PROJE RAPORU

PROJE

RAPOR TARİHİ	PROJE ADI	HAZIRLAYAN:
22 Haziran 2022	Depo Ürün Yönetim Sistemi	Sezer VAROL Gökhan HASBAL

PROJE ÖZETİ

Bir şirketin dışarıdan aldığı ürünleri veya kendi ürünlerini depolayabileceği ve sevkiyat yapabileceği, VarBal Storage isimli bir MVC projesidir. Sezer **Var**ol ve Gökhan Has**bal** tarafından 31 Mayıs – 23 Haziran arasında geliştirildi.

PROJENİN BÖLÜMLERİ

İLERLEME	YAPILAN İŞLEMLER
MVC , ENTİTYFRAMEWORK YAPISININ OIUŞTURULMASI	Visual Studio programıyla Asp.Net Web Applicaition .net framework projesi oluşturuldu .
TEMPLATE ARAŞTIRILMASI	Themeforest'ten kendi projemize uygun taslak bulundu.
TEMPLATE'İNPROJEYE AKTARILMASI	Entity framework'un içinde default halde gelen Content sayfasının tüm içeriği silinip template ait dosyalar buraya aktarıldı ve tüm href ,src linkleri kendi kullanacağımız adreslere göre güncelledik , yine default halde gelen fonts ve js dosyaları silindi projeden, ki mevcut olanla karışmasın.
TEMPLATE'İN ÇALIŞIR HALE GETİRİLMESİ	Templatein çalışır hale gelmesi için öncelikle ana indexi belirleyip index sayfasından bazı elemeler yapılması gerekmektedir. Bu elemeler neticesinde tüm sayfaları içinde barındırıcak olan layout sayfası oluşturulur ve bu sayfada navbar ve footer vardır.
LAYOUT YAPISININ ŞEKİLLENDİRİLMESİ	Önce kullanılacak olan navbar yapısı ve footer kodları belirlenerek sayfaya enjekte edilir. @renderbody'in altı ve üstü kullanılır renderbodynin üstü navbar altı footer yapısını barındırır.
MODEL NESNELERİNİN SİSTEMATİK ŞEKİLDE ADLANDIRILMASI	Kullanacağım model'ler oluşturup foreign key bağlantıları oluşturuldu içermesi gereken bilgiler aktarıldı. Bu modeller neticesinde controllerlar oluşturuldu.
CONTROLLERLARIN OLUŞTURULMASI	Kullandığımız modeller neticesinde controllerlar oluşturulup isimlendirilmesi yapıldı ve bu controllerların barındırdığı sayfalar tek tek oluşturuldu. Sayfalara modeller çağırılıp gerekli ilişkilendirilmeler yapıldı.
VERİTABANI İŞLEMLERİNİ ADIMLARI	Veri tabanı işlemleri için webconfig sayfasında yapılması gereken birkaç adım mevcuttur <connectionstring> html tadıyla database adı vs. tanımlanır. Datacontext yapısı

	gene burada tanımlanır datacontext i tüm sayfalardaki veri işlemlerinde kullanırız. O yüzden sayfalara çağırdığımız kütüphanelere falan dikkat etmemiz gerekmektedir.
VERİ TABANININ OLUŞTURULMASI	Projede kullanmam gereken veritabanı için modeller sayfama bir datacontext modeli oluştururuz bu datacontext modelininin içinde dbset kodu ile modelleri database'e set ederiz.
VERİ TABANININ OLUŞTURULMASI	Package Manager Console'dan migration yapısı çağırılır. 1-)Enable-Migrations 2-)add-migration Varbal 3-)add-migration Varbal 4-)update-database yapılarak veritabanı oluşturulur.
VERİ TABANINDAKİ TABLOLAR	About,About Feature,Address,Booking,Brand,Category,Contact Us,Invoice,Logs,NewsGet,Order,Our Worker,Permission,Seo,Shelf,Shipment,User,Vehicle,Vision And Mission ,Product,Slider, ,Contact tabloları mevcuttur bunlar her biri modellerin içinde hangi columnları olacaksa yazılıp kodlandırılması gerçekleşir properity yapılarıyla.
SAYFA YAPILARININ OLUŞTURULMASI	Controller'daki action result'ta bulunan sayfalar kullanmamız gerektiği gibi (varsa adminlayout, userlayout, layout veya layoutsuz olarak oluşturulur).
SAYFALARIN DÜZENLENMESİ	Hangi sayfada ne yapılması isteniyorsa ona clean kod mantığıyla sayfalarımız kodlanır ve sayfamızda kullanacak veritabanından gelecek veriler varsa modeller yardımıyla çağırılır.
VERİ EKLEME,GUNCELLEME İŞLEMLERİNİN OLUŞTURULMASI	Gerekli controller'larla ekleme silme düzenleme işlemleri gerçekleştirme işlemleri yapılmaktadır.

