0 references
    public async Task<IActionResult> Delete(int id)
    {
        var product = await _service.GetByIdAsync(id);
        await _service.DeleteAsync(product);
        return CreateActionResult(CustomResponseDto<NoContentDto>.SuccessfulResponse(204));

    }

    /// GET api/products/GetProductsWithBrand
    [HttpGet("[action]")]
    0 references
    public async Task<IActionResult> GetProductsWithBrand()
    {
        return CreateActionResult(await _service.GetProductsWithBrand());
    }

    /// GET api/products/GetProductsWithType
    [HttpGet("[action]")]
    0 references
    public async Task<IActionResult> GetProductsWithType()
    {
        return CreateActionResult(await _service.GetProductsWithType());
    }
```

Şekil 3.30. ProductsController görünümü



```
[Route("api/[controller]")]
[ApiController]
1 reference
public class BasketsController : CustomBaseController
{

    private readonly IBasketService _basketService;
    private readonly IMapper _mapper;
    private readonly UserManager<AppUser> _userManager;


    0 references
    public BasketsController(IBasketService basketService, IMapper mapper, UserManager<AppUser> userManager)
    {
        _basketService = basketService;
        _mapper = mapper;
        _userManager = userManager;
    }

    [Authorize(AuthenticationSchemes = "Bearer")]
    [HttpGet]
    0 references
    public async Task<IActionResult> GetBasketById(string id)
    {
        var basket = await _basketService.GetBasketAsync(id);
        if (basket == null)
        {
            return Unauthorized(CustomResponseDto<NoContentDto>.FailResponse(404, "basket not found"));
            //return Ok(new CustomerBasket(id));
        }
        return Ok(basket);
    }

    [Authorize(AuthenticationSchemes = "Bearer")]
    [HttpPost]
    0 references
    public async Task<IActionResult> UpdateBasket(CustomerBasketDto customerBasket)
    {
        var basket = _mapper.Map<CustomerBasketDto, CustomerBasket>(customerBasket);
        var updatedBasket = await _basketService.CreateOrUpdateBasketAsync(basket);
        return Ok(updatedBasket);

    }
```

```
    [Authorize(AuthenticationSchemes = "Bearer")]
    [HttpDelete]
    0 references
    public async Task DeleteBasket(string basketId)
    {
        await _basketService.DeleteBasketAsync(basketId);
    }

    [Authorize(AuthenticationSchemes = "Bearer")]
    [HttpGet("[action]")]
    0 references
    public async Task<IActionResult> GetBasketByEmail(string email)
    {
        var user = await _userManager.FindByEmailAsync(email);        // database tarafında doğrula

        if (user == null)
        {
            return Unauthorized(CustomResponseDto<NoContentDto>.FailResponse(401, "user not found"));
        }

        var basket = await _basketService.GetBasketByUserIdAsync(user.Id);
        var basketDto = _mapper.Map<CustomerBasketDto>(basket);
        return Ok(basketDto);
    }

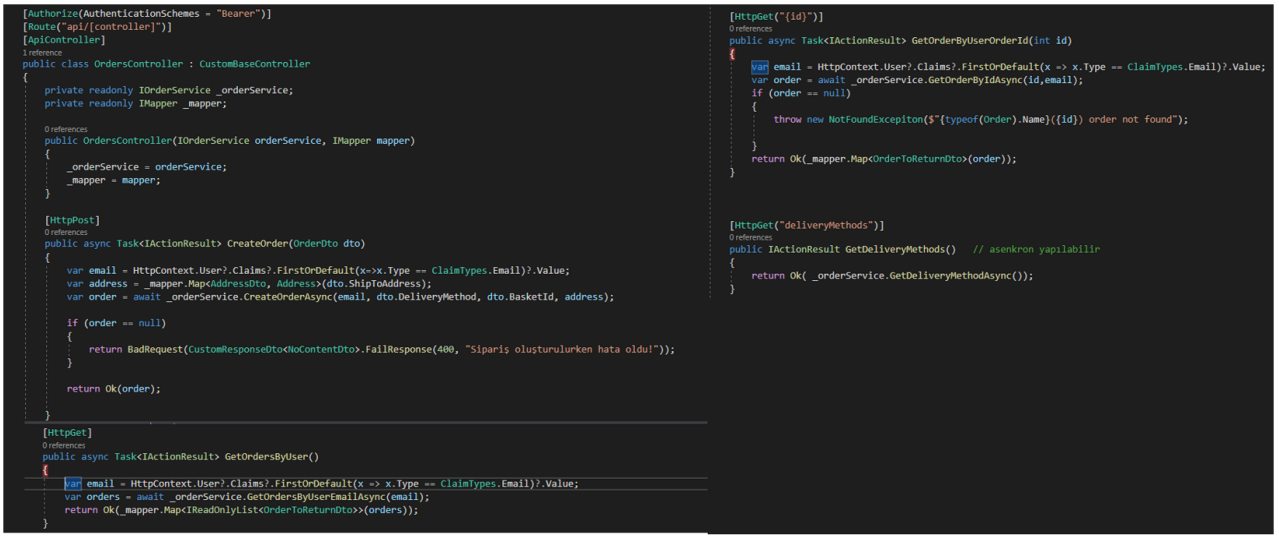
    [Authorize(AuthenticationSchemes = "Bearer")]
    [HttpPost("[action]")]
    0 references
    public async Task<IActionResult> AddProductToBasket(AddProductToBasketDto parameters)
    {
        var email = HttpContext.User?.Claims?.FirstOrDefault(x => x.Type == ClaimTypes.Email)?.Value;
        var user = await _userManager.FindByEmailAsync(email);        // database tarafında doğrula

        var basket =  await _basketService.GetBasketByUserIdAsync(user.Id);
        var addedProduct =  await _basketService.AddProductToBasketAsync(basket.Id, parameters.productId, parameters.quantity);

        var basketDto = _mapper.Map<CustomerBasketDto>(basket);
        return Ok(basketDto);

    }
```

Şekil 3.33. BasketsController görünümü



```
[Authorize(AuthenticationSchemes = "Bearer")]
[Route("api/[controller]")]
[ApiController]
1 reference
public class OrdersController : CustomBaseController
{
    private readonly IOrderService _orderService;
    private readonly IMapper _mapper;

    0 references
    public OrdersController(IOrderService orderService, IMapper mapper)
    {
        _orderService = orderService;
        _mapper = mapper;
    }

    [HttpPost]
    0 references
    public async Task<IActionResult> CreateOrder(OrderDto dto)
    {
        var email = HttpContext.User?.Claims?.FirstOrDefault(x=>x.Type == ClaimTypes.Email)?.Value;
        var address = _mapper.Map<AddressDto, Address>(dto.ShipToAddress);
        var order = await _orderService.CreateOrderAsync(email, dto.DeliveryMethod, dto.BasketId, address);

        if (order == null)
        {
            return BadRequest(CustomResponseDto<NoContentDto>.FailResponse(400, "Sipariş oluşturulurken hata oldu!"));
        }

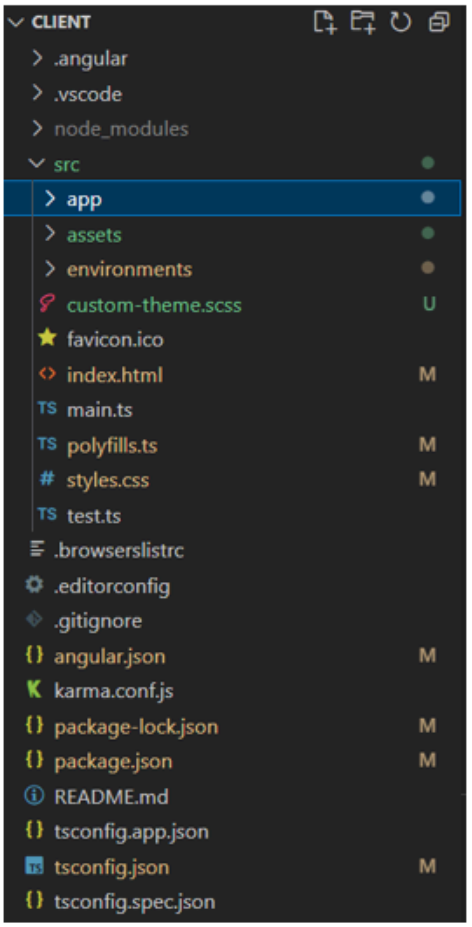
        return Ok(order);

    }
    [HttpGet]
    0 references
    public async Task<IActionResult> GetOrdersByUser()
    {
        var email = HttpContext.User?.Claims?.FirstOrDefault(x => x.Type == ClaimTypes.Email)?.Value;
        var orders = await _orderService.GetOrdersByUserEmailAsync(email);
        return Ok(_mapper.Map<IReadOnlyList<OrderToReturnDto>>(orders));
    }
```

```
[HttpGet("{id}")]
0 references
public async Task<IActionResult> GetOrderByUserOrderId(int id)
{
    var email = HttpContext.User?.Claims?.FirstOrDefault(x => x.Type == ClaimTypes.Email)?.Value;
    var order = await _orderService.GetOrderByIdAsync(id,email);
    if (order == null)
    {
        throw new NotFoundExcepiton($"{typeof(Order).Name}({id}) order not found");
    }
    return Ok(_mapper.Map<OrderToReturnDto>(order));
}


