



MANİSA CELAL BAYAR ÜNİVERSİTESİ

HASAN FERDİ TURGUTLU TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

NESNEYE YÖNELİK PROGRAMLAMA DERSİ DÖNEM PROJESİ

SİPARİŞ OTOMASYONU

PROJE EKİBİ:

MUHAMMED FATİH ARSLAN 212802009

MUHAMMET EMİN ÇAKIR 192802046

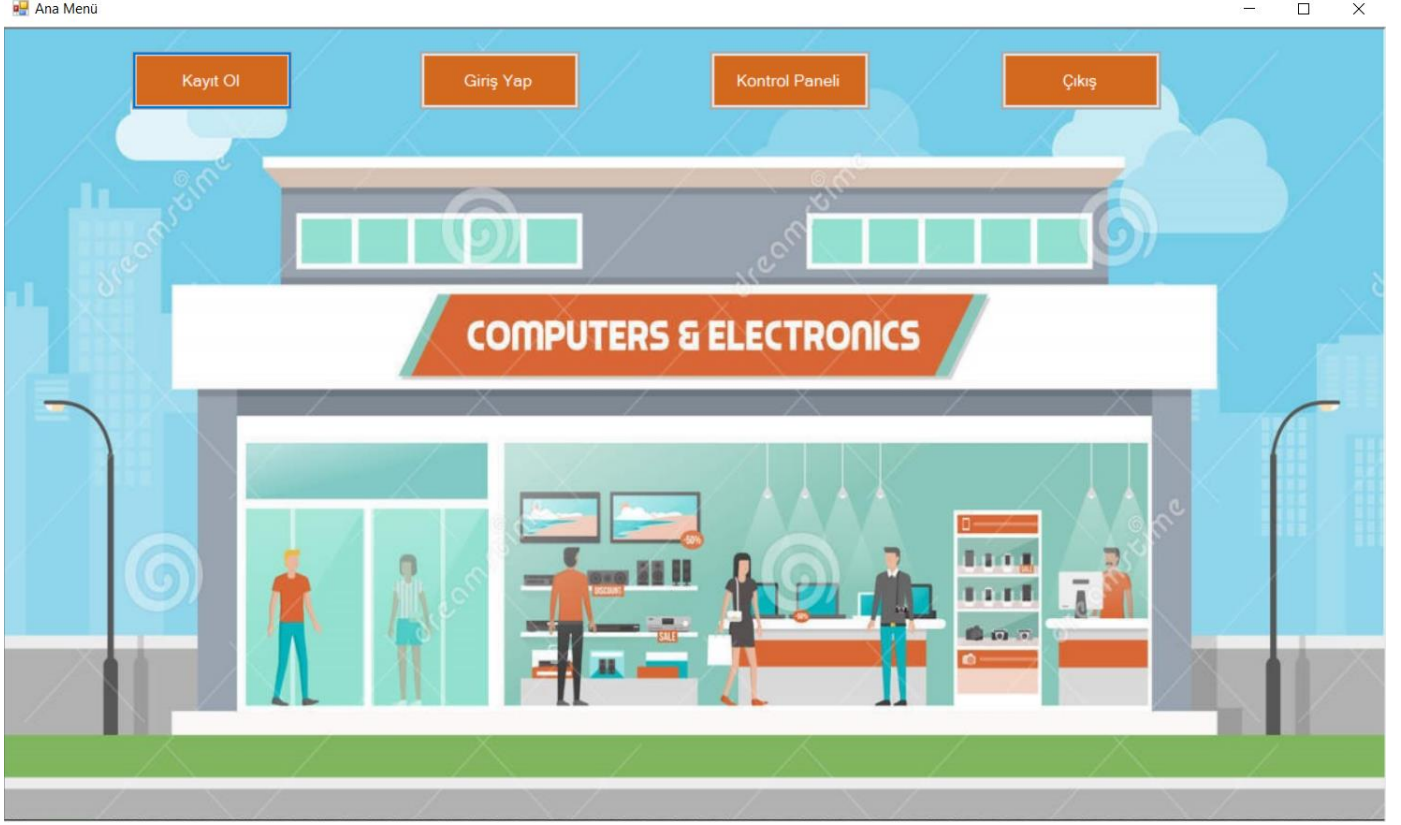
DENİZ ÖZTEPE 202803065

ÖZET

Projenin amacı, kayıtlı kullanıcılara ürün bilgilerine ulaşıp istedikleri üründen istedikleri adette sipariş verebilme ve yöneticilere ürün ekleme, çıkarma ve güncelleme imkanı sunan bir sipariş otomasyonu uygulaması geliştirmektir.

Uygulama, visual studio geliştirme ortamı kullanılarak c# diliyle kodlanmıştır.

Uygulamanın Kullanımı



İlk olarak karşımıza çıkan ana menüden kullanıcılar kayıt ol butonuna basarak kayıt yaptırabilir ve giriş yap butonuyla kayıtlı müşteri olarak giriş yaparak kullanıcı ekranına erişebilirler.

Ürünler



Masaüstü Bilgisayar

Fiyat: 15000₺

Ağırlık: 6,000kg

Stok: 10

● Ürünü Seç



Fotoğraf Makinesi

Fiyat: 8000₺

Ağırlık: 0,500kg

Stok: 10

● Ürünü Seç



Dizüstü Bilgisayar

Fiyat: 12000₺

Ağırlık: 2,000kg

Stok: 10

● Ürünü Seç



Yazıcı

Fiyat: 2000₺

Ağırlık: 4,000kg

Stok: 10

● Ürünü Seç



Cep Telefonu

Fiyat: 5000₺

Ağırlık: 0,160kg

Stok: 10

● Ürünü Seç



Xbox

Fiyat: 14000₺

Ağırlık: 4,000kg

Stok: 10

● Ürünü Seç



Tablet

Fiyat: 5000₺

Ağırlık: 0,400kg

Stok: 10

● Ürünü Seç



Playstation

Fiyat: 14000₺

Ağırlık: 4,000kg

Stok: 10

● Ürünü Seç

Tüm Ürünler

Masaüstü Bilgisayar

Seçili Ürün

Masaüstü Bilgisayar

Adet

Sepete Ekle

Sepetim

Siparişlerim

Kullanıcı ekranında ürünlerle alakalı bilgileri görerek istedikleri ürünlerden stoktaki ürün miktarını geçmeyecek kadar adette sepet oluşturup sipariş verebilirler. Verilen siparişlerin ürün adetleri veritabanındaki ürün stok bilgisinden otomatik olarak azaltılacaktır.

Sepet

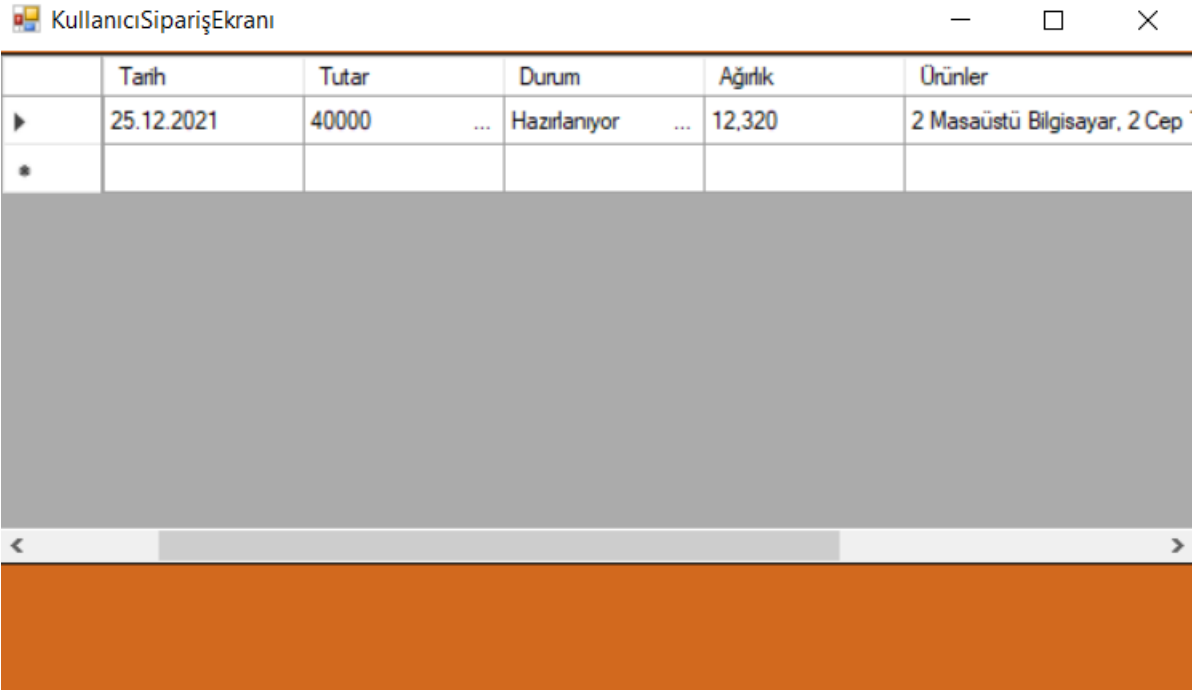
	Ürün	Fiyat	Ağırlık	Adet	Tutar
▶	Dizüstü Bilgisayar...	12000	4,000	2	24000
	Cep Telefonu ...	5000	0,320	2	10000
*					