Varbal Storage Zaman Çizelgesi

Başlangıç Tarihi: 31.05.2022

Yapılan İşlemler:	hafta 1							hafta 2							hafta 3							hafta 4						
	may	haz						haz							haz							haz						
	sal	çar	per	cum	cmt	paz	pzt	sal	çar	per	cum	cmt	paz	pzt	sal	çar	per	cum	cmt	paz	pzt	sal	çar	per	cum	cmt	paz	pzt
	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Araştırma Geliştirme	x	x																										
Modeller Tasarlandı			x																									
Template Bulunup Düzenlendi				x	x		x																					
Controllers Oluşturuldu							x																					
Kullanıcı Kayıt İşlemleri								x	x																			
Giriş Çıkış İşlemleri								x	x																			
İlgili Actionlar Oluşturuldu										x	x	x																
Layout Oluşturulup Düzenlendi														x														
Layout'ta Görüncek Sayfaların Tasarımı Yapıldı														x														
Navbar ve Footer Bölümlerinde Goruncekler Belirlendi														x														
Oluşturulan Sayfa Tasarımları Navbar ve Footerda Görüncek Hale Getirildi															x													
Home,About Us,Contact Us,Locations,Booking, My Pages Sayfalarının Tüm Tasarımları Gerçekleştirilip İlgili Actionlarında Yerlerini Alması Sağlanmıştır.																x	x	x	x									
Tüm Oluşturduğumuz Bu Sayfalar İçin Admin Panel ve Admin_Layout Oluşturuldu ve Düzenlendi																					x	x	x	x				
Proje Sunumu Hazırlandı ve Sunum Yapıldı																								x				

Projemizde Kod Yapılarımıza Örnekler Aşağıda Gösterildiği Gibidir.

CRUD (create-read-update-delete) işlemlerimize örnek

Listeleme ve değer ekleme

```
public ActionResult Index()
{
    var category = db.Category.Where(x => x.Delete == false).ToList();
    return View(category);
}
[HttpGet]
0 references
public ActionResult Add()
{
    ViewBag.category = db.Category.ToList();
    return View();
}
[HttpPost]
0 references
public ActionResult Add(Category category)
{
    Category newcategory = new Category();
    newcategory.Name = category.Name;
    newcategory.Description = category.Description;
    newcategory.MainCategoryId = category.MainCategoryId;
    newcategory.Status = category.Status;
    db.Category.Add(newcategory);
    db.SaveChanges();
    ViewBag.category = db.Category.ToList();
    ViewBag.mesaj = "Kategori Ekleme Başarılı";
    return View();
}
```