[HttpGet("deliveryMethods")]
0 references
public IActionResult GetDeliveryMethods()   // asenkron yapılabilir
{
    return Ok( _orderService.GetDeliveryMethodAsync());
}
```

Şekil 3.34. OrdersContoller görünümü

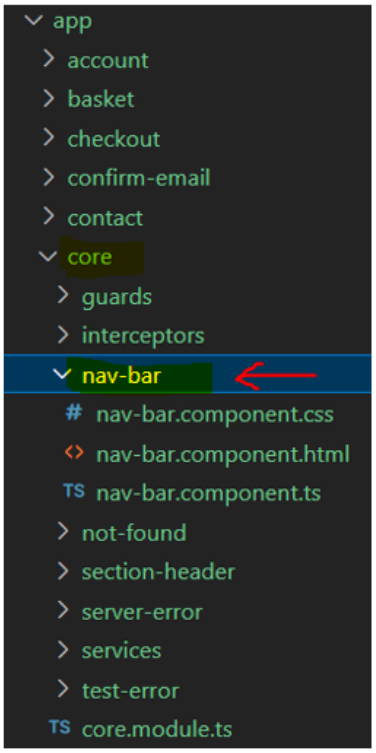


Şekil 5.5. Angular projesi dosya yapısı

```
@NgModule({
  declarations: [
    AppComponent,
    ContactComponent,
    TestComponent

  ],
  imports: [
    BrowserModule,
    //CommonModule,
    AppRoutingModule,
    HttpClientModule,
    CoreModule,
    ShopModule,
    SharedModule,
    BasketModule,
    HomeModule,
    NgxSpinnerModule,
    SwiperModule,
    BrowserAnimationsModule,
    FavoriteModule,
    NgbModule,
    ConfirmEmailModule
```

Şekil 5.6. App.module.ts dosyası görünümü



Şekil 5.7. nav-bar bileşeni (component) oluşturulması

```ts
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { ICheapProducts } from '../shared/models/ICheapProducts';
import { IPagination } from '../shared/models/IPagination';

@Injectable({
  providedIn: 'root'
})
export class HomeService {

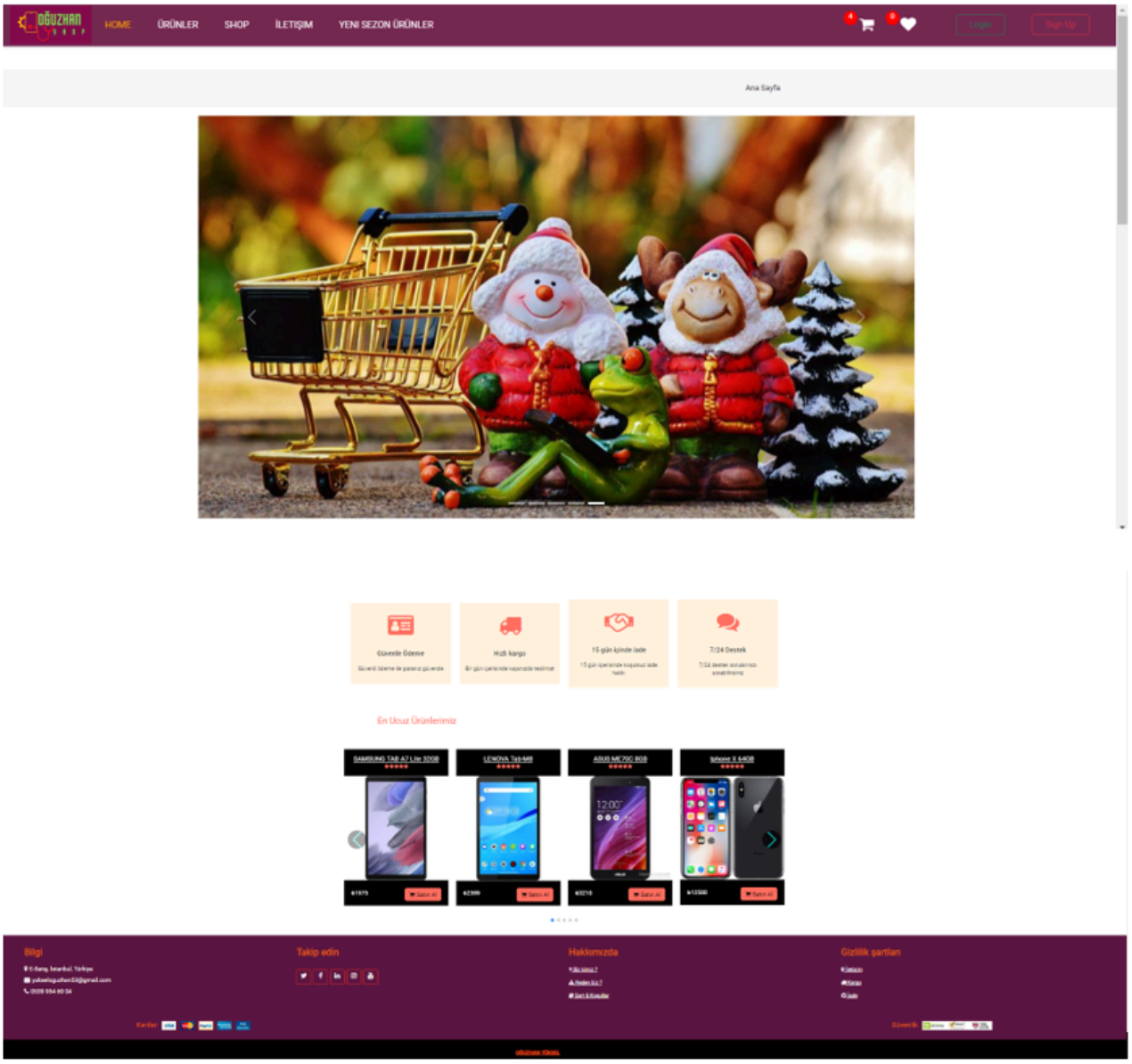
  baseUrl = "https://localhost:44347/api/";
  constructor( private http : HttpClient) {

   }

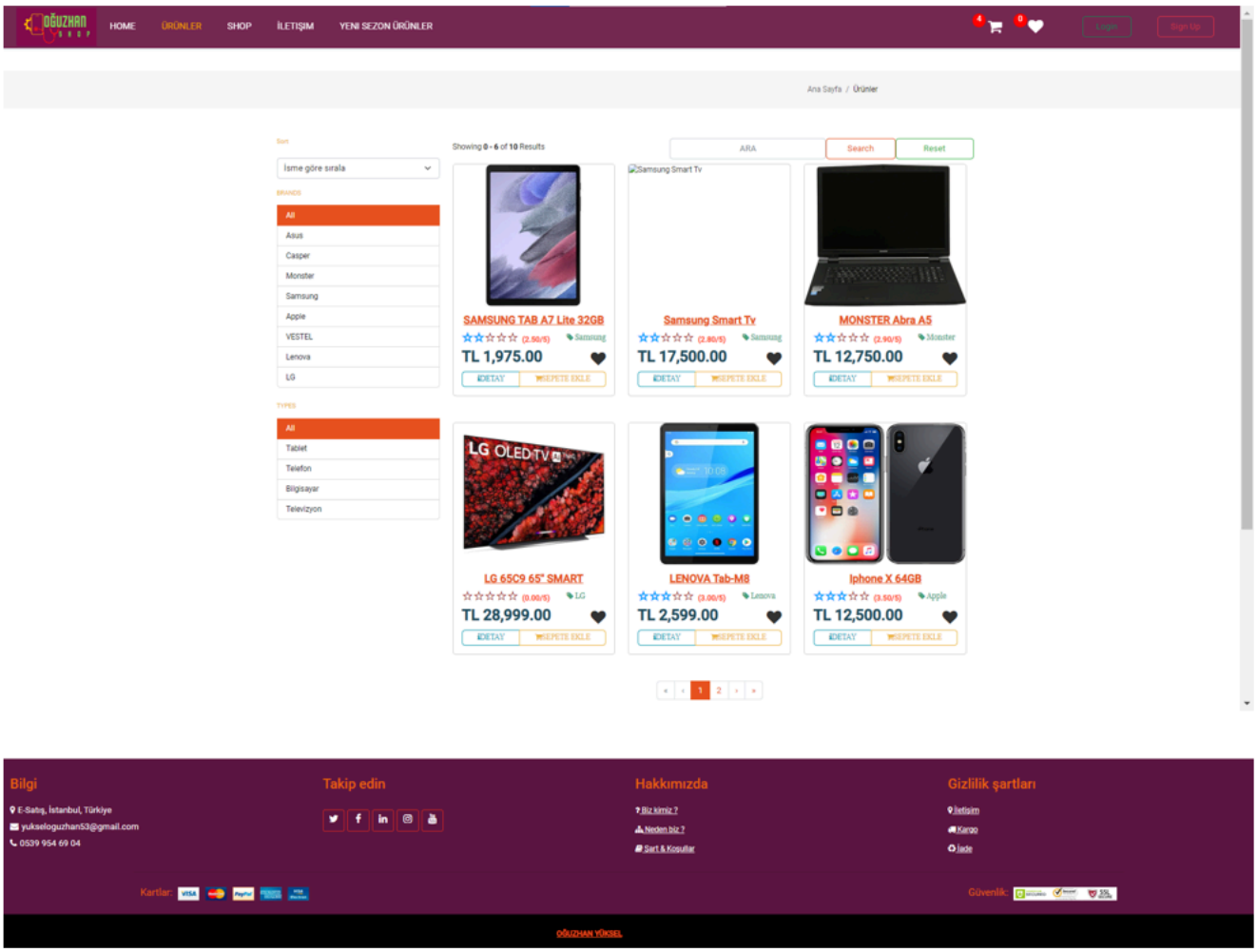
  getCheapProducts()   {
    return this.http.get<ICheapProducts>(this.baseUrl + "Products/cheap-products");
  }