Ürünü çıkar

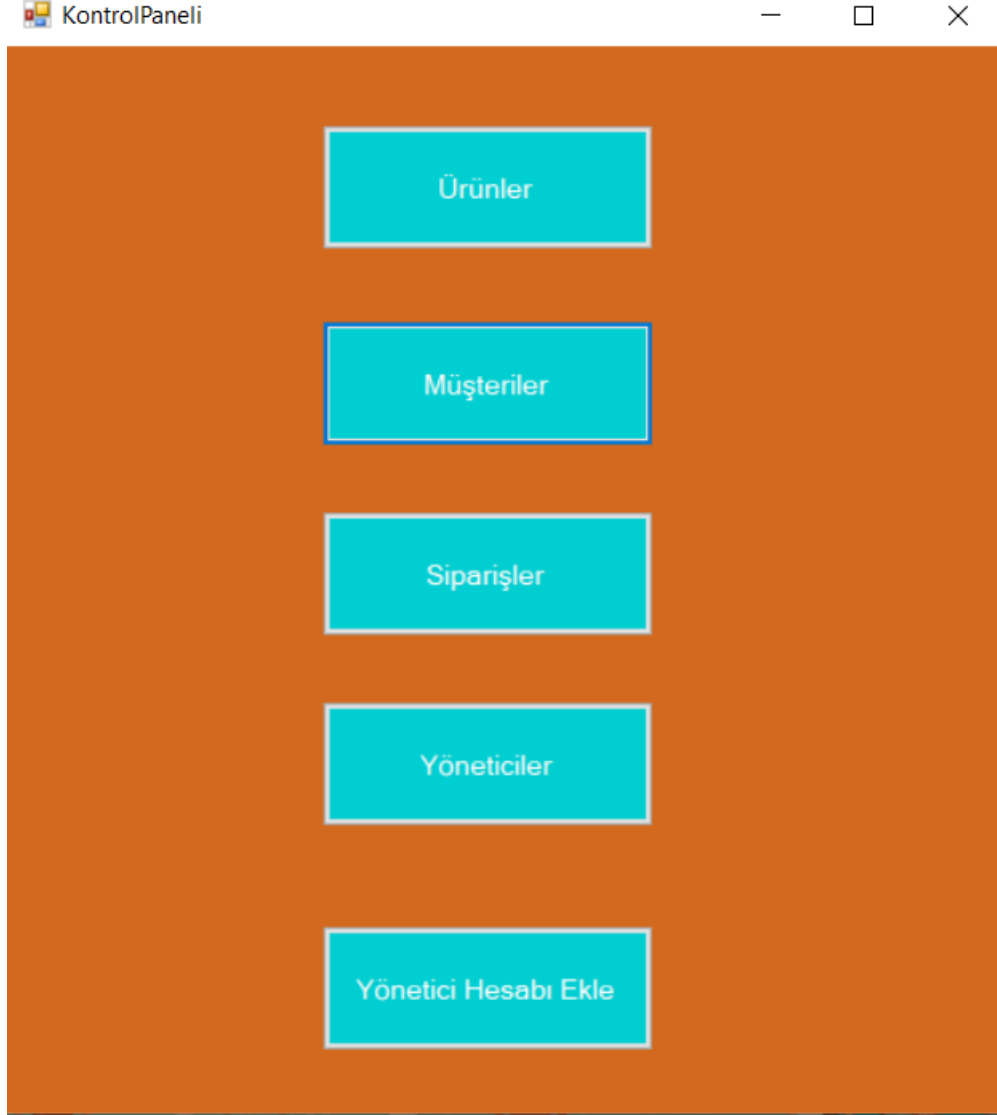
Siparişi tamamla

Toplam tutar:

34000₺



Siparişlerim butonuyla siparişlerinin durumunu ve geçmiş siparişlerini görebilirler.



Yöneticiler kontrol paneli butonuna tıklayıp yönetici girişi yaptıktan sonra kontrol paneline ulaşabilirler.Kontrol panelinde ürünlerini, kayıtlı müşterilerini, siparişleri ve yöneticileri görebilirler ve yeni yönetici hesabı oluşturabilirler.

ÜrünListesi

Ürün	Fiyat	Ağırlık	Stok
Masaüstü Bilgisayar	15000	6,000	10
Dizüstü Bilgisayar	12000	2,000	10
Cep Telefonu	5000	0,160	10
Fotoğraf Makinesi	8000	0,500	10
Tablet	5000	0,400	10
Yazıcı	2000	4,000	10
XBox	14000	4,000	10
Playstation	14000	4,000	10
*			

Ürün Ekle Ürün Çıkar Ürün Güncelle

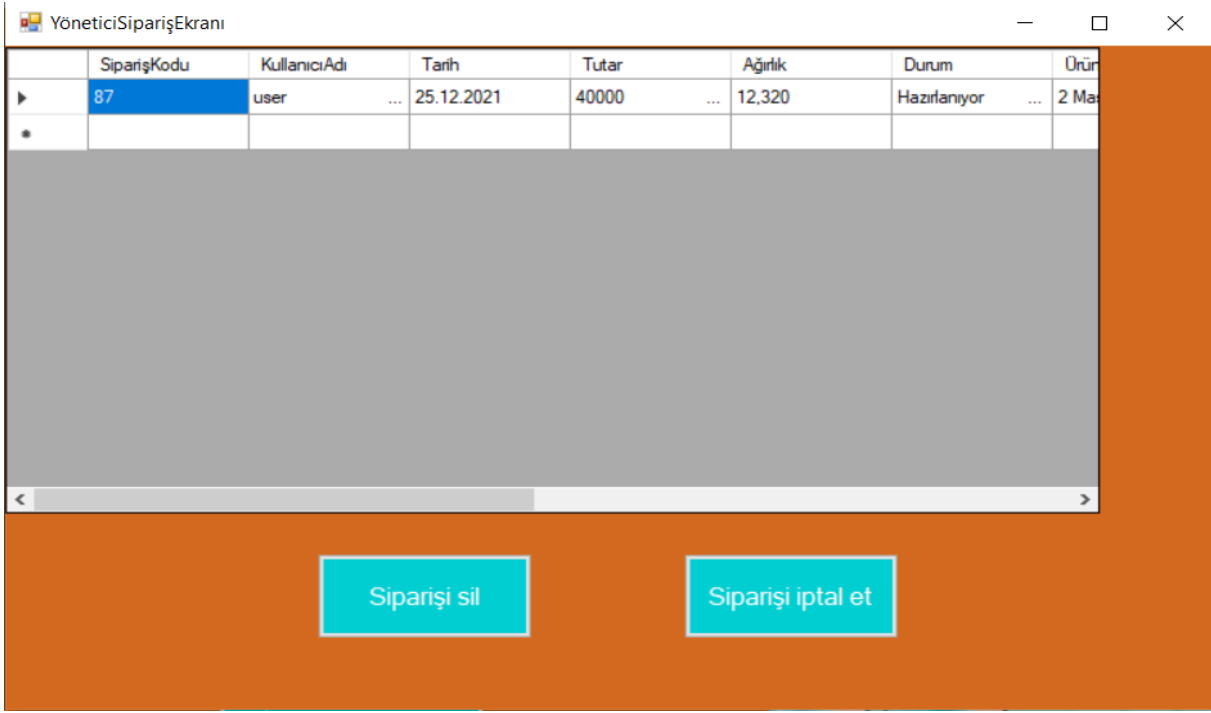
Ürünler ekranında yeni ürün ekleyebilir, ürün silebilir ya da ürün bilgilerini güncelleyebilirler.