Basit, birden fazla tablo bilgileri gerektirmeyen actionlarımız için, uygun modeller ve LINQ sorgularıyla bilgiler listelendi.
ekleme ksıımlarında bu tablolar, formlar aracılığıyla alınıp, database'imize kaydedildi.

```
[HttpGet]
0 references
public ActionResult Edit(int Id)
{
    var category = db.Category.Find(Id);
    if (category != null)
    {
        ViewBag.category = db.Category.ToList();
        if (category.MainCategoryId != 0)
        {
            ViewBag.maincategory = db.Category.FirstOrDefault(x => x.Id == category.MainCategoryId).Name;
        }
        return View(category);
    }
    else
    {
        //return Redirect("~/Category");
        return RedirectToAction("index");
    }
}
[HttpPost]
0 references
public ActionResult Edit(Category category, HttpPostedFileBase Image)
{
    var editcategory = db.Category.Find(category.Id);
    editcategory.Name = category.Name;
    editcategory.Description = category.Description;
    editcategory.Status = category.Status;
    editcategory.MainCategoryId = category.MainCategoryId;
    string imagepath = "";
    string imagename = "";
    try
    {
        if (Image != null && Image.ContentLength > 0)
        {
            imagename = Guid.NewGuid().ToString().Substring(0, 10) + "-" + Path.GetFileName(Image.FileName);
            imagepath = Path.Combine(Server.MapPath("~/Content/images/CategoryImage"), imagename);
            Image.SaveAs(imagepath);
            editcategory.Image = imagename;
        }
        ViewBag.mesaj = "Kategori Düzenleme Başarılı";
    }
    catch
    {
        ViewBag.mesaj = "Unexpected Error";
    }
    db.SaveChanges();
    return Redirect("~/Category/Edit?id=" + category.Id);
}
```

Düzenleme kısmında ise, düzenlenecek tablo girdisinin id'si tutuldu, Find methodu ile bulunup cshtml kısmında yine form vasıtasıyla yeni girdiler alınarak database güncellendi. bazı controllerlara gösterildiği gibi fotoğraf eklenme özelliği konuldu.

```
[HttpGet]
0 references
public ActionResult Edit(int Id)
{
    var category = db.Category.Find(Id);
    if (category != null)
    {
        ViewBag.category = db.Category.ToList();
        if (category.MainCategoryId != 0)
        {
            ViewBag.maincategory = db.Category.FirstOrDefault(x => x.Id == category.MainCategoryId).Name;
        }
        return View(category);
    }
    else
    {
        //return Redirect("~/Category");
        return RedirectToAction("index");
    }
}
[HttpPost]
0 references
public ActionResult Edit(Category category, HttpPostedFileBase Image)
{
    var editcategory = db.Category.Find(category.Id);
    editcategory.Name = category.Name;
    editcategory.Description = category.Description;
    editcategory.Status = category.Status;
    editcategory.MainCategoryId = category.MainCategoryId;
    string imagepath = "";
    string imagename = "";
    try
    {
        if (Image != null && Image.ContentLength > 0)
        {
            imagename = Guid.NewGuid().ToString().Substring(0, 10) + "-" + Path.GetFileName(Image.FileName);
            imagepath = Path.Combine(Server.MapPath("~/Content/images/CategoryImage"), imagename);
            Image.SaveAs(imagepath);
            editcategory.Image = imagename;
        }
        ViewBag.mesaj = "Kategori Düzenleme Başarılı";
    }
    catch
    {
        ViewBag.mesaj = "Unexpected Error";
    }
    db.SaveChanges();
    return Redirect("~/Category/Edit?id=" + category.Id);
}
```

Tek buton tıklanmasıyla verilerin status değişimi ve delete işleminin yapılması için uygun actionlar açılıp, index.cshtml içerisine yerleştirildi.

```
public ActionResult Delete(int Id)
{
    var category = db.Category.Find(Id);
    category.Delete = true;
    db.SaveChanges();
    return RedirectToAction("index");
}
0 references
public ActionResult CategoryStatusEdit(int Id)
{
    var category = db.Category.Find(Id);
    category.Status = !category.Status;
    db.SaveChanges();
    return RedirectToAction("index");
}
```