}
```

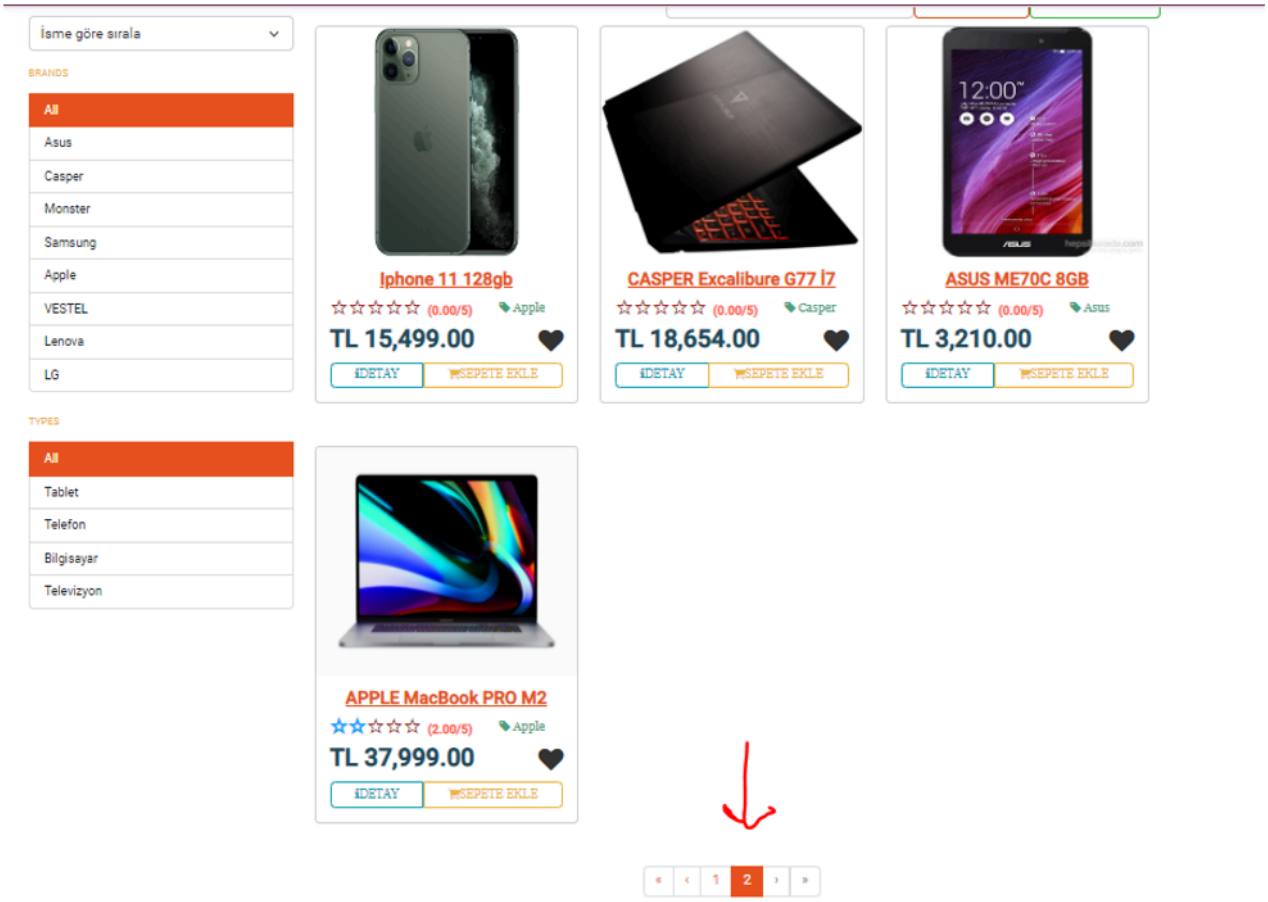
Şekil 5.9. home.service.ts dosyasının oluşturulması



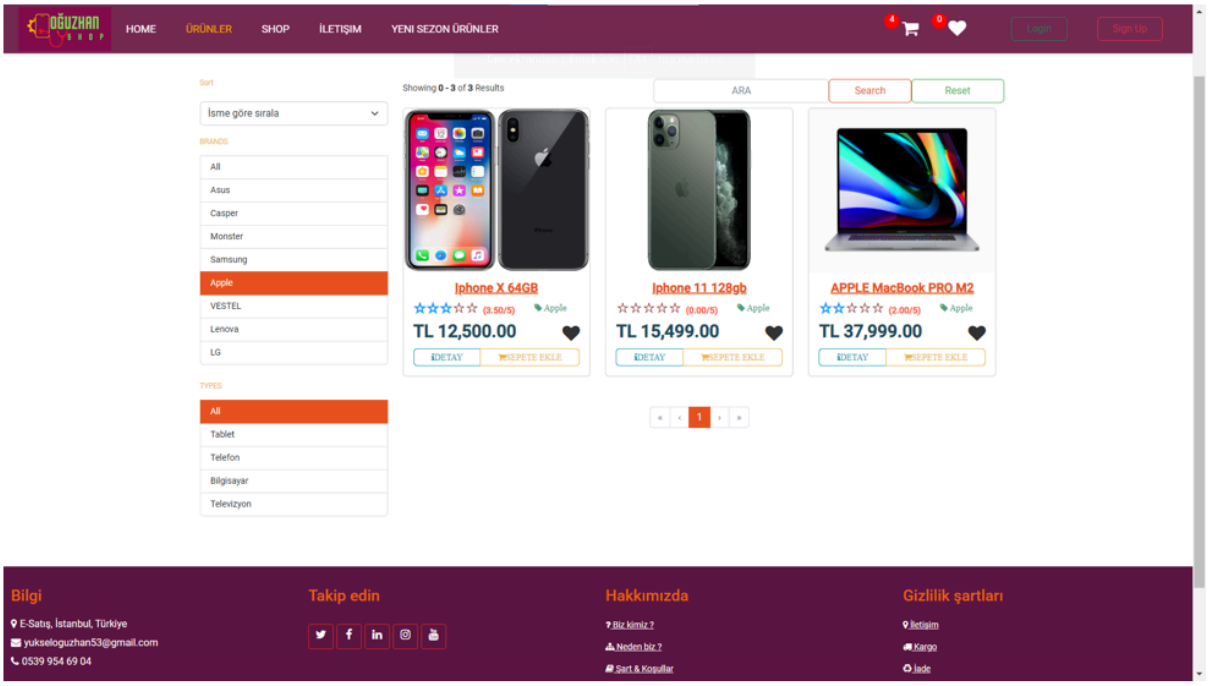
Şekil 5.11. Proje anasayfa görünümü



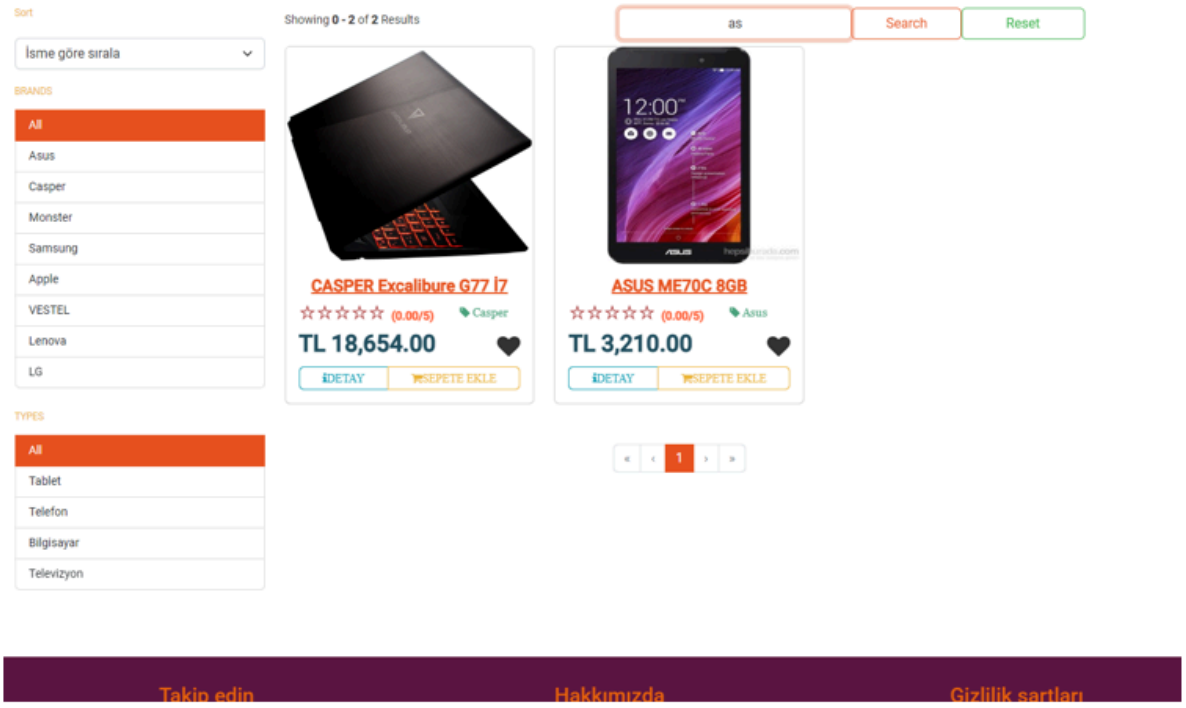
Şekil 5.13. Ürünler sayfası görünümü



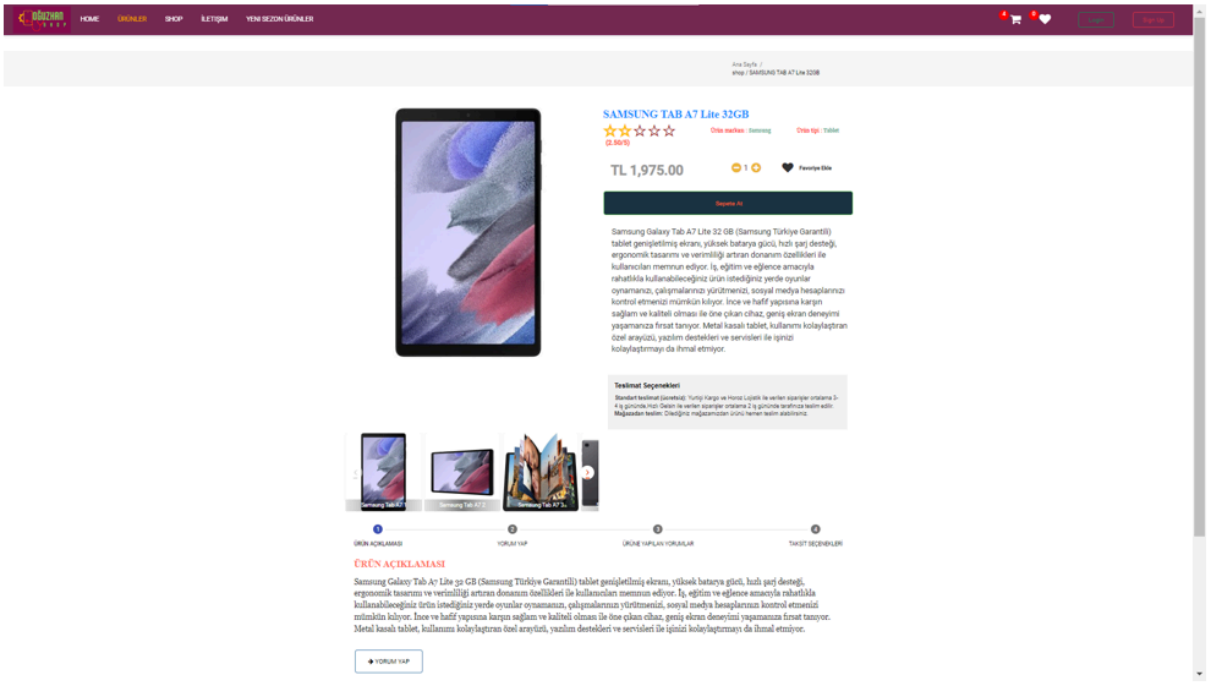
Şekil 5.14. Sayfalama eklenmesi



Şekil 5.15. Projede ürün filtrelemenin gösterilmesi

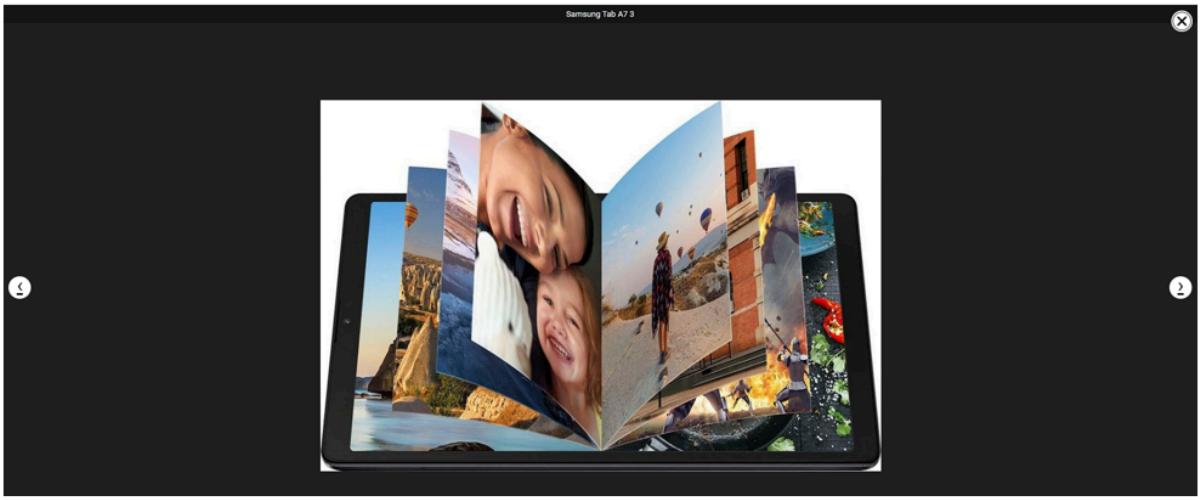


Şekil 5.16. Proje ürün arama işleminin gösterilmesi



