OSMANİYE KORKUT ATA ÜNİVERSİTESİ
KADİRLİ MESLEK YÜKSEKOKULU
İNTERNET VE AĞ TEKNOLOJİLERİ BÖLÜMÜ
2020-2021 YAZ DÖNEMİ STAJ ÖDEVİ

27.08.2021

Öğrenci Adı-Soyadı	: Mustafa KALENDER
Öğrenci Numarası	: 2020325007
Bölümü	: İnternet ve Ağ Teknolojileri
Staj Dönemi	: 2020-2021 Yaz Dönemi
Staj Konusu	: Görsel Programlama
Danışman	: Öğr. Gör. Tamer ATCIOĞLU

İÇİNDEKİLER

GİRİŞ

Bu çalışmada 2020-2021 Yaz döneminde Staj ödevi konusu olan Görsel Programlama işlenecektir. Konu Microsoft Visual Studio 2019, Microsoft Access Database Engine 2016, Microsoft 365 sürümleri ile 2021 yılındaki uygulamalar ve yönetmelikler esas alınarak anlatılmıştır. Görsel programlama konusunun önemi; yazılımların bilgisayar kullanıcıları tarafından kullanılabilmesi için görsel programlamaya ihtiyaç duyulmasından kaynaklanır. Bunun için de çeşitli görsel ögeler Microsoft Visual Studio gibi arayüz yazılımları tarafından metinsel programlama yapılarına çevrilir. Görsel programlama dilleri aracılığıyla yapılan bu iş, temelde programların daha anlaşılır hale gelmesini amaçlar.

Çalışmanın amacı; 2021-2022 Eğitim-Öğretim yılı, Güz ve Bahar yarıyıllarının ders programlarında işlenmesi amaçlanan alan dersleri konularından Görsel Programlama konusu için araştırma ve proje örneği tasarlayarak konuyu öğrenmektir. Staj ödevi için seçilen proje ise Microsoft Visual Studio arayüzü ile C# programlama dili ve Microsoft Access Veri tabanı kullanarak küçük çaplı işletmelerin stoktaki ürünlerini ve bu ürünlerin satışlarını bilgisayar ortamında kullanımı kolay ve sade bir arayüz ile kolaylıkla takip edebilmesini hedefleyen bir stok ve satış takip programı kodlamaktır.

Giriş kısmından sonraki bölümlerde Staj konusu olan Görsel Programlama ile C# programlama dili, Microsoft Access Database ve Microsoft Visual Studio arayüzü hakkında temel bilgiler, Microsoft Visual Studio 2019 yazılımının C# dil eklentileriyle kurulumu, projesi geliştirilecek Stok Takip programının amacı, programın görsel tasarımı, programın yapması hedeflenen işlevleri kodlama adımları, programın oluşturulan kurulum dosyası ile son kullanıcıya kurulum aşamaları ve programın kullanımı esnasında karşılaşılabilecek hatalardan bahsedilecektir. Sonuç kısmında ise çalışmanın kazandırdığı bilgiler yer alacaktır.

1.GÖRSEL PROGRAMLAMA

1.1.Görsel Programlama Nedir?

Görsel programlama, insanların illüstrasyon ve görsel objeler kullanarak süreçleri tanımlamasına izin veren bir programlama dili türüdür. Tipik bir metin tabanlı programlama dili, programcının bir bilgisayar gibi düşünmesini sağlarken, görsel bir programlama dili, programcının süreci insanlar için anlamlı olan terimlerle ve görsel objelerle tanımlamasını sağlar. Görsel programlama ile, kullanılan programlama dilinin öğeleri grafiksel yapılar şeklinde sağlanır. Bu yapıların görünümüne ve etiketlerine dayanarak bir geliştirici, programlamada hangi görevlerin hangi bileşenler kullanılarak çözülebileceğini anlayabilir.

Görsel programlama ile metin tabanlı programlama arasındaki farkın ne kadar büyük olduğu görsel programlama aracına bağlıdır. Örnek verecek olursak: Görsel bir programlama aracıyla yapılacaklar listesi oluşturmak için programcı, uygulamanın akışını çizer. Ortaya çıkan akış şeması, ekranları, kullanıcı etkileşimlerini ve her aşamada verilere ne olduğunu açıklar. Araç daha sonra bunu yazılıma dönüştürür.