Daha detaylı sayfalar için, şekilde görüldüğü gibi çok amaçlı modeller oluşturuldu yine uygun sorgularla bu modeller doldurulup, ilgili yerlerde kullanılmak üzere view ile cshtml kısmına aktarıldı

```
namespace VarBal.AdminModel
{
    2 references
    public class IndexModel
    {
        1 reference
        public List<User> User { get; set; }
        0 references
        public List<Permission> Permission { get; set; }
        1 reference
        public List<Shipment> Shipment { get; set; }
        0 references
        public List<Contact> Contact { get; set; }
        1 reference
        public List<ContactUs> ContactUs { get; set; }
        0 references
        public List<Vehicle> Vehicles { get; set; }
        1 reference
        public List<Warehouse> Warehous { get; set; }
        1 reference
        public List<Product> Product { get; set; }
        1 reference
        public int ProductCount  { get; set; }
        1 reference
        public int ProductCountTrue  { get; set; }
        1 reference
        public int VehiclesCount { get; set; }
        1 reference
        public int VehiclesCountTrue { get; set; }
        1 reference
        public int UserTotalCount { get; set; }
        1 reference
        public int UserActiveCount { get; set; }
        1 reference
        public int UserPasiveCount { get; set; }
    }
}
```

```
0 references
public ActionResult Index()
{
    IndexModel model = new IndexModel()
    {
        User = db.User.ToList(),
        UserTotalCount = db.User.Where(x => x.Delete == false).Count(),
        UserActiveCount = db.User.Where(x => x.Status == true && x.Delete == false).Count(),
        UserPasiveCount = db.User.Where(x => x.Status == false && x.Delete == false).Count(),



        ContactUs = db.ContactUs.Take(6).ToList(),//son 6 malzeme
        Shipment = db.Shipment.Take(6).ToList(), //son 6 malzeme
        Warehous = db.Warehouse.ToList(),

        Product = db.Product.ToList(),
        ProductCountTrue = db.Product.Where(x => x.Status == true && x.Delete == false).Count(),
        ProductCount = db.Product.Where(x => x.Delete == false).Count(),

        VehiclesCount = db.Vehicle.Where(x =>x.Delete == false).Count(),
        VehiclesCountTrue= db.Vehicle.Where(x => x.Status == true && x.Delete == false).Count(),


    };

    var Permision = db.Permission.ToList();


    return View(model);
}
```

Admin panelde bulunan notification kısmı işlevsel hale getirildi, ve görülmesi istenen verilerin sorgular yardımıyla çağrılması sağlandı. bu bildirimlerin, tıklandığında ilgili sayfanın detayına gitmesi, ve view değerlerinin otomatik değiştirilip bu bildirim listesinden düşmesi sağlandı. Notification kısmına, bütün bildirimleri tek tıklama ile kaldıracak bir buton tasarlandı.

```
0 references
public ActionResult Notification()
{
    var NewsGet = db.NewsGet.Where(x => x.Delete == false && x.View == false).ToList();
    var User = db.User.Where(x => x.Delete == false && x.View == false).ToList();
    var Product = db.Product.Where(x => x.Delete == false && x.View == false).ToList();
    var Order = db.Order.Where(x => x.View == false).ToList();
    var Shipment = db.Shipment.Where(x => x.Delete == false && x.View == false).ToList();
    var ContactUs = db.ContactUs.Where(x => x.Delete == false && x.View == false).ToList();
    var Booking = db.Booking.Where(x => x.Delete == false && x.View == false).ToList();

    List<Notification> notifications = new List<Notification>();
    Notification newnotification;

    foreach (var item in NewsGet)...
    foreach (var item in User)...
    foreach (var item in Order)...
    foreach (var item in Product)...
    foreach (var item in Shipment)...
    foreach (var item in Booking)
    {
        newnotification = new Notification();
        newnotification.Icon = "mdi-archive";
        newnotification.NameSurname = item.NameSurname;
        newnotification.Content = item.Note;
        newnotification.Color = "bg-dark";
        newnotification.Link = "Booking/Detail?id=" + item.Id;
        notifications.Add(newnotification);
    }
    foreach (var item in ContactUs)...
    var model = notifications.OrderByDescending(x => x.Date).ToList();
    return PartialView(model);
}
0 references
public ActionResult ClearNotification()
{
    db.NewsGet.Where(x => x.Delete == false && x.View == false).ToList().ForEach(x => x.View = true);
    db.User.Where(x => x.Delete == false && x.View == false).ToList().ForEach(x => x.View = true);
    db.Order.Where(x => x.View == false).ToList().ForEach(x => x.View = true);
    db.Product.Where(x => x.View == false).ToList().ForEach(x => x.View = true);
    db.Shipment.Where(x => x.View == false).ToList().ForEach(x => x.View = true);
    db.ContactUs.Where(x => x.View == false).ToList().ForEach(x => x.View = true);
    db.Booking.Where(x => x.View == false).ToList().ForEach(x => x.View = true);

    db.SaveChanges();
    return Redirect(Request.UrlReferrer.ToString());
}
```