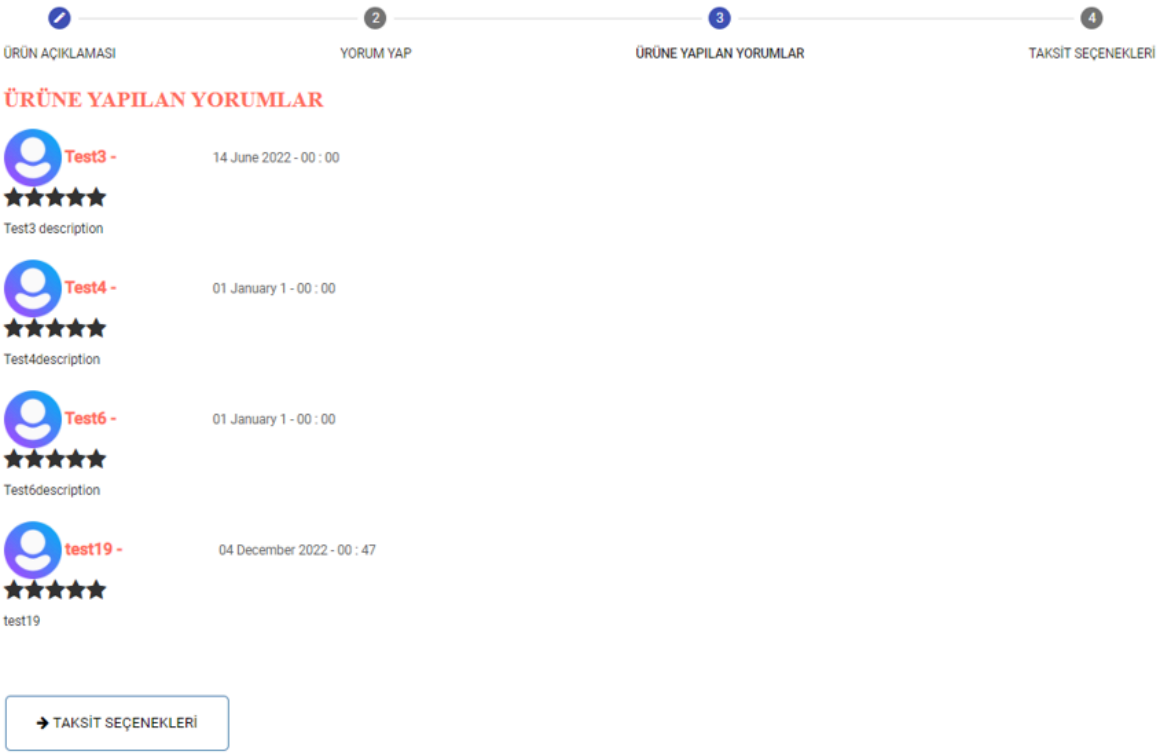
Şekil 5.17. Ürün detay sayfası görünümü

Ürüne ait resimleri görüp daha yakından inceleyebilir.

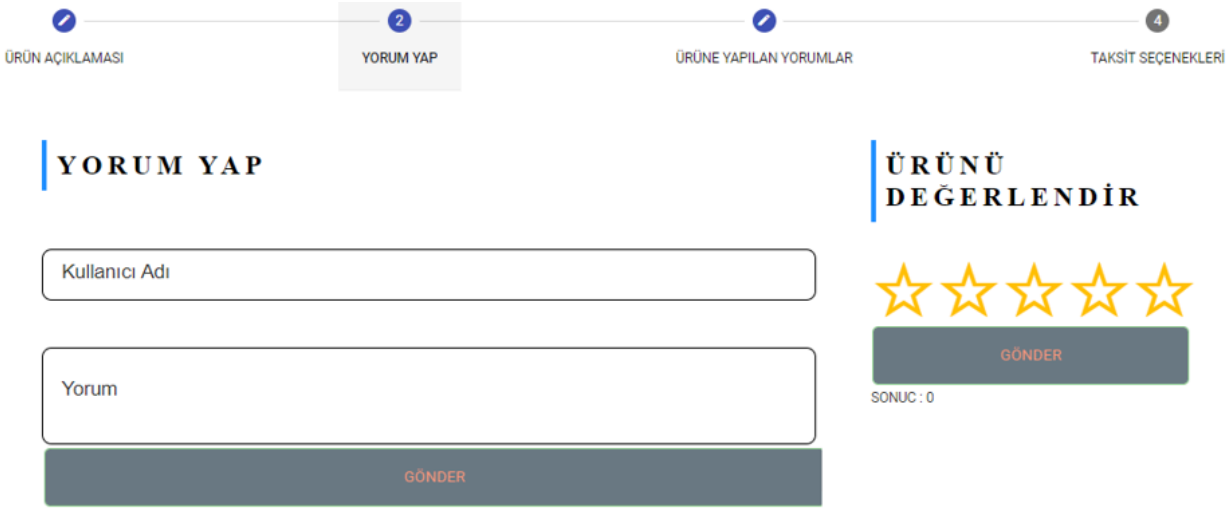


Şekil 5.18. Ürün resimleri görünümü

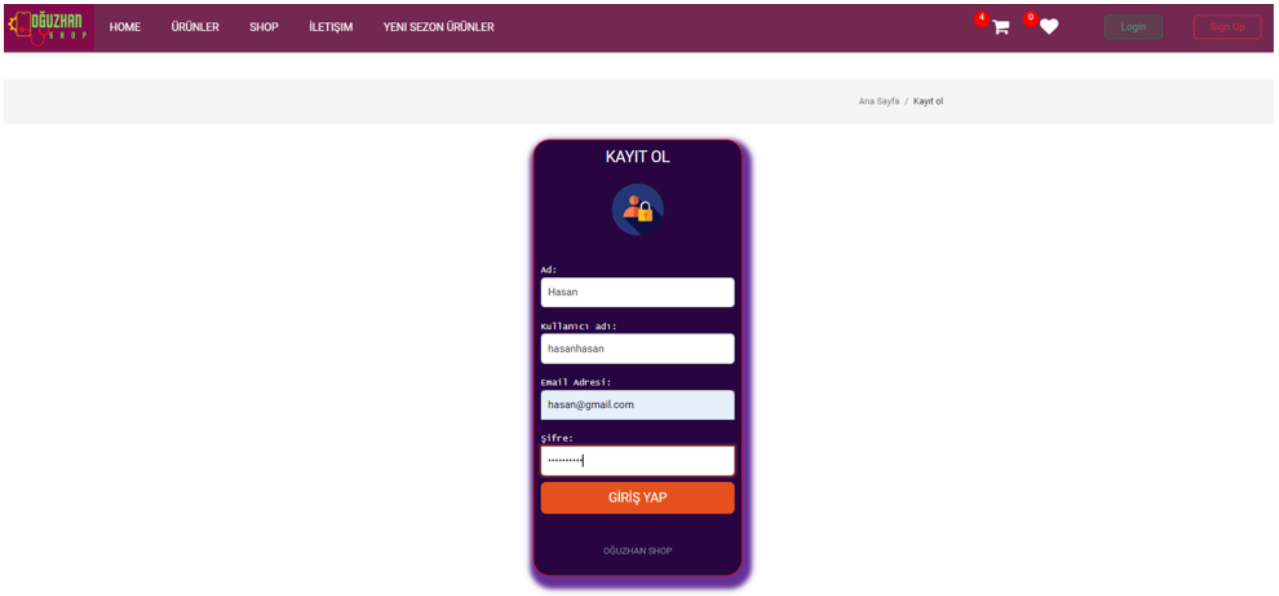
Ürün detayına girdiğinde ürüne ait yorumları görebilir.



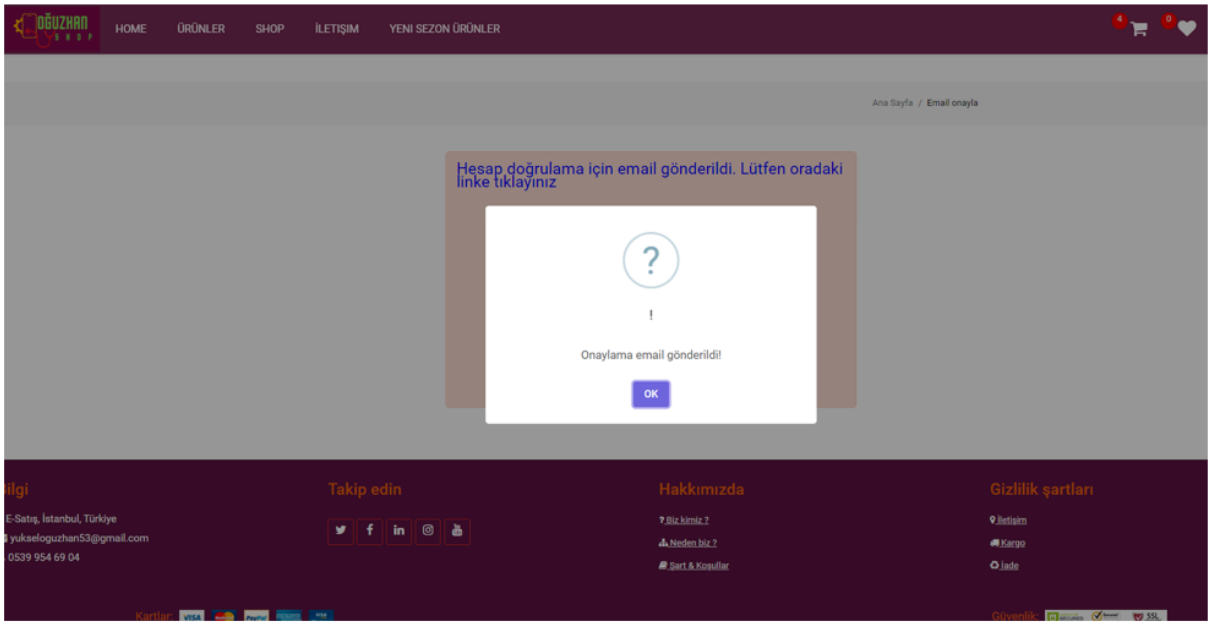
Şekil 5.19. Ürüne yapılan yorumlar



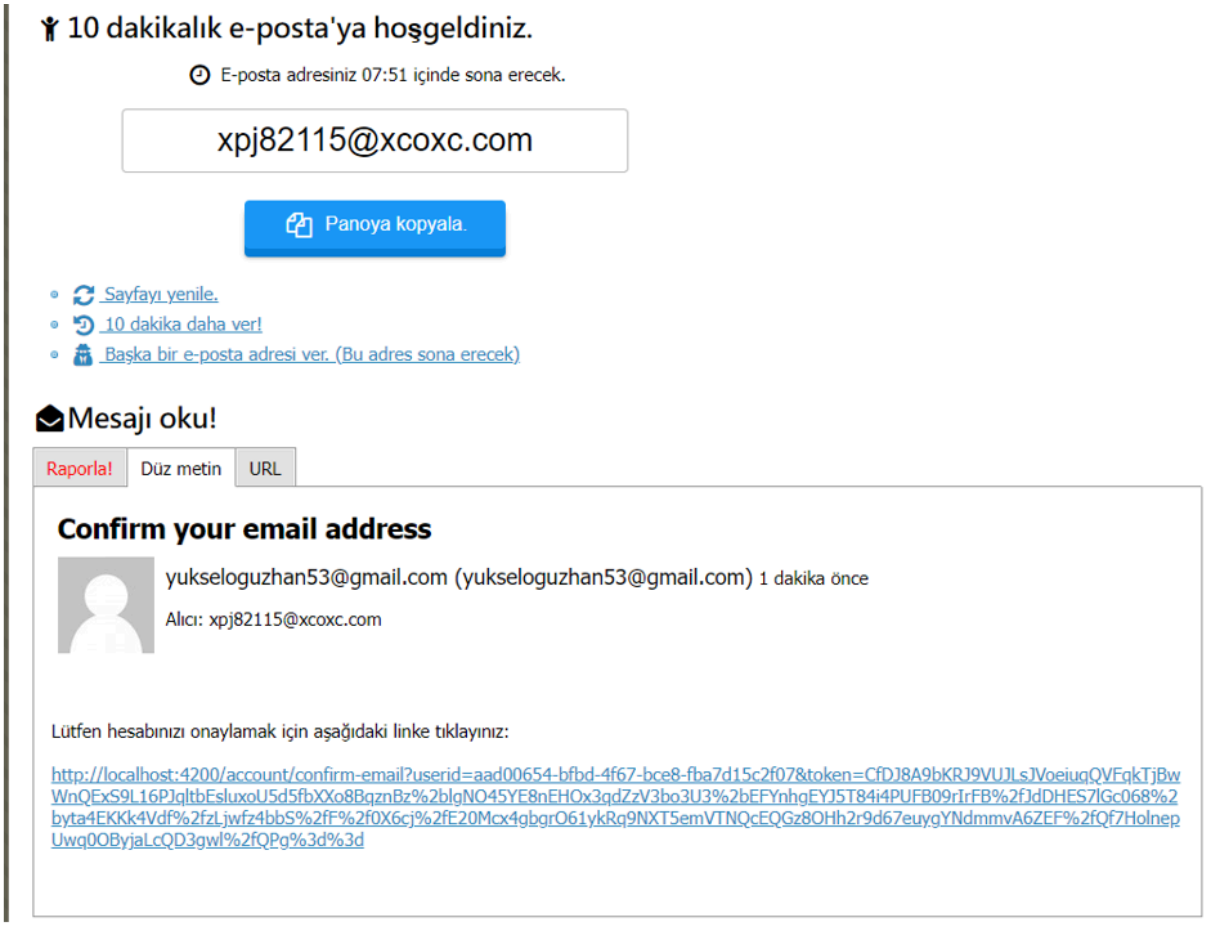