MusteriListesi

MusteriID	Ad	Soyad	KullanıcıAdı	Şifre	Adres
29	Deniz	Öztepe	ddd	123	turgutlu
28	Emin	Çakır	eee	123	turgutlu
27	Fatih	Arslan	fff	123	Turgutlu
*					

Müşteriyi Sil

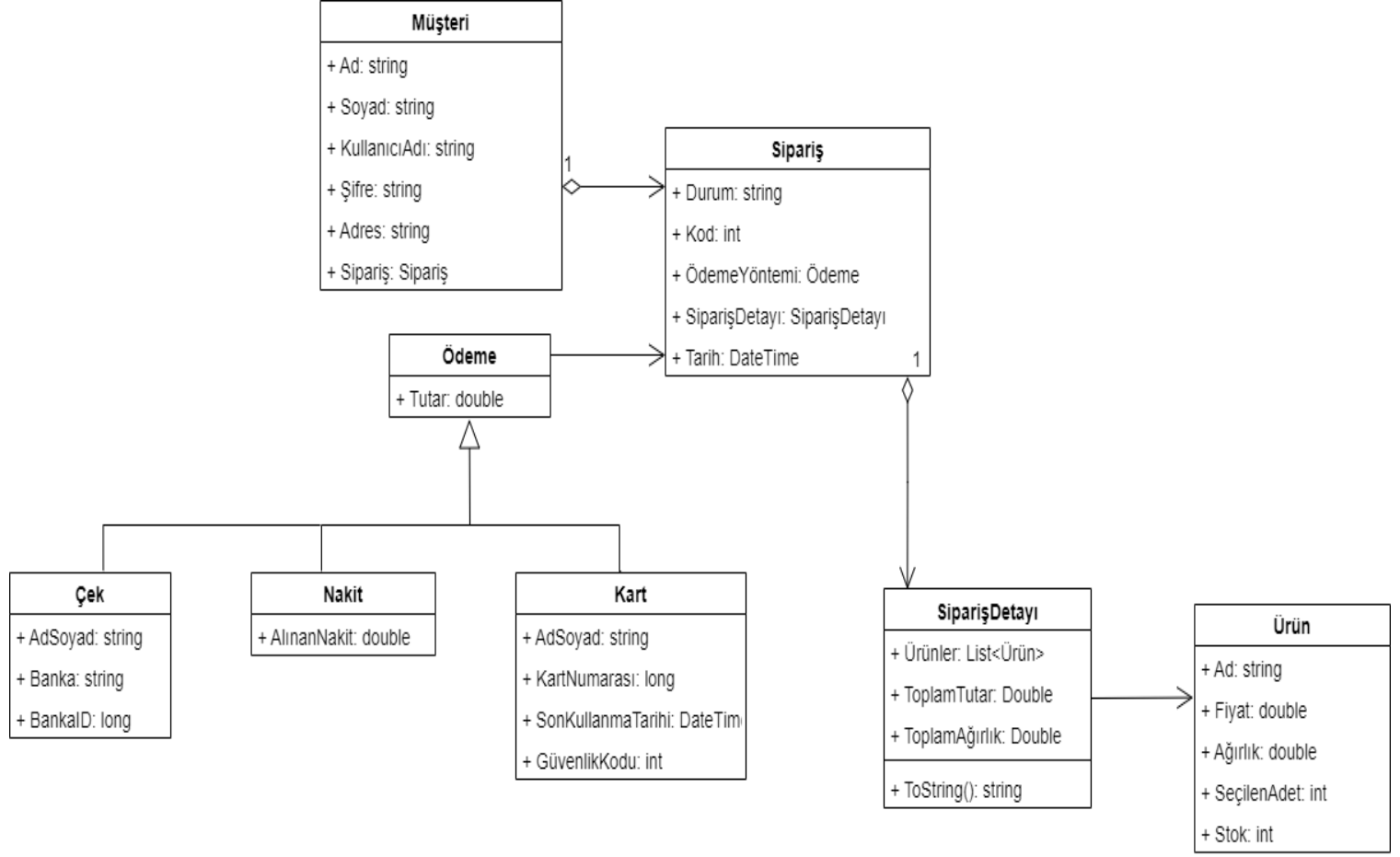
Müşteriler ekranında kayıtlı müşterilerini görebilir ve müşterilerin hesaplarını silebilirler.



Siparişler ekranında aldıkları siparişleri görebilir, iptal edebilir ve silebilirler.

İptal edilen siparişlerin ürün adetleri veritabanındaki ürün stok bilgisine otomatik olarak eklenecektir.

UML DİYAGRAMI



KODLAR

Kullanıcı Kaydı

```
private void buttonKayit_Click(object sender, EventArgs e)
{
    if(textBoxAd.Text != String.Empty && textBoxSoyad.Text != String.Empty &&
        textBoxKa.Text != String.Empty && textBoxSifre.Text != String.Empty && textBoxAdres.Text != string.Empty)
    {
        try
        {
            SqlCommand commandAdd = new SqlCommand("Insert into Musteriler (Ad, Soyad, KullanıcıAdı, Şifre, Adres) " +
                "values (@Ad, @Soyad, @KullanıcıAdı, @Şifre, @Adres)", Sqlİşlemleri.connection);
            commandAdd.Parameters.AddWithValue("@Ad", textBoxAd.Text);
            commandAdd.Parameters.AddWithValue("@Soyad", textBoxSoyad.Text);
            commandAdd.Parameters.AddWithValue("@KullanıcıAdı", textBoxKa.Text);
            commandAdd.Parameters.AddWithValue("@Şifre", textBoxSifre.Text);
            commandAdd.Parameters.AddWithValue("@Adres", textBoxAdres.Text);
            Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
            commandAdd.ExecuteNonQuery();
            this.Hide();
            MessageBox.Show("Kayıt tamamlandı");
        }
        catch (Exception)
        {
            MessageBox.Show("Bu kullanıcı adı alınmış, lütfen başka bir kullanıcı adı seçin");
        }
    }
    else
    {
        MessageBox.Show("Lütfen tüm alanları doldurun.");
    }
}
```

Kayıt formundaki tüm alanlar doldurulduktan sonra kayıt ol butonuna basıldığında veritabanındaki Müşteriler tablosuna kullanıcının girdiği bilgiler kaydediliyor. Tabloda KullanıcıAdı sütunu primary key olarak işaretlendiğinden eğer kullanıcının seçtiği kullanıcı adı tabloda zaten varsa bu işlem yapılamıyor ve exception devreye giriyor, kullanıcıya başka bir kullanıcı adı seçmesi gerektiğine dair uyarı gösteriliyor. Formdaki tüm alanlar doldurulmadan kayıt ol butonuna basıldığında ise tüm alanları doldurması gerektiğine dair uyarı gösteriliyor.

Kullanıcı Girişi

63 references

```
public partial class Kullanıcı_Girişi : Form
{
    public static Müşteri müşteri = new Müşteri();
    public static string user;
    public Kullanıcı_Girişi()
    {
        InitializeComponent();
    }
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    if(textBoxKa.Text != String.Empty && textBoxŞifre.Text != String.Empty)
    {
        SqlCommand commandlog = new SqlCommand("Select * from Musteriler where KullanıcıAdı = @KullanıcıAdı and Şifre = @Şifre", SqlI
        commandlog.Parameters.AddWithValue("@KullanıcıAdı", textBoxKa.Text);
        commandlog.Parameters.AddWithValue("@Şifre", textBoxŞifre.Text);
        SqlDataAdapter da = new SqlDataAdapter(commandlog);
        DataTable dt = new DataTable();
        da.Fill(dt);
        if (dt.Rows.Count > 0)
        {
            müşteri.KullanıcıAdı = textBoxKa.Text;
            müşteri.Şifre = textBoxŞifre.Text;
            müşteri.Ad = dt.Rows[0]["Ad"].ToString();
            müşteri.Soyad = dt.Rows[0]["Soyad"].ToString();
            müşteri.Adres = dt.Rows[0]["Adres"].ToString();
            Kullanıcı_Ekranı ke = new Kullanıcı_Ekranı();
            ke.StartPosition = FormStartPosition.CenterScreen;
            ke.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Kullanıcı adı ya da şifre yanlış");
        }
    }
    else
    {
        MessageBox.Show("Lütfen kullanıcı adı ve şifrenizi girin");
    }
}
```

```
    {
        else
        {
            MessageBox.Show("Lütfen kullanıcı adı ve şifrenizi girin");
        }
    }
}
```

Giriş formunda boş bırakılan alan olduğunda kullanıcıya tüm alanları doldurması gerektiğine dair uyarı veriliyor. Kullanıcı adı ve şifre girildikten sonra giriş butonuna basıldığında veritabanından kullanıcı adı ve şifre değerinin girilen bilgilere eşit olduğu satır alınıp bir DataTable'a ekleniyor eğer DataTable'daki

satır sayısı 0 ise bu bilgilere sahip bir kayıtlı kullanıcı olmadığı anlaşıyor ve kullanıcıya bilgilerini yanlış girdiğine dair uyarı gösteriliyor. Doğru bilgiler girildiğinde KullanıcıGirişi formuna tanımlanmış Müşteri türündeki müşteri nesnesinin propertylerine kullanıcının girdiği bilgiler atanıyor ve kullanıcı ekranı açılıyor.