Kullanıcı işlemleri için yaratılan giriş ve çıkış action içerikleri. Kullanıcı adı ve şifre ile giriş yapılması uygun görüldü.

```
[HttpGet]
0 references
public ActionResult Login()
{
    return View();
}
[HttpPost]
0 references
public ActionResult Login(User user)
{
    var usercontrol = db.User.FirstOrDefault(x => x.Username == user.Username && x.Password == user.Password);
    if (usercontrol != null)
    {
        FormsAuthentication.SetAuthCookie(usercontrol.Id.ToString(), false); //adım2

        ViewBag.mesaj = "Giriş Başarılı";
        return RedirectToAction("Booking");
    }
    else
    {
        ViewBag.mesaj = "Giriş Bilgileri Uyuşmamaktadır";
        return View();
    }

}
0 references
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    return Redirect("~/Home/Login");
}
```

Bütün depolarımızı listeyen bir facility action'ı ve bu sayfa içinde, tıklanıldığında ilgili depo'nun detayına giden facilitydetail action ve sayfası düzenlendi.

```
0 references
public ActionResult Facility()
{
    FacilityModel facility = new FacilityModel();
    facility.Warehouse = db.Warehouse.ToList();
    facility.Address = db.Address.ToList();
    return View(facility);

}
0 references
public ActionResult FacilityDetail(int id)
{
    FacilityModel model = new FacilityModel();

    var warehouse = db.Warehouse.Find(id);
    var address = db.Address.Find(id);

    model.SingleWarehouse = warehouse;
    model.SingleAddress = address;

    model.Address = db.Address.ToList();

    return View(model);
}
```

Üye kaydı ile birlikte, üyenin mail adresine bu üye için VarBal tarafından tasarlanmış bir mail yollamak için Smtp yapısı kullanıldı.

```
[HttpPost]
0 references
public ActionResult Add(User user)
{
    var emailcontrol = db.User.FirstOrDefault(m => m.Email == user.Email);

    if (emailcontrol == null)
    {
        var fromAddress = new MailAddress("[redacted]");
        var toAddress = new MailAddress(user.Email);
        var subject = "VarBal | Yeni Kayıt Bilgilendirme";
        var code = "Varbal-" + Guid.NewGuid().ToString().Substring(0, 5);
        User newuser = new User();
        newuser = user;
        newuser.Password = code;
        db.User.Add(newuser);
        db.SaveChanges();
        try
        {
            using (var smtp = new SmtpClient
            {
                Host = "smtp.gmail.com",
                Port = 587,//Port=25,
                EnableSsl = true,
                DeliveryMethod = SmtpDeliveryMethod.Network,
                UseDefaultCredentials = false,
                Credentials = new NetworkCredential(fromAddress.Address, "[redacted]"),
                Timeout = 30000
            })
            {
                using (var message = new MailMessage(fromAddress, toAddress)
                {
                    Subject = subject,
                    Body = "Merhaba " + user.Email + ", kullanıcı kaydınız başarılı bir şekilde gerçekleşti <br> Şifreniz=" + code + "<br>Giriş yapmak için <a href='https://localhost:44353/Admin/Login'>tıklayınız</a><br>Gökhan Hasbal <br><img ' src ='https://
                      blog.varbal.com.tr/wp-content/uploads/2021/03/Screenshot.png'><br>VarBal Genel Müdür",
                    IsBodyHtml = true
                })
                {
                    smtp.Send(message);
                }
            }
        }
        catch
        {
            ViewBag.mesaj = "Beklenmedik Bir Hata Gerçekleşti. Lütfen Tekrar Deneyiniz.";
            return View(model());
        }
    }
    else
    {
        ViewBag.mesaj = emailcontrol.Email + "  mail adresi sistemde mevcut";
        return View(model());
    }
    var lastuser = db.User.ToList().LastOrDefault();
    return Redirect("~/User/Edit?id=" + lastuser.Id);
}
```