Şekil 5.20. Ürünü değerlendirme ve yorum yapma



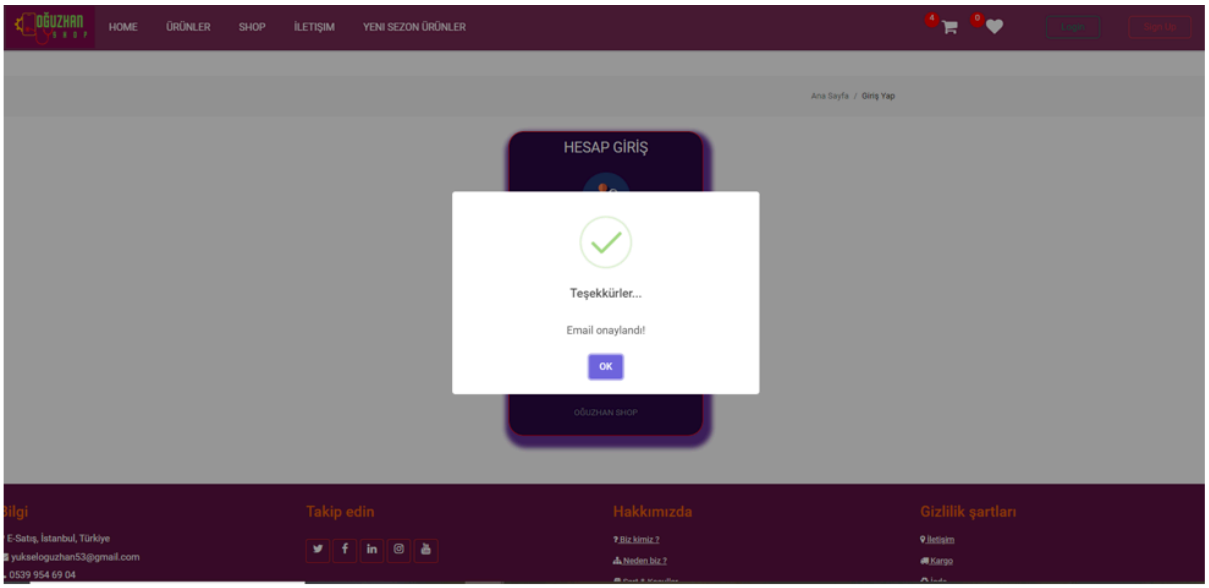
Şekil 5.22. Register sayfa görünümü



Şekil 5.23. Hesap onay email'ı gönderilmesi

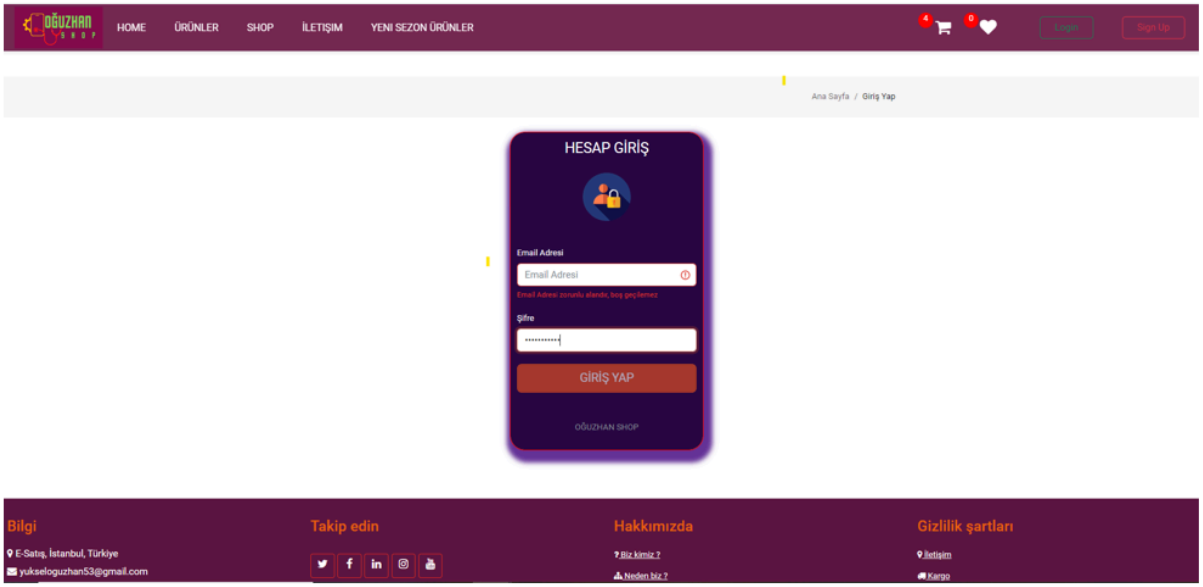


Şekil 5.24. Kullanıcının hesabına onay linki gelmesi

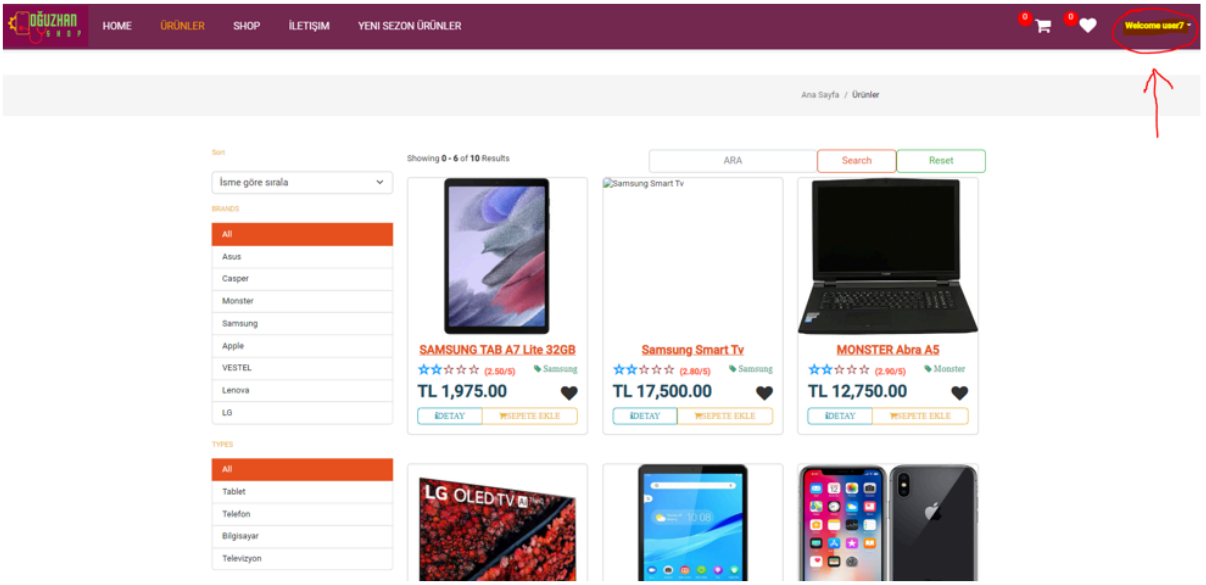


Şekil 5.25. Kullanıcının hesabını onaylaması

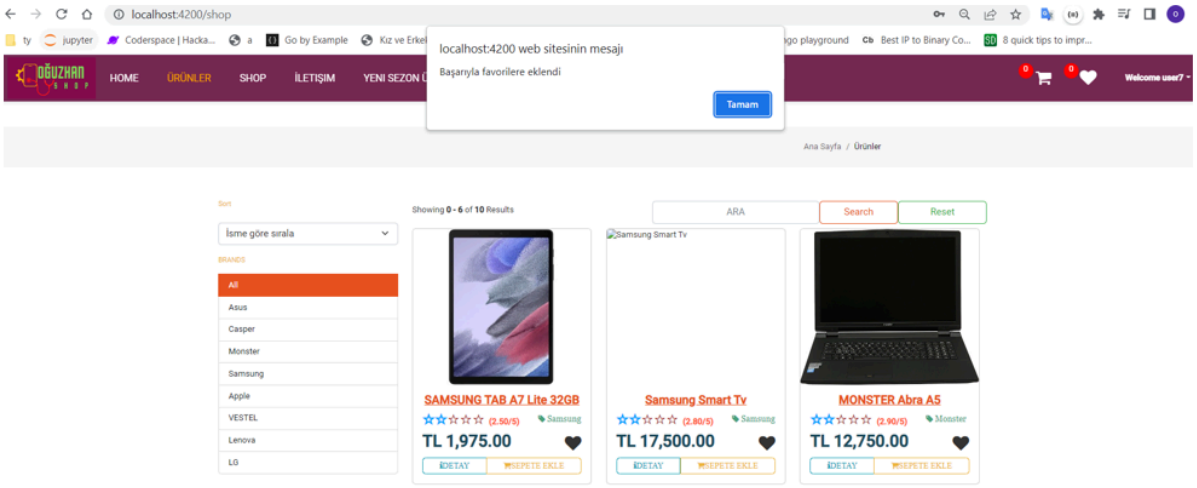
Daha sonra üye bilgilerini (email ve şifre) girdikten sonra sisteme girebilir.