Kullanıcı Ekranı

```
private void Kullanıcı_Ekranı_Load(object sender, EventArgs e)
{
    this.Height = 720;
    this.Width = 1200;
    this.StartPosition = FormStartPosition.CenterScreen;
    SqlCommand commandGet = new SqlCommand("Select Ürün from Ürünler", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    SqlDataReader reader = commandGet.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("Ürün", typeof(string));
    dt.Load(reader);
    comboBoxDigerÜrün.ValueMember = "Ürün";
    comboBoxDigerÜrün.DataSource = dt;

    SqlCommand getMasaüstü = new SqlCommand("Select * from Ürünler where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    getMasaüstü.Parameters.AddWithValue("@Ürün", "Masaüstü Bilgisayar");
    SqlDataReader reader2 = getMasaüstü.ExecuteReader();
    using(reader2)
    {
        if(reader2.Read())
        {
            MasaüstüFiyat.Text = reader2["Fiyat"].ToString() + "₺";
            MasaüstüAğırlık.Text = reader2["Ağırlık"].ToString() + "kg";
            MasaüstüStok.Text = reader2["Stok"].ToString();
        }
    }
}
```

Load eventinde yapılan işlemler:

Kullanıcı ekranı yüklenirken veritabanındaki ürünler tablosundan ürün isimleri alınıyor ve tüm ürünlerin listelendiği combobox'a ekleniyor.

Ekranda görselleri ve özellikleriyle birlikte gösterilen ürünler için ürün bilgileri de veritabanından alınıyor ve ürünün görselinin altındaki label'lara text olarak atanıyor. Ekran görüntüsünde görünen "GetMasaüstü" komutu masaüstü bilgisayar ürünü için bu işlemin örneğidir ve ekrandaki tüm ürünler için bu işlem uygulanmıştır.

```

1 reference
private void radioButtonTablet_CheckedChanged(object sender, EventArgs e)
{
    textBoxSeçiliÜrün.Text = "Tablet";
}

1 reference
private void radioButtonYazıcı_CheckedChanged(object sender, EventArgs e)
{
    textBoxSeçiliÜrün.Text = "Yazıcı";
}

1 reference
private void radioButtonMasaüstüBilgisayar_CheckedChanged(object sender, EventArgs e)
{
    textBoxSeçiliÜrün.Text = "Masaüstü Bilgisayar";
}

1 reference
private void radioButtonDizüstüBilgisayar_CheckedChanged(object sender, EventArgs e)
{
    textBoxSeçiliÜrün.Text = "Dizüstü Bilgisayar";
}

```

Ekranı gösterilen ürünlerin altındaki radiobuttonlara basıldığında seçili ürün textBox'ında gösterilmeleri sağlanıyor.

```

1 reference
private void comboBoxDiğerÜrün_SelectedIndexChanged(object sender, EventArgs e)
{
    textBoxSeçiliÜrün.Text = comboBoxDiğerÜrün.SelectedValue.ToString();
}

```

Tüm ürünlerin gösterildiği combobox'daki seçili ürünün değiştirilmesi durumunda da seçili ürün textBox'ında combobox'taki değer gösteriliyor.

Sepete ekle butonunda yapılan işlemler:

```
1 reference
private void buttonSepeteEkle_Click(object sender, EventArgs e)
{
    if(textBoxAdet.Text == String.Empty)
    {
        MessageBox.Show("Lütfen adet giriniz.");
    }
    else
    {
        Ürün ürün = new Ürün { Ad = textBoxSeçiliÜrün.Text, SeçilenAdet = int.Parse(textBoxAdet.Text) };
        SqlCommand commandGet = new SqlCommand("Select * from Ürünler where Ürün = @Ürün", Sqlİşlemleri.connection);
        commandGet.Parameters.AddWithValue("@Ürün", ürün.Ad);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        SqlDataReader reader = commandGet.ExecuteReader();
        using (reader)
        {
            if (reader.Read())
            {
                ürün.Fiyat = float.Parse(reader["Fiyat"].ToString());
                ürün.Ağırlık = float.Parse(reader["Ağırlık"].ToString());
                ürün.Stok = int.Parse(reader["Stok"].ToString());
            }
        }
    }
}
```

Eğer ürün seçilip adet değeri girilmeden butona basılmışsa adet seçilmesi gerektiğine dair uyarı gösteriliyor.

Ürün ve adet seçilmiş ise veritabanından seçilen ürünün bilgileri alınıyor ve oluşturulan ürün nesnesinin propertylerine atanıyor.

```
if (ürün.SeçilenAdet > ürün.Stok)
{
    MessageBox.Show("Stokta yeterli ürün yok, lütfen başka bir adet giriniz.");
}
```

Eğer girilen adet stoktaki ürün miktarından fazlaysa yeterli ürün bulunmadığına dair uyarı gösteriliyor.

```
try
{
    string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;
    SqlCommand command1 = new SqlCommand($"Create table {user}Sepet (Ürün nchar(30), Fiyat nchar(50), Ağırlık decimal(7,3), Adet int, Tutar decimal(15,2))");
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    command1.ExecuteNonQuery();

    SqlCommand command2 = new SqlCommand($"Insert into {user}Sepet (Ürün, Fiyat, Ağırlık, Adet, Tutar) values (@Ürün, @Fiyat, @Ağırlık, @Adet, @Tutar)");
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    command2.Parameters.AddWithValue("@Ürün", ürün.Ad);
    command2.Parameters.AddWithValue("@Fiyat", ürün.Fiyat);
    command2.Parameters.AddWithValue("@Ağırlık", ürün.Ağırlık*ürün.SeçilenAdet);
    command2.Parameters.AddWithValue("@Adet", ürün.SeçilenAdet);
    command2.Parameters.AddWithValue("@Tutar", ürün.Fiyat*ürün.SeçilenAdet);
    command2.ExecuteNonQuery();
    MessageBox.Show("Ürün eklendi.");
}
```

Stoktaki miktar girilen adet değerinden fazla olduğunda ise bir sql komutuyla veritabanında kullanıcının kullanıcı adı ve sepet (örnek: fatihsepet) isminde bir tablo oluşturuluyor ve seçilen ürün bu tabloya ekleniyor.

```

catch (Exception)
{
    string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;

    SqlCommand commandCheck = new SqlCommand($"Select * from {user}Sepet where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandCheck.Parameters.AddWithValue("@Ürün", ürün.Ad);
    SqlDataAdapter da = new SqlDataAdapter(commandCheck);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if(dt.Rows.Count > 0)
    {
        SqlCommand commandUpdate = new SqlCommand($"Update {user}Sepet set Adet = @Adet, Tutar = @Tutar, Ağırlık = @Ağırlık", Sqlİşlemleri.connection);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        string yeniTutar = (float.Parse(dt.Rows[0]["Tutar"].ToString()) + ürün.SeçilenAdet * ürün.Fiyat).ToString();
        string yeniAdet = (int.Parse(dt.Rows[0]["Adet"].ToString()) + ürün.SeçilenAdet).ToString();
        string yeniAğırlık = (float.Parse(dt.Rows[0]["Ağırlık"].ToString()) + (ürün.SeçilenAdet * ürün.Ağırlık)).ToString();
        commandUpdate.Parameters.AddWithValue("@Ürün", ürün.Ad);
        commandUpdate.Parameters.AddWithValue("@Tutar", float.Parse(yeniTutar));
        commandUpdate.Parameters.AddWithValue("@Adet", yeniAdet);
        commandUpdate.Parameters.AddWithValue("@Ağırlık", float.Parse(yeniAğırlık));
        commandUpdate.ExecuteNonQuery();
    }
}

```

```

else
{
    SqlCommand command2 = new SqlCommand($"Insert into {user}Sepet (Ürün, Fiyat, Ağırlık, Adet, Tutar) values (@Ürün, @Fiyat, @Ağırlık, @Adet, @Tutar)", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    command2.Parameters.AddWithValue("@Ürün", ürün.Ad);
    command2.Parameters.AddWithValue("@Fiyat", ürün.Fiyat);
    command2.Parameters.AddWithValue("@Ağırlık", ürün.Ağırlık * ürün.SeçilenAdet);
    command2.Parameters.AddWithValue("@Adet", ürün.SeçilenAdet);
    command2.Parameters.AddWithValue("@Tutar", ürün.Fiyat * ürün.SeçilenAdet);
    command2.ExecuteNonQuery();
}
MessageBox.Show("Ürün eklendi.");

```

Kullanıcı ikinci ürünü eklediğinde ise veritabanında bu isimde bir tablo zaten mevcut olduğundan bu işlem gerçekleştirilemiyor ve exception devreye giriyor. Bu durumda da tablo oluşturma komutu kullanılmadan sadece mevcut tabloya ekleme yapılıyor. Eğer kullanıcı aynı üründen tekrar ekleme yapmışsa tabloya yeni satır eklenmeyip o ürünün satırındaki adet, ağırlık ve tutar değerleri güncelleniyor.