Sevkiyat yaratıldığında, ürün sevkiyatın başladığı deponun stokundan azaltıldı, ilgili bilgiler sevkiyat tablosuna aktarıldı.

```
[HttpPost]
0 references
public ActionResult Create(Shipment shipment)
{
    var stoksil = shipment.Stock.Value;
    var urunid = shipment.ProductId;

    Shipment newshipment = new Shipment();
    newshipment.VehicleId = shipment.VehicleId;
    newshipment.Description = shipment.Description;
    newshipment.ProductId = shipment.ProductId;
    newshipment.UserId = shipment.UserId;
    newshipment.SenderId = shipment.SenderId;
    newshipment.ReceiverId = shipment.ReceiverId;
    newshipment.Stock = shipment.Stock;
    newshipment.SendTime = DateTime.Now;
    //newshipment.ReceiveTime = shipment.ReceiveTime;
    newshipment.Status = false;
    //newshipment.Delete = false;
    //newshipment.View = false;
    newshipment.Process = shipment.Process;
    db.Shipment.Add(newshipment);

    var editproduct = db.Product.Find(urunid);
    editproduct.Stock -= stoksil;
    db.SaveChanges();


    return View(model());
}
```

Alıcı kişi, bu sevkiyatın detay sayfasına girip ürünü kabul ettiğinde. Bu ürün, bu depoda yoksa, ürün eski depo bilgileriyle bu depoya eklendi, eğer bu depoda var ise ürün stoku güncellendi. Kontrol id'ler ve her ürünün kendine özgü VarBal barkod numarası ile yapıldı.

```
[HttpPost]
0 references
public ActionResult Detail(Shipment shipment, Product product)
{
    var editshipment = db.Shipment.Find(shipment.Id);
    editshipment.Status = true;
    editshipment.ReceiveTime = DateTime.Now;
    editshipment.Process = shipment.Process;

    var stokekle = editshipment.Stock.Value;        // böyle herhalde

    var urun = db.Product.Where(x => x.Id == editshipment.ProductId).FirstOrDefault();

    //var depo = db.Shipment.Find(shipment.ReceiverId);
    var depo = db.Product.Where(x => x.WarehouseId == editshipment.ReceiverId).FirstOrDefault();


    Product newproduct = new Product
    {
        Name = product.Name,
        Description = product.Description,
        RegisterDate = DateTime.Now,
        Barcode = product.Barcode,
        Price = product.Price,
        Tax = product.Tax,
        CategoryId = product.CategoryId,
        BrandId = product.BrandId,
        ShelfId = product.ShelfId,
        WarehouseId = product.WarehouseId,
        Status = product.Status,
        Delete = product.Delete,
        View = product.View,
        Image = product.Image,
        Stock = product.Stock

    };

    if (urun.Barcode != depo.Barcode)
    {
        db.Product.Add(newproduct);
        db.SaveChanges();
    }
    else
    {
        var urunguncelle = db.Product.Where(x => x.Barcode == urun.Barcode && x.WarehouseId == editshipment.ReceiverId).FirstOrDefault();
        urunguncelle.Stock += stokekle;
        db.SaveChanges();
    }
    return RedirectToAction("index");
}
```

Proje Ortakları	Template	Model	View	Contoller	Kodlama
Sezer VAROL	√	√	√	√	√
Gökhan HASBAL	√	√	√	√	√