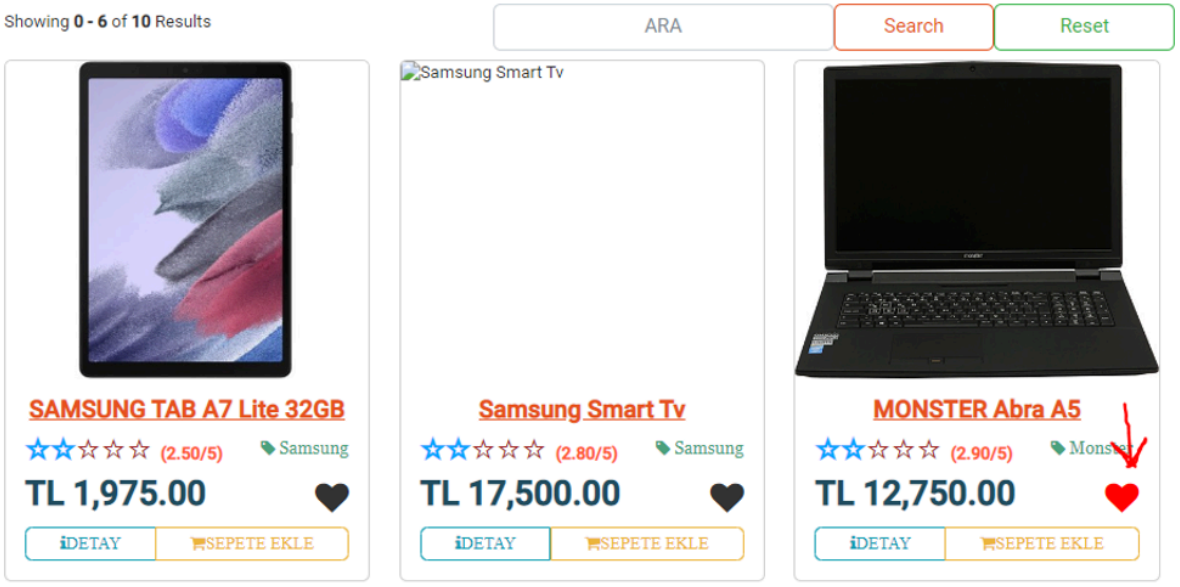
Şekil 5.26. Hesap girişi ekranı



Şekil 5.27. Kullanıcının giriş yapması



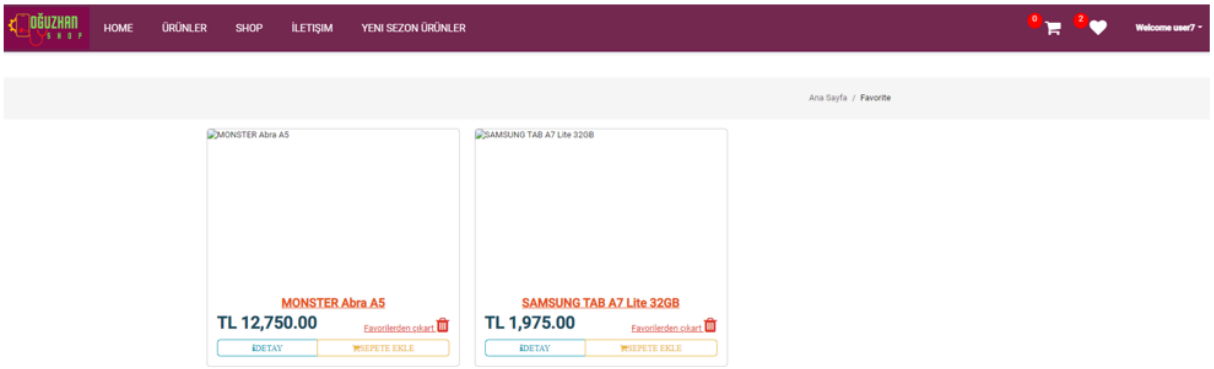
Şekil 5.29. Ürünü favorilere ekleme



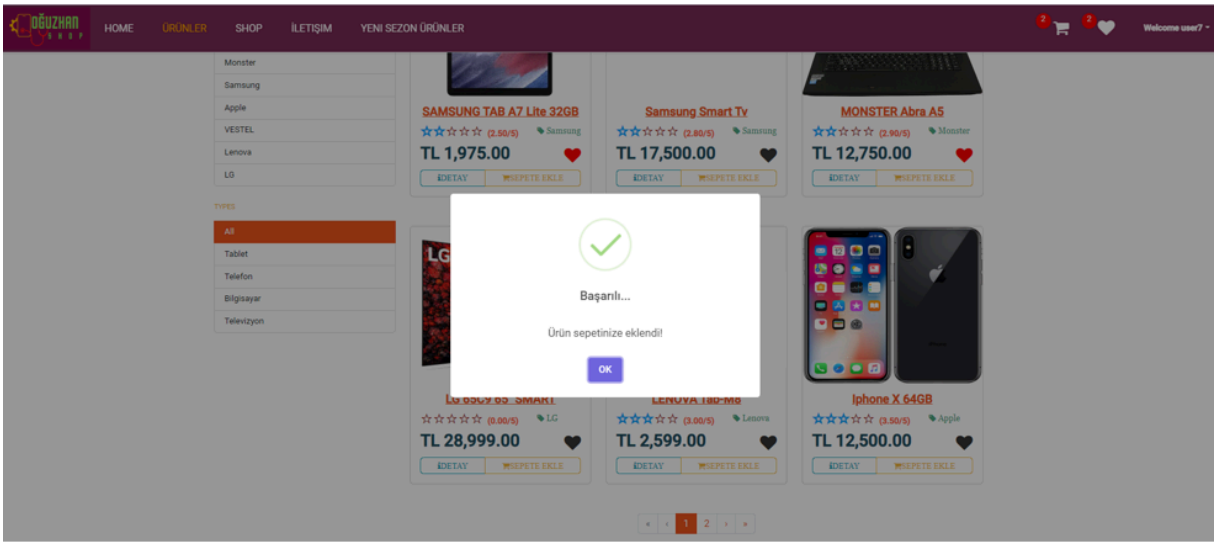
Şekil 5.30. Favorilere eklenmiş ürün görünümü



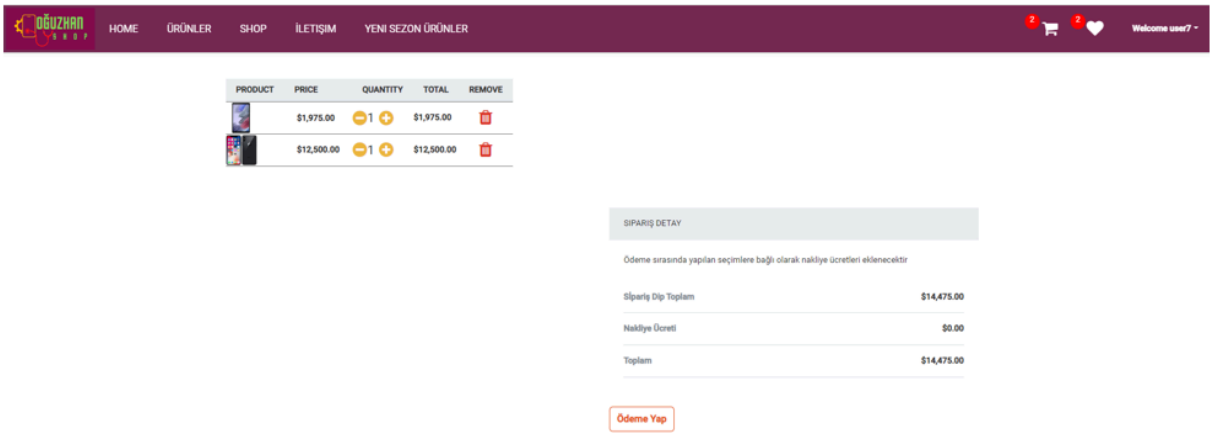
Şekil 5.31. Kullanıcının favori ürün sayısı



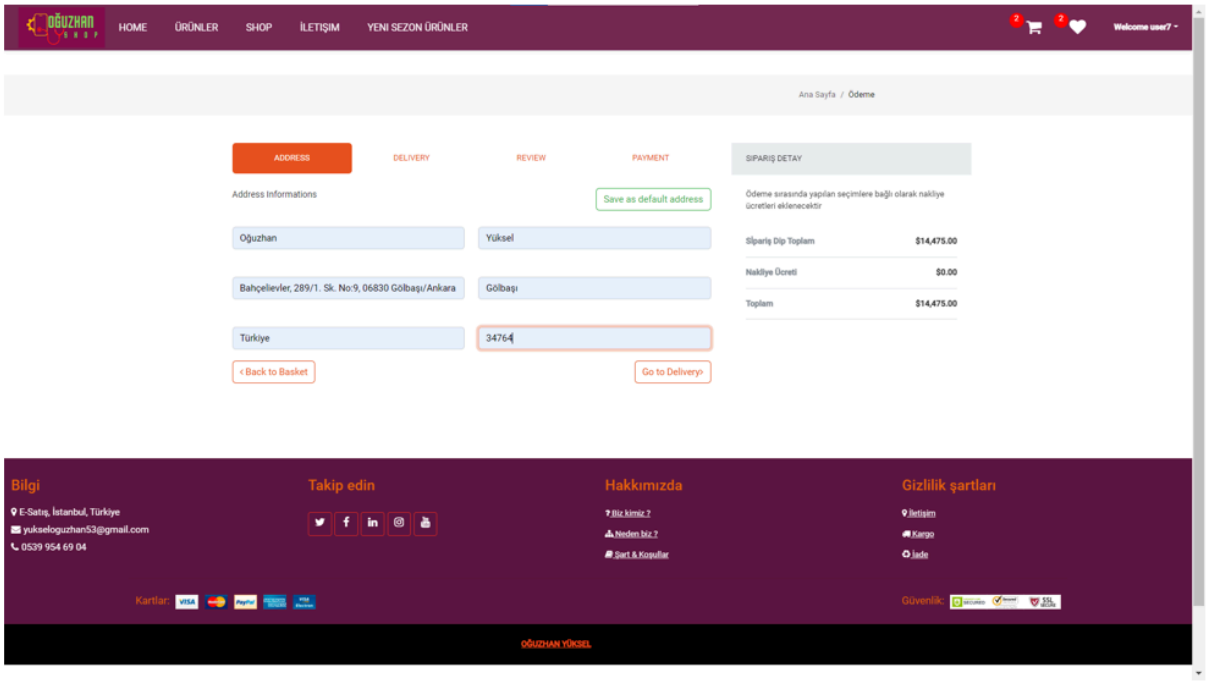
Şekil 5.32. Kullanıcının favori listesi