Sepet Ekranı

Load eventinde yapılan işlemler:

```
public void RefreshPage()
{
    try
    {
        string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;
        double toplamTutar;
        SqlCommand commandGet = new SqlCommand($"Select * from {user}Sepet", Sqlİşlemleri.connection);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        SqlDataAdapter da = new SqlDataAdapter(commandGet);
        DataTable dt = new DataTable();
        da.Fill(dt);
        dataGridView1.DataSource = dt;
        SqlCommand commandCost = new SqlCommand($"Select Sum(Tutar) from {user}Sepet", Sqlİşlemleri.connection);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        toplamTutar = (double)commandCost.ExecuteScalar();
        textBoxToplamTutar.Text = toplamTutar.ToString() + "₺";
    }
    catch (Exception)
    {
        textBox1.Text = "Sepetiniz boş";
        textBox1.Visible = true;
        buttonÜrünÇıkar.Enabled = false;
        buttonSiparişTamamla.Enabled = false;
    }
}
```

Reference

```
private void Sepet_Load(object sender, EventArgs e)
{
    RefreshPage();
}
```

Sepet yüklenirken yapılması gereken işlemler RefreshPage isimli metotta tanımlanmış ve load eventine bu metot eklenmiştir.

RefreshPage metodunda veritabanından kullanıcının kullanıcı adı ve sepet (örnek: fatihsepet) isimli tablodaki bilgiler alınıyor ve sepet ekranındaki datagridview'e aktarılıyor. Tutar sütunundaki değerlerin toplamı toplamTutar textbox'ına yazdırılıyor.

Eğer böyle bir tablo yoksa yani kullanıcı sepet oluşturmadan sepetini görüntülemek istediye datagridview'in üzerine yerleştirilmiş ve visible özelliği

false olarak ayarlanmış olan textbox görünür olarak değiştiriliyor ve “Sepetiniz boş” yazdırılıyor.

Ürün çıkar butonu

```
1 reference
private void buttonÜrünÇıkar_Click(object sender, EventArgs e)
{
    string ürün = dataGridView1.CurrentRow.Cells["Ürün"].Value.ToString();
    string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;
    SqlCommand commandDelete = new SqlCommand($"Delete from {user}Sepet where Ürün = @Ürün", Sqlİşlemleri.connection);
    commandDelete.Parameters.AddWithValue("@Ürün", ürün);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandDelete.ExecuteNonQuery();

    RefreshPage();
}
```

Ürün çıkar butonuna basıldığında veritabanındaki tablodan seçilen satırdaki ürün isminin olduğu satır siliniyor. RefreshPage() metoduyla sayfa yenileniyor ve ürünün tablodan çıkarıldığı sepet sayfasını kapatıp tekrar açmadan görülebiliyor.

Siparişi tamamla butonu

```
1 reference
private void buttonSiparişTamamla_Click(object sender, EventArgs e)
{
    ÖdemeYöntemi öy = new ÖdemeYöntemi();
    öy.Show();
    this.Hide();
}
```

Siparişi tamamla butonuna basıldığında ödeme yöntemi seçme formu açılıyor.

Kapıda Ödeme

```
private void buttonNakit_Click(object sender, EventArgs e)
{
    string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;
    Kullanıcı_Girişi.müşteri.Sipariş = new Sipariş { SiparişDetayı = new SiparişDetayı { Ürünler = new List<Ürün>(), ToplamAğırlık =

    SqlCommand commandCost = new SqlCommand($"Select Sum(Tutar) from {user}Sepet", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.ToplamTutar = (double)commandCost.ExecuteScalar();

    SqlCommand commandWeight = new SqlCommand($"Select sum(Ağırlık) from {user}Sepet", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.ToplamAğırlık = double.Parse(commandWeight.ExecuteScalar().ToString());

    Kullanıcı_Girişi.müşteri.Sipariş.ÖdemeYöntemi = new Nakit() {AlınanNakit = Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.Toplam
    Kullanıcı_Girişi.müşteri.Sipariş.Durum = "Hazırlanıyor";
    Kullanıcı_Girişi.müşteri.Sipariş.Tarih = DateTime.Now;
```

Kapıda ödeme butonuna basıldığında kullanıcı girişinde oluşturulmuş müşteri için bir sipariş nesnesi oluşturuluyor. Sepet tablosundaki tutar ve ağırlık değerlerinin toplamaları hesaplanarak sipariş detayının toplamTutar ve toplamAğırlık propertylerine atanıyor, sipariş ödeme yöntemine nakit, sipariş tarihine günün tarihi atanıyor ve sipariş durumu “Hazırlanıyor” olarak ayarlanıyor.

```
SqlCommand commandGet1 = new SqlCommand($"Select * from {user}Sepet", Sqlİşlemleri.connection);
Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
SqlDataAdapter da = new SqlDataAdapter(commandGet1);
DataTable dt = new DataTable();
da.Fill(dt);

Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.Ürünler.Clear();
for (int i = 0; i < dt.Rows.Count; i++)
{
    string Ad = dt.Rows[i]["Ürün"].ToString();
    SqlCommand commandGet2 = new SqlCommand($"Select * from Ürünler where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandGet2.Parameters.AddWithValue("@Ürün", Ad);
    SqlDataAdapter da2 = new SqlDataAdapter(commandGet2);
    DataTable dt2 = new DataTable();
    da2.Fill(dt2);

    string Fiyat = dt.Rows[i]["Fiyat"].ToString();
    string Ağırlık = dt.Rows[i]["Ağırlık"].ToString();
    string Adet = dt.Rows[i]["Adet"].ToString();
    string Stok = dt2.Rows[0]["Stok"].ToString();
    Ürün ürün = new Ürün { Ad = Ad, Fiyat = float.Parse(Fiyat), Ağırlık = float.Parse(Ağırlık), SeçilenAdet = int.Parse(Adet), S
    Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.Ürünler.Add(ürün);
}

foreach (Ürün u in Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.Ürünler)
```

Kullanıcının sepetinden ürün ismi alınıyor ve bu isimde bir ürün nesnesi oluşturuluyor, daha sonra veritabanındaki Ürünler tablosundan o ürüne ait

özellikler alınarak ürünün gerekli propertylerine bu değerler atanıyor. Tüm propertyleri doldurulmuş olan ürün nesnesi sipariş detayının ürünler listesine ekleniyor.

```
foreach (Ürün a in Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.Ürünler)
{
    a.Stok -= a.SeçilenAdet;
    SqlCommand commandUpdate = new SqlCommand("Update Ürünler set Stok = @Stok where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandUpdate.Parameters.AddWithValue("@Ürün", a.Ad);
    commandUpdate.Parameters.AddWithValue("@Stok", a.Stok);
    commandUpdate.ExecuteNonQuery();
}
```

Sipariş detayının ürünler listesindeki her ürün için stok değeri kullanıcının satın aldığı adet kadar azaltılıyor ve veritabanındaki ürünler tablosu yeni stok değerine göre güncelleniyor.

```
SqlCommand commandAdd = new SqlCommand("Insert into Siparişler (KullanıcıAdı, Tarih, Tutar, Ağırlık, Durum, Ürünler) values (@KullanıcıAdı, @Tarih, @Tutar, @Ağırlık, @Durum, @Ürünler)", Sqlİşlemleri.connection);
SqlCommand commandAdd.Parameters.AddWithValue("@KullanıcıAdı", user);
SqlCommand commandAdd.Parameters.AddWithValue("@Tarih", Kullanıcı_Girişi.müşteri.Sipariş.Tarih.ToString("d/M/yyyy"));
SqlCommand commandAdd.Parameters.AddWithValue("@Tutar", Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.ToplamTutar);
SqlCommand commandAdd.Parameters.AddWithValue("@Ağırlık", Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.ToplamAğırlık);
SqlCommand commandAdd.Parameters.AddWithValue("@Durum", Kullanıcı_Girişi.müşteri.Sipariş.Durum);
SqlCommand commandAdd.Parameters.AddWithValue("@Ürünler", Kullanıcı_Girişi.müşteri.Sipariş.SiparişDetayı.ToString());
SqlCommand commandAdd.ExecuteNonQuery();
```