Görsel Programlama öğeleri, yapboz parçaları gibi birbirine uyacak şekilde tasarlanmıştır. Öğeler mantıksal olarak birbirine uymazsa, bu düzenleyici tarafından tespit edilir ve renk kullanılarak belirtilir. Her şey, geliştiricinin kullanıcıya vermek istediği deneyimi oluşturmak için bilgisayarın atması gereken kesin adımları tanımlaması ile ilgilidir.

1.2.Görsel Programlama Dilleri Nelerdir?

1. Visual Studio
    - 1.1. Visual Basic
    - 1.2. Visual C++
    - 1.3. Visual C#
2. Rad Studio
    - 2.1. Delphi
    - 2.2. C++ Builder
3. Dev Pascal
4. Dev C++
5. Lazarus
6. Mono Develop
7. Android Studio
8. Eclipse

2.VERİ TABANI VE MİCROSOFT ACCESS DATABASE (ACCESS VERİ TABANI)

2.1.Veri Tabanı

Veri tabanı, bilgi toplamak, depolamak ve düzenlemek için bir araçtır. Veri tabanları insanlar, ürünler, siparişler veya başka herhangi bir şey hakkında bilgi depolamaya yarar.

Geniş kapsamlı veri tabanları ise veri tabanı yönetim yazılımlarıyla (DBMS, Database Management System) birlikte kullanılır. Makine diliyle kullanıcı arasında çalışan ve tüm verilerin yönetimini üstlenen veri tabanı programları, kullanım kolaylığı ve güvenliği amaçlar.

Veri tabanı yönetimi ile veri kaybının önüne geçen programlar, kamusal ve özel hizmetlerin sürekli ve hızlı şekilde sunulmasını sağlar.

2.2.Microsoft Access Database (Access Veri Tabanı)

Microsoft Access Veri tabanı, Windows işletim sisteminde çalışmak üzere hazırlanan veri tabanı programıdır. Bu veri tabanında bulunan nesnelerin birçoğu sihirbazlar yardımıyla kolayca hazırlanabilir. Access veri tabanı, başka bir kaynaktan gelen verileri veya kodu kullanmak üzere özel olarak tasarlanmadıkça, tablolarını formlar, raporlar, makrolar ve modüller gibi diğer nesnelerle birlikte tek bir dosyada depolar.

Access 2007 biçiminde (Ayrıca Access, 2016, Access 2013 ve Access 2010 tarafından da kullanılır) oluşturulan veri tabanları .accdb dosya uzantısına ve önceki Access biçimlerinde oluşturulan veri tabanları .mdb dosya uzantısına sahiptir.

Veri tabanı yönetim sistemleri arasına Access çok sonradan girmiş olmasına rağmen bu alanda önemli ölçüde başarı sağlayarak küçük ölçekli veri tabanları için çok kullanılan bir paket haline gelmiştir. Bunda, Access'in yazılım araçlarının yüksek kullanıcı kolaylığına sahip olmasının etkisi büyüktür.

3.MİCROSOFT VİSUAL STUDİO VE C#

3.1.C#

Microsoft tarafından geliştirilen C# (C Sharp), .NET ortamında kullanılmak üzere sunucu ve gömülü sistemleri için tasarlanan modern bir programlama dilidir.

C# (C Sharp) Microsoft tarafından geliştirilmiş olsa da ECMA ve ISO standartları altına alınmıştır. Dilin tasarlanmasına Pascal, Delphi derleyicileri ve J++ programlama dilinin tasarımcısı Anders Hejlsberg öncülük etmiştir.

Zamanla gelişen ve bazı alanlarda Java programlama dilini örnek alan C# gerek masaüstü, web uygulamaları gibi birçok ortamda kullanılmaktadır.

3.2.Microsoft Visual Studio

Microsoft Visual Studio, Microsoft tarafından geliştirilen ve bilgisayar programları, web siteleri, web uygulamaları ve mobil uygulamalar gibi farklı yazılım geliştirme türleri için kullanılan bir geliştirme yazılımıdır.

Yazılım geliştirme sürecini kolaylaştırmak için tamamlama araçları, derleyiciler ve diğer özellikleri içerir. Visual Studio, değişik programlama dillerini destekler, bu da kod editörü ve hata ayıklayıcısının neredeyse tüm programlama dillerini desteklemesini sağlamaktadır. Dahili diller C/C++ (Görsel yoluyla C++), VB.NET (Visual Basic .NET üzerinden), C# (Visual C# ile), ve F# (Visual Studio 2010 itibarıyla) içermektedir.