Şekil 5.33. Kullanıcının ürünü sepetine eklemesi

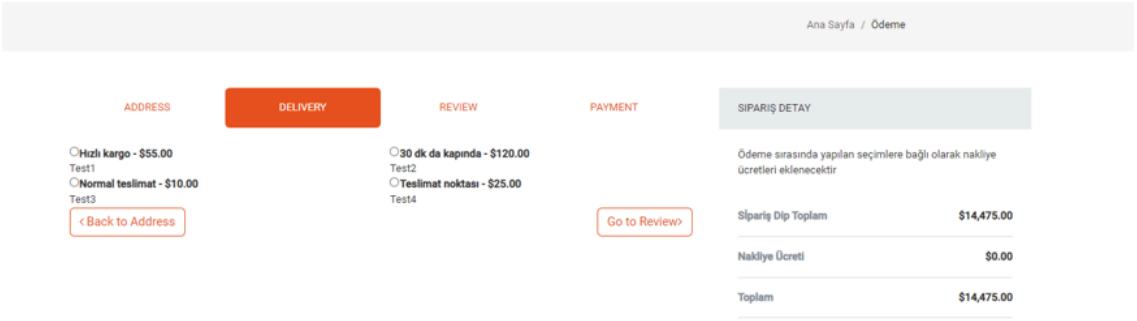


Şekil 5.34. Kullanıcının sepeti görünümü



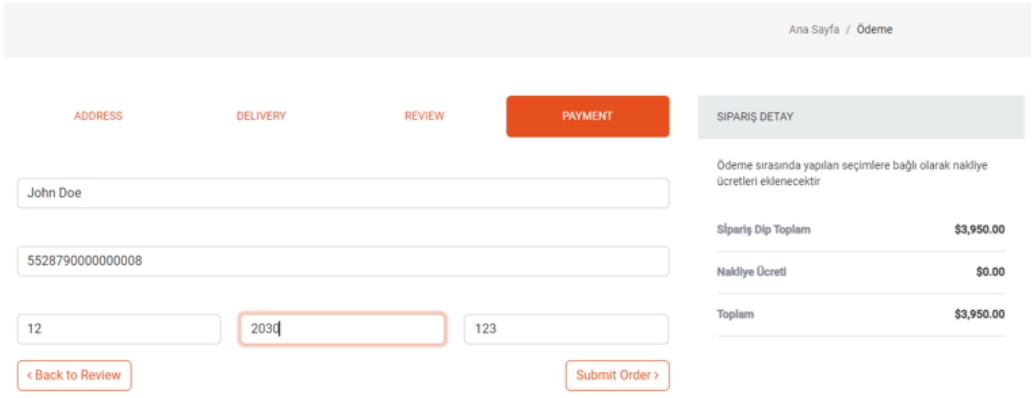
Şekil 5.36. Sipariş vermek için adres bilgileri girilmesi

Kullanıcı aşağıdaki gibi kargo seçeneklerinden uygun olanı seçer.



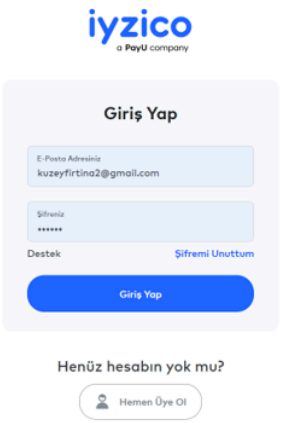
Şekil 5.38. Kullanıcının kargo seçeneği seçme ekranı

Sipariş vermek isteyen kullanıcı kart bilgilerini girer ve siparişini vermiş olur.