Sipariş veritabanındaki siparişler tablosuna ekleniyor.

```
public override string ToString()
{
    List<string> l = new List<string>();
    foreach(Ürün ürün in Ürünler)
    {
        ürün.Ad = ürün.Ad.Trim();
        l.Add($"{ürün.SeçilenAdet} {ürün.Ad}");
    }
    string s = String.Join(", ", l);
    return s;
}
```

Sipariş eklenirken sipariş detayının ToString() metodu kullanılarak ürünler listesindeki ürünler ve adetler alınarak string bir değer oluşturuluyor. Böylece siparişler tablosundaki ürünler kısmında sipariş detayının sadece ürün ve adet kısımları görülüyor(Örnek: 2 tablet, 2 yazıcı).

```
SqlCommand command1 = new SqlCommand("Select SiparişKodu from Siparişler", Sqlİşlemleri.connection);
Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
SqlDataAdapter adapter = new SqlDataAdapter(command1);
DataTable table = new DataTable();
adapter.Fill(table);
string k = table.Rows[table.Rows.Count - 1]["SiparişKodu"].ToString();
Kullanıcı_Girişi.müşteri.Sipariş.Kod = int.Parse(k);
```

Sipariş eklendikten sonra veritabanındaki siparişler tablosunda otomatik olarak oluşan sipariş kodu değeri alınıyor ve siparişin kod propertyisine atanıyor.

```
SqlCommand commandDelete = new SqlCommand($"Drop table {user}Sepet", Sqlİşlemleri.connection);
Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
commandDelete.ExecuteNonQuery();
this.Hide();
Kullanıcı_Girişi.müşteri.Sipariş = null;
MessageBox.Show("Siparişiniz oluşturuldu.");
```

Kullanıcının sepeti veritabanından siliniyor(kullanıcı sepetleri, sepete ürün eklendiğinde otomatik olarak oluşturulup sipariş tamamlandığında otomatik olarak silinecek şekilde kodlandı) ve kullanıcıya siparişinin alındığı bilgisi veriliyor.

ÇekleÖdeme

Ad Soyad:

Banka:

Banka ID:

Ödemeyi Tamamla

KartlaÖdeme

Ad Soyad:

Kart numarası:

Güvenik kodu (CVC):

Son kullanma tarihi: /

Ödemeyi Tamamla

Ödeme yöntemi olarak kart ya da çek seçildiğinde açılan forma gerekli bilgiler girildikten sonra siparişin ödeme yöntemi propertyesine kart yada çek değeri atandıktan sonra aynı işlemler gerçekleştiriliyor.

Kullanıcı Sipariş ekranı

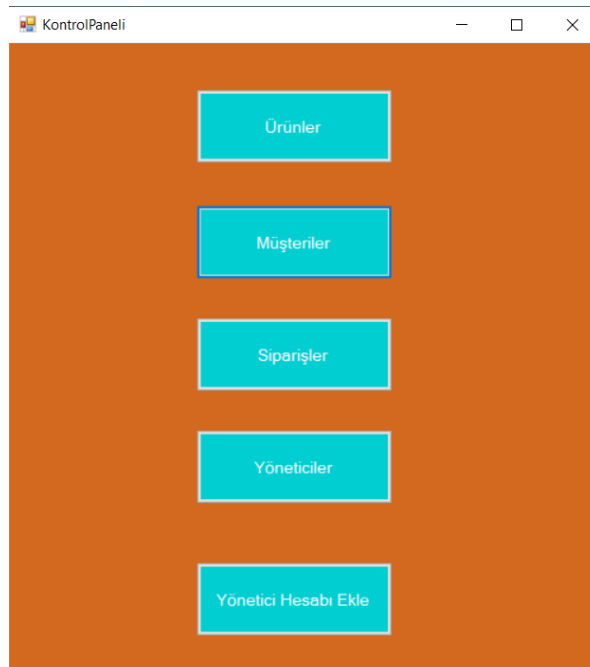
```
private void KullanıcıSiparişEkranı_Load(object sender, EventArgs e)
{
    string user = Kullanıcı_Girişi.müşteri.KullanıcıAdı;
    SqlCommand commandGet = new SqlCommand("Select SiparişKodu, Tarih, Tutar, Durum, Ağırlık, Ürünler from Siparişler where KullanıcıAdı=@KullanıcıAdı", connection);
    commandGet.Parameters.AddWithValue("@KullanıcıAdı", user);
    SqlDataAdapter da = new SqlDataAdapter(commandGet);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    dataGridView1.Columns[5].Width = 400;
    if(dt.Rows.Count == 0)
    {
        textBox1.Visible = true;
    }
}
```

Sipariş tamamlandıktan sonra kullanıcı siparişlerim butonuna bastığında veritabanındaki siparişler tablosundan, kullanıcı adı sütununda kullanıcının kullanıcı adının yazdığı satırlar alınıyor ve kullanıcı sipariş ekranındaki datagridview'e aktarılıyor. Kullanıcı sipariş oluşturmadan siparişlerini görüntülemek istediğinde datagridview'in üstüne yerleştirilmiş ve visible özelliği false olarak ayarlanmış olan textbox görünür hale getiriliyor ve sipariş listeniz boş yazdırılıyor.

Yönetici Girişi

```
private void buttonGiriş_Click(object sender, EventArgs e)
{
    if(textBoxKa.Text != String.Empty && textBoxŞifre.Text != string.Empty)
    {
        SqlCommand commandlog = new SqlCommand("Select * from Yöneticiler where KullanıcıAdı = @KullanıcıAdı and Şifre = @Şifre",
        commandlog.Parameters.AddWithValue("@KullanıcıAdı", textBoxKa.Text);
        commandlog.Parameters.AddWithValue("@Şifre", textBoxŞifre.Text);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        SqlDataAdapter da = new SqlDataAdapter(commandlog);
        DataTable dt = new DataTable();
        da.Fill(dt);
        if (dt.Rows.Count > 0)
        {
            KontrolPaneli kp = new KontrolPaneli();
            kp.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Kullanıcı adı ya da şifre yanlış");
        }
    }
    else
    {
        MessageBox.Show("Lütfen kullanıcı adı ve şifrenizi girin");
    }
}
```

Kontrol paneli butonuna basıldığında yönetici girişi formu açılıyor,formda boş alan bırakılırsa tüm alanların doldurulması gerektiğine dair uyarı veriliyor. Forma kullanıcı adı ve şifre girilip giriş yap butonuna basıldığında veritabanındaki Yöneticiler tablosundan kullanıcı adı ve şifre değerlerinin girilen değerlere eşit olduğu satırlar alınıyor ve bir DataTable'a aktarılıyor.DataTable'daki datır sayısı 0 ise yöneticiye bilgilerini yanlış girdiği uyarısı veriliyor.Bilgiler doğru girilirse kontrol paneli açılıyor.



Müşteri Listesi

```
public void RefreshPage()
{
    SqlCommand commandList = new SqlCommand("Select * from Musteriler", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    SqlDataAdapter da = new SqlDataAdapter(commandList);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}

1 reference
private void MusteriListesi_Load(object sender, EventArgs e)
{
    RefreshPage();
}
```

Load event'inde yapılan işlemler RefreshPage() isimli bir metotta tanımlanmış ve load eventine bu metot eklenmiştir.

RefreshPage() metodunda, veritabanındaki müşteriler tablosundaki bilgiler alınarak bir datagridview'e aktarılıyor ve müşteriler ekranında gösteriliyor.

Müşteri sil butonu:

```
private void buttonSil_Click(object sender, EventArgs e)
{
    string m = dataGridView1.CurrentRow.Cells["KullanıcıAdı"].Value.ToString();
    SqlCommand commandDelete = new SqlCommand("Delete from Musteriler where KullanıcıAdı = @m",Sqlİşlemleri.connection);
    commandDelete.Parameters.AddWithValue("@m",m);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandDelete.ExecuteNonQuery();
    RefreshPage();
}
```

Datagridviewde seçili satırdaki kullanıcı adı değeri alınıyor ve veritabanındaki müşteriler tablosundan kullanıcı adı sütunu bu değere eşit olan satır siliniyor.RefreshPage() metoduyla sayfa yenileniyor ve müşterinin silindiği sayfayı kapatıp açmak zorunda kalmadan görülebiliyor.