4.PROJE BÖLÜMLERİ

4.1.Projenin Amacı

Proje; küçük çaplı işletmelerin stoktaki ürünlerini ve ürün satışlarını kayıt altına alarak bilgisayar ortamında kolay ve sade bir arayüz programı ile kolaylıkla takip etmesini hedefleyen bir stok ve satış takip programı kodlamaktır. Kodlanacak programın adı Stok Takip 'tir. Programda son kullanıcının erişebileceği 7 adet pencere olacaktır. Bunlar; stoklara yetkisiz işlemi engellemek ve programın güvenliği için açılışta yetkili kişinin giriş bilgilerini kontrol edecek kullanıcı adı ve şifre kontrol penceresi, işletmenin stok ürünlerini kategorilere ve markalara ayırması için kategori ve marka ekleme-silme pencereleri, ürünlerin stok durumları ve barkod bilgilerinin ekleneceği ürün ekleme penceresi, stoktaki ürünlerin bilgileri ve stok adetlerinin gözlemleneceği ve değiştirilebileceği ürün listeleme penceresi, stoktaki ürünlerin sepete eklenip satışının yapılacağı satış penceresi ve satışı yapılan ürünlerin bilgilerini takip edilebileceği satış listeleme penceresidir.

4.2.Projenin Kodlanması İçin Gerekli Olan Yazılımlar

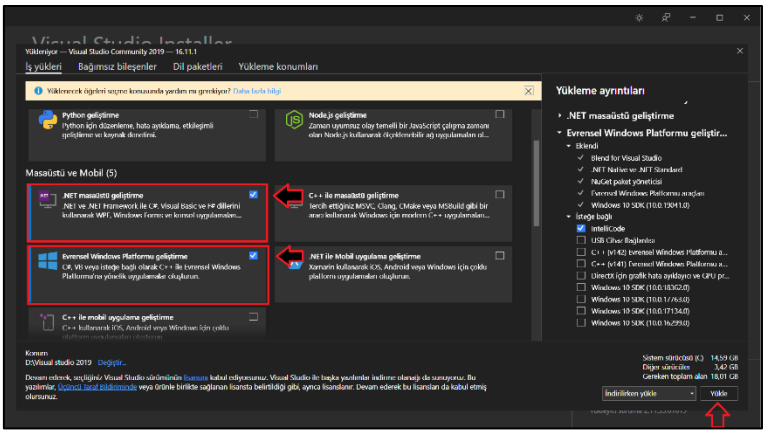
Proje kodlamasını yapabilmek için; 2021 Ağustos güncellemesi yapılmış 64 bit Windows 10 işletim sistemi, Access veri tabanı oluşturmak için Microsoft 365 uygulamaları, .NET Framework 4.0 veya üzeri sürümleri, Visual Studio 2019 Community yazılımlarının yüklü olması gerekmektedir.

4.3.Visual Studio 2019 Community Programının C# Eklentileriyle Kurulumu

Visual Studio 2019 Community kurulum dosyasını indirmek için https://visualstudio.microsoft.com/tr/ adresine giriş yapılır. Açılan sayfada, "Visual Studio'yu indirin" kutucuğu tıklanıp açılan menüden "Community 2019" bağlantısına tıklanır.

İndirme işlemi tamamlandıktan sonra tarayıcının indirilenler sekmesinden veya dosyanın indirildiği klasöre girilerek "vs_Community.exe" dosyası çalıştırılır.

Ardından açılan ekranda "Devam" kutucuğuna tıklanır.



Resim 1

Visual Studio Installer kurulum işlemi tamamlandıktan sonra ".NET masaüstü geliştirme" ve "Evrensel Windows Platformu Geliştirme" seçenekleri işaretlenip "Yükle" kutucuğuna tıklanarak kodlama için gerekli yazılımları ve eklentileri yükleme işlemi başlatılır. *Resim 1*

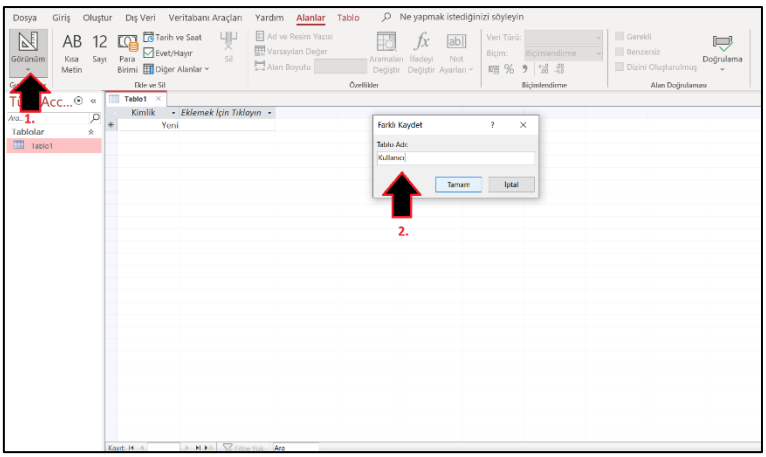
Programın kurulumu tamamladığında, X (çarpı) butonuna tıklanarak kurulum aşaması bitirilir.