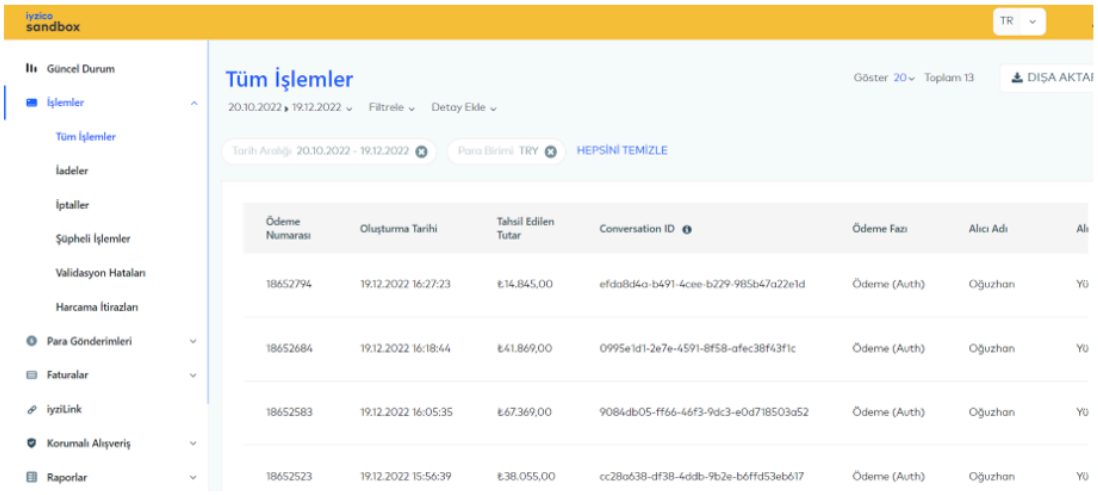


Şekil 5.39. Kullanıcının kart bilgilerini girme ekranı

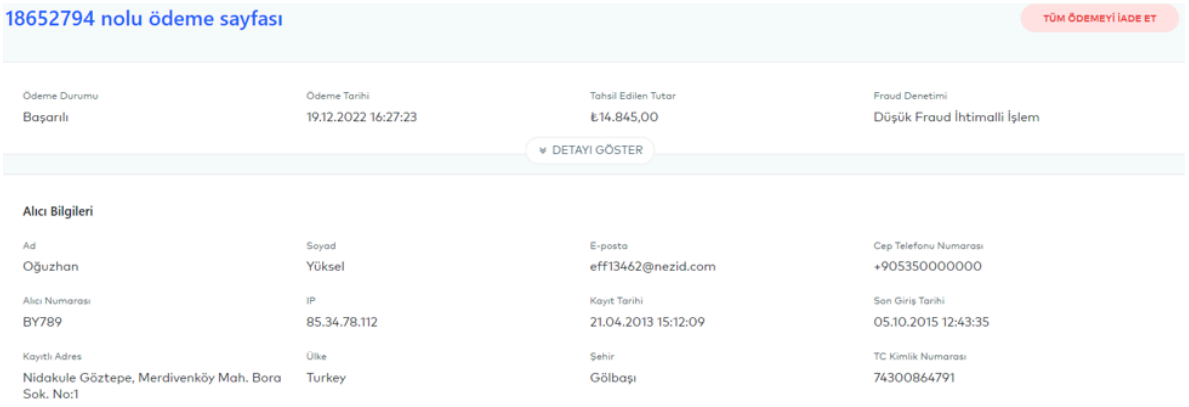
Ödemeleri control için IYZICO ödeme servisini kullandım.



Şekil 5.40. İyzico'ya giriş yapma

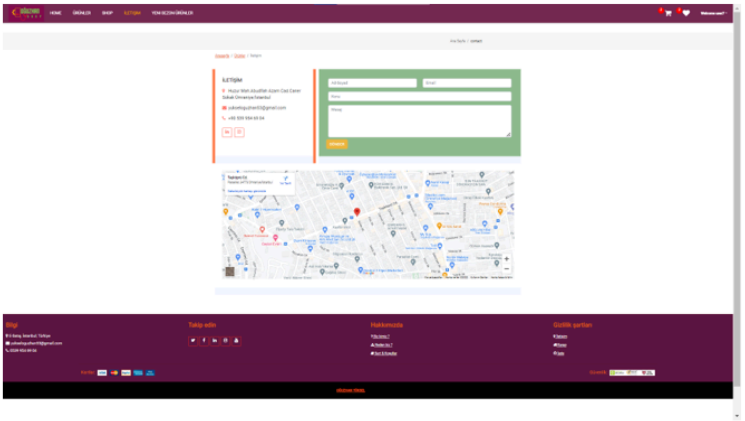


Şekil 5.41. Yapılan ödeme listesi



Şekil 5.42. Yapılan ödeme detayını görme

Kullanıcıların istek ve taleplerini iletmesi için iletişim sayfası oluşturdum.



Şekil 5.43. İletişim sayfası tasarımı