Ürün Listesi

Ürün	Fiyat	Ağırlık	Stok
Masaüstü Bilgisayar	15000	6,000	10
Dizüstü Bilgisayar	12000	2,000	10
Cep Telefonu	5000	0,160	10
Fotoğraf Makinesi	8000	0,500	10
Tablet	5000	0,400	10
Yazıcı	2000	4,000	10
XBox	14000	4,000	10
Playstation	14000	4,000	10
*			

Ürün Ekle

Ürün Çıkar

Ürün Güncelle

Load event'inde yapılan işlemler

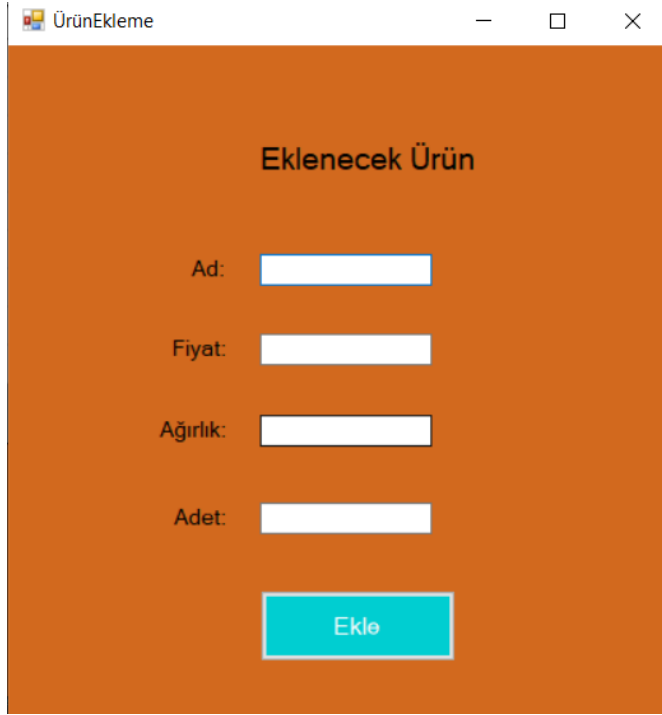
```
public void RefreshPage()
{
    SqlCommand commandList = new SqlCommand("Select * from Ürünler", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    SqlDataAdapter da = new SqlDataAdapter(commandList);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    dataGridView1.Columns[0].Width = 250;
}

1 reference
private void ÜrünListesi_Load(object sender, EventArgs e)
{
    RefreshPage();
}
```

Load event'inde yapılan işlemler RefreshPage() isimli bir metotta tanımlanmış ve load eventine bu metot eklenmiştir.

RefreshPage() metodunda veritabanındaki ürünler tablosundaki bilgiler alınıyor ve bir datagridview'e aktarılıyor.

Ürün ekleme:



```
private void buttonEkle_Click(object sender, EventArgs e)
{
    SqlCommand commandAdd = new SqlCommand("Insert into Ürünler (Ürün, Fiyat, Ağırlık, Stok) values (@Ürün, @Fiyat, @Ağırlık, @Stok) values (@Ürün, @Fiyat, @Ağırlık, @Stok)", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandAdd.Parameters.AddWithValue("@Ürün", textBoxAd.Text);
    commandAdd.Parameters.AddWithValue("@Fiyat", textBoxFiyat.Text);
    commandAdd.Parameters.AddWithValue("@Ağırlık", textBoxAğırlık.Text);
    commandAdd.Parameters.AddWithValue("@Stok", textBoxAdet.Text);
    commandAdd.ExecuteNonQuery();
    MessageBox.Show("Ürün Eklendi");
    ÜrünListesi üe = new ÜrünListesi();
    üe.RefreshPage();
}
```

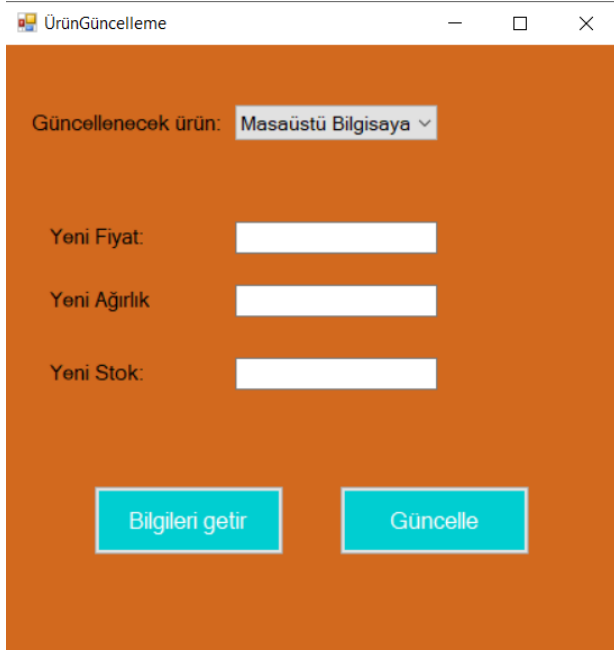
Formda gerekli bilgiler girilip ekle butonuna basıldığında veritabanındaki ürünler tablosuna bilgileri girilen ürün ekleniyor.

Ürün Silme:

```
1 reference
private void buttonÇıkar_Click(object sender, EventArgs e)
{
    string m = dataGridView1.CurrentRow.Cells["Ürün"].Value.ToString();
    SqlCommand commandDelete = new SqlCommand("Delete from Ürünler where Ürün = @m", Sqlİşlemleri.connection);
    commandDelete.Parameters.AddWithValue("@m", m);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandDelete.ExecuteNonQuery();
    RefreshPage();
}
```

Datagridview'de seçili satırdaki ürün ismi alınıyor ve veritabanındaki ürünler tablosundan bu isimdeki ürüne sahip satır siliniyor.RefreshPage() metoduyla sayfa yenileniyor ve ürünün tablodan çıkarıldığı görülüyor.

Ürün Güncelleme:



```
private void ÜrünGüncelleme_Load(object sender, EventArgs e)
{
    SqlCommand commandGet = new SqlCommand("Select Ürün from Ürünler", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    SqlDataReader reader = commandGet.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("Ürün", typeof(string));
    dt.Load(reader);
    comboBoxÜrünler.ValueMember = "Ürün";
    comboBoxÜrünler.DataSource = dt;
}
```

Güncelleme ekranı yüklenirken veritabanındaki ürün isimleri alınıyor ve ürünler combobox'ına ekleniyor

```

private void buttonBilgi_Click(object sender, EventArgs e)
{
    string isim = comboBoxÜrünler.SelectedValue.ToString();
    SqlCommand commandGet = new SqlCommand("Select * from Ürünler where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandGet.Parameters.AddWithValue("@Ürün", isim);
    SqlDataReader reader = commandGet.ExecuteReader();
    using (reader)
    {
        if (reader.Read())
        {
            textBoxFiyat.Text = reader["Fiyat"].ToString();
            textBoxAğırlık.Text = reader["Ağırlık"].ToString();
            textBoxStok.Text = reader["Stok"].ToString();
        }
    }
}

```

Bilgileri getir butonuna basıldığında ürünün mevcut bilgileri güncelleme formunda dolduruluyor.