4.4.Projenin Tasarım Ve Kodlanma Adımları

4.4.1.Access Veri Tabanı Oluşturma

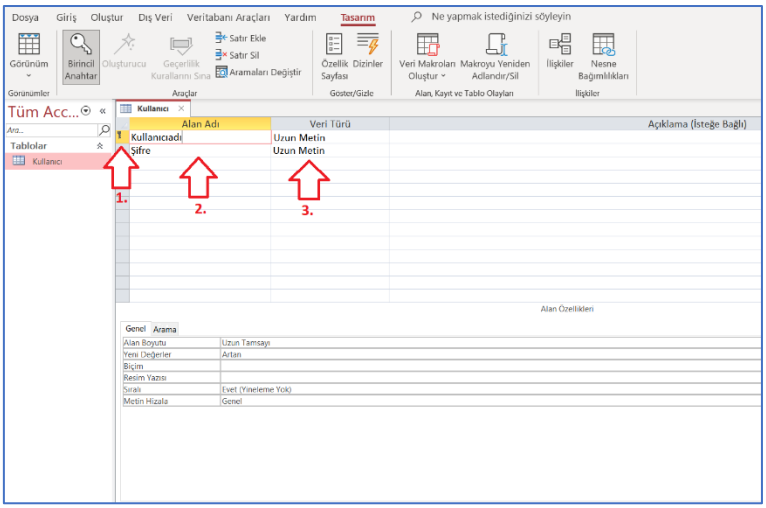
Not: Bu aşama için bilgisayarda Microsoft 365 uygulamaları kurulu olmalıdır.

Access programı açıldıktan sonra ana pencereden veya soldaki "Yeni" sekmesinden "Boş veri tabanı" tıklanır. Açılan pencerede klasör simgesi tıklanıp dosyanın oluşturulacağı klasör "Masaüstü" seçilir. "Dosya adı" alanına ise projede kullanılacak veri tabanı dosyasının adı olan "StokVerisi" yazılır ve "Oluştur" butonuna tıklanarak veri tabanı dosyası oluşturulur.



Resim 2

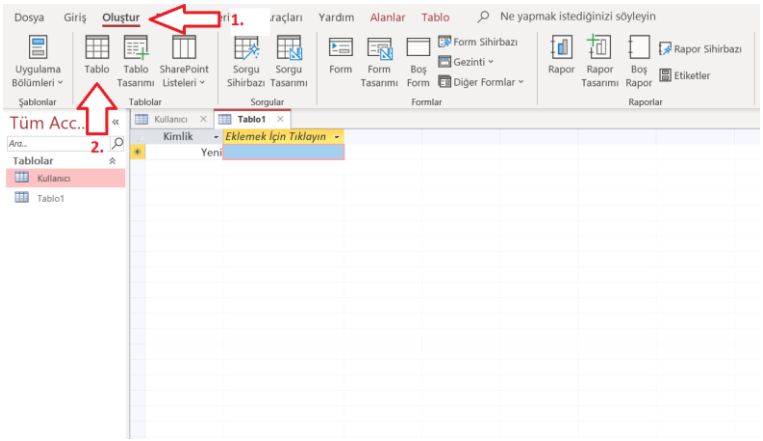
Oluşturulan boş veri tabanı penceresindeki ilk tabloyu adlandırmak ve tablo verilerini girmek için "Alanlar" menüsü seçilerek "Görünüm" butonuna tıklanır. Açılan "Farklı Kaydet" penceresinde "Tablo Adı" alanına projenin giriş penceresinde kontrol edilecek Kullanıcı adı ve Şifre verilerinin kaydedileceği tablo adı olan "Kullanıcı" yazılıp "Tamam" butonuna tıklanarak tablo adı değiştirilir. *Resim 2*