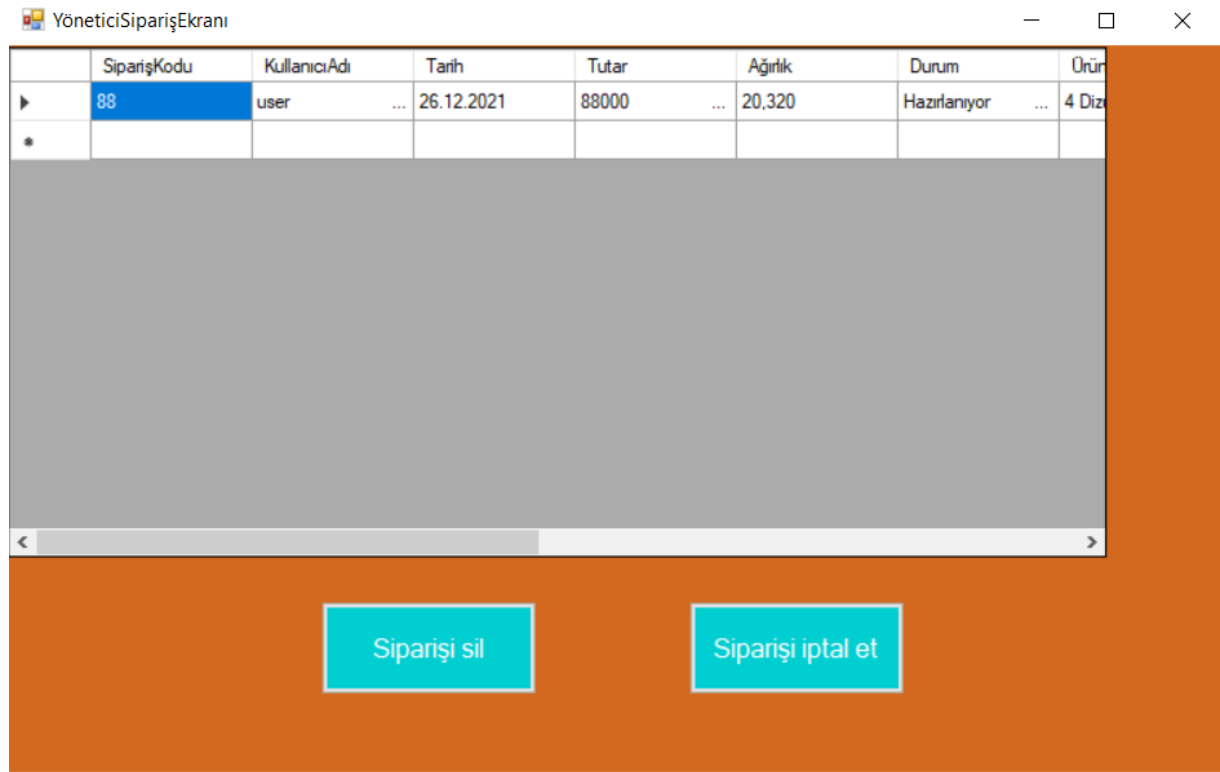
```

private void buttonGüncelle_Click(object sender, EventArgs e)
{
    string ad = comboBoxÜrünler.SelectedValue.ToString();
    string fiyat = textBoxFiyat.Text;
    double ağırlık = float.Parse(textBoxAğırlık.Text);
    string stok = textBoxStok.Text;
    SqlCommand commandUpdate = new SqlCommand("Update Ürünler set Fiyat = @Fiyat, Ağırlık = @Ağırlık, Stok = @Stok where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandUpdate.Parameters.AddWithValue("@Fiyat", fiyat);
    commandUpdate.Parameters.AddWithValue("@Ağırlık", ağırlık);
    commandUpdate.Parameters.AddWithValue("@Stok", stok);
    commandUpdate.Parameters.AddWithValue("@Ürün", ad);
    commandUpdate.ExecuteNonQuery();
    MessageBox.Show("Ürün güncellendi");
}

```

Güncelle butonuna basıldığında veritabanındaki Ürünler tablosu yeni girilen verilere göre güncelleniyor.

Siparişler



Load event'inde yapılan işlemler RefreshPage() isimli bir metotta tanımlanmış ve load eventine bu metot eklenmiştir.

```
public void RefreshPage()
{
    SqlCommand commandGet = new SqlCommand("Select * from Siparişler", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    SqlDataAdapter da = new SqlDataAdapter(commandGet);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
    dataGridView1.Columns[6].Width = 700;
}

1 reference
private void YöneticiSiparişEkranı_Load(object sender, EventArgs e)
{
    RefreshPage();
}
```

RefreshPage() metodunda veritabanındaki siparişler tablosundaki bilgiler alınıyor ve bir datagridview'e aktarılıyor.

Sipariş iptal butonu:

```
private void buttonİptal_Click(object sender, EventArgs e)
{
    string m = dataGridView1.CurrentRow.Cells["SiparişKodu"].Value.ToString();
    string ürün = "";
    string adet = "";
    string stok = "";
    string yeniStok = "";
    SqlCommand command1 = new SqlCommand("Select Ürünler from Siparişler where SiparişKodu = @m", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    command1.Parameters.AddWithValue("@m", m);
    SqlDataAdapter da = new SqlDataAdapter(command1);
    DataTable dt = new DataTable();
    da.Fill(dt);
```

Siparişi iptal et butonuna basıldığında datagridview'de seçili siparişin sipariş kodu alınıyor. Veritabanındaki siparişler tablosundan bu sipariş koduna sahip siparişin bilgileri alınıyor.

```
string ürünler = dt.Rows[0]["Ürünler"].ToString();
string[] s = ürünler.Split(',');
for (int i = 0; i < s.Length; i++)
{
    adet = String.Join("", s[i].Where(Char.IsDigit).ToArray());
    ürün = s[i].Replace(adet, " ");
    ürün = ürün.Trim();

    SqlCommand command2 = new SqlCommand("Select Stok from Ürünler where Ürün = @Ürün", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    command2.Parameters.AddWithValue("@Ürün", ürün);
    SqlDataReader reader = command2.ExecuteReader();
    using (reader)
    {
        if (reader.Read())
        {
            stok = reader["Stok"].ToString();
            yeniStok = (int.Parse(adet) + int.Parse(stok)).ToString();
            SqlCommand command3 = new SqlCommand("Update Ürünler set Stok = @Stok where Ürün = @Ürün", Sqlİşlemleri.connection);
            Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
            command3.Parameters.AddWithValue("@Stok", yeniStok);
            command3.Parameters.AddWithValue("@Ürün", ürün);
            reader.Close();
            command3.ExecuteNonQuery();
        }
    }
}
```

Sipariş tablosundaki ürünler sütunundaki değer(örnek: 2 tablet, 2 yazıcı) ürünler isimli bir string değişkene atanıyor. Bu string, split metoduyla bölünerek bir string dizesi elde ediliyor (örnek: {"2 tablet", "2 yazıcı"}). Bu dizedeki her eleman için adet ve ürün değeri ayrılarak adet ve ürün isimli değişkenlere atanıyor ve veritabanındaki Ürünler tablosundaki stok bilgileri siparişteki adede göre güncelleniyor.


```
SqlCommand command4 = new SqlCommand("Update Siparişler set Durum = @Durum where SiparişKodu = @Kod", Sqlİşlemleri.connection);
Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
command4.Parameters.AddWithValue("@Durum", "İptal edildi");
command4.Parameters.AddWithValue("@Kod", m);
command4.ExecuteNonQuery();
RefreshPage();
```

Veritabanındaki siparişler tablosunda seçili siparişin durum bilgisi “İptal edildi” olarak değiştiriliyor.RefreshPage() metoduyla sayfa yenileniyor ve siparişin iptal edildiği görülebiliyor.

Siparişi sil butonu:

```
1 reference
private void buttonSil_Click(object sender, EventArgs e)
{
    string m = dataGridView1.CurrentRow.Cells["SiparişKodu"].Value.ToString();
    string durum = "İptal edildi";
    SqlCommand commandGet = new SqlCommand("Select Durum from Siparişler where SiparişKodu = @SiparişKodu and Durum = @Durum", Sqlİşlemleri.connection);
    Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
    commandGet.Parameters.AddWithValue("@Durum", durum);
    commandGet.Parameters.AddWithValue("@SiparişKodu", m);
    SqlDataAdapter da = new SqlDataAdapter(commandGet);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if(dt.Rows.Count > 0)
    {
        SqlCommand commandDelete = new SqlCommand("Delete from Siparişler where SiparişKodu = @m and Durum = @Durum", Sqlİşlemleri.connection);
        commandDelete.Parameters.AddWithValue("@m", m);
        commandDelete.Parameters.AddWithValue("@Durum", durum);
        Sqlİşlemleri.CheckConnection(Sqlİşlemleri.connection);
        commandDelete.ExecuteNonQuery();
        RefreshPage();
    }
    else
    {
        MessageBox.Show("Siparişi silmek için iptal etmelisiniz");
    }
}
```

Datagridview’de seçili satırdaki sipariş koduna sahip ve durum değeri “İptal edildi” olan sipariş veritabanındaki siparişler tablosundan alınıyor. Böyle bir sipariş yoksa “Siparişi silmek için iptal etmelisiniz uyarısı veriliyor”. Aksi durumda sipariş veritabanından siliniyor.RefreshPage() metoduyla sayfa yenileniyor ve siparişin silindiği görülebiliyor.

KAYNAKLAR

<https://social.msdn.microsoft.com/Forums/tr-TR/home>

<https://stackoverflow.com/>

<https://www.c-sharpcorner.com/>