Resim 3

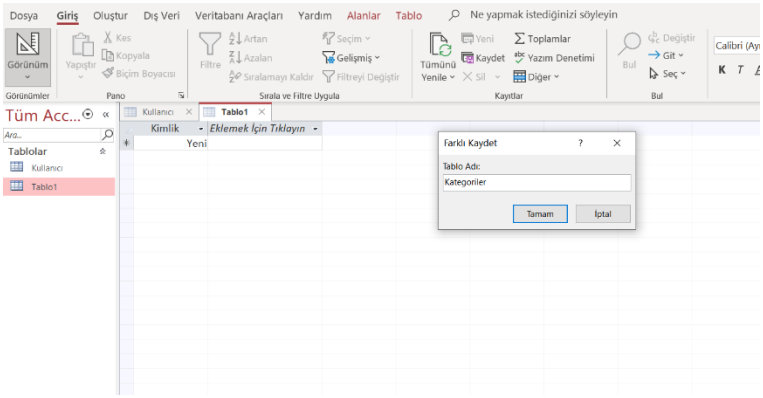
Soldaki "Tablolar" sekmesinden Kullanıcı tablosu tıklanarak seçilir. Kullanıcı tablosunda, "Alan Adı" altında birinci sütun sağ tuşla tıklanarak hücrenin "Birincil Anahtar" özelliği kapatılır. Bu işlem daha sonra oluşturulacak tabloların hepsinde uygulanmalıdır. *Resim 3*

"Alan Adı" altındaki birinci hücre sol tuşla tıklanıp ilk alanın adı, "Kullanıcıadı", ardından ikinci hücre tıklanarak ikinci alanın adı "Şifre" olarak adlandırılır ve veri türleri "Uzun Metin" olarak seçilir. Daha sonra Access uygulamasında "Tasarım" menüsü seçilip "Görünüm" butonuna tıklanarak Kullanıcı tablo penceresinden, Kullanıcıadı sütunu birinci hücresi seçilerek programa girişte kullanılacak kullanıcı adı olan "admin", Şifre sütunu birinci hücresi seçilerek girişte kullanılacak şifre olan "password" yazılıp "kaydet" butonuna tıklanarak tablo kaydedilir. *Resim 3*



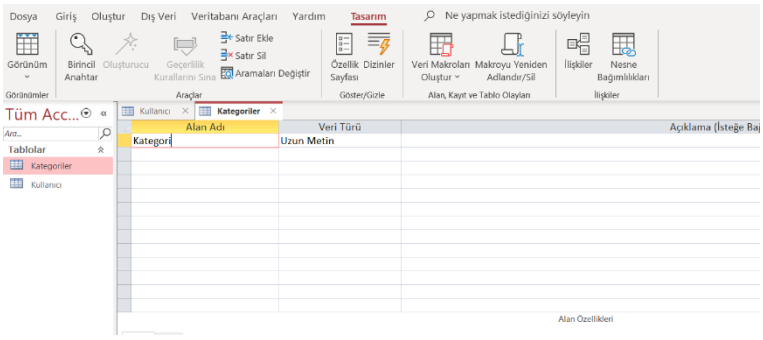
Resim 4

Daha sonra yeni bir tablo oluşturmak için program menüsünden "Oluştur" menüsü seçilip "Tablo" butonuna tıklanarak yeni tablo oluşturulur. *Resim 4*



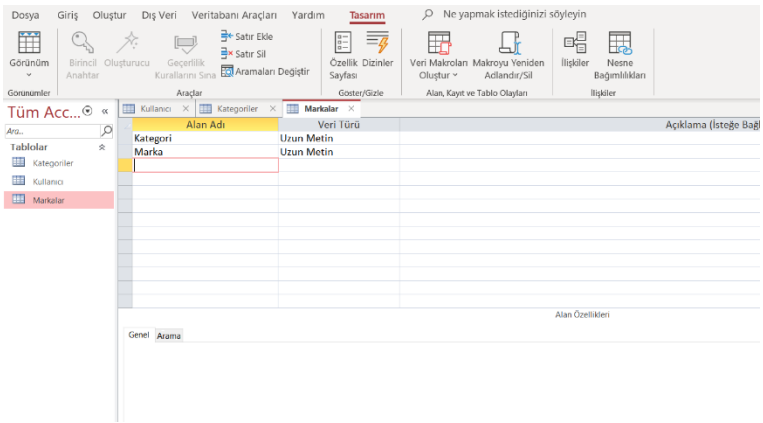
Resim 5

Soldaki "Tablolar" sekmesinden yeni oluşturulan tablo seçilerek *Resim 2*'teki adımlar tekrarlanarak tablo adı "Kategoriler" olarak değiştirilir. *Resim 5*



Resim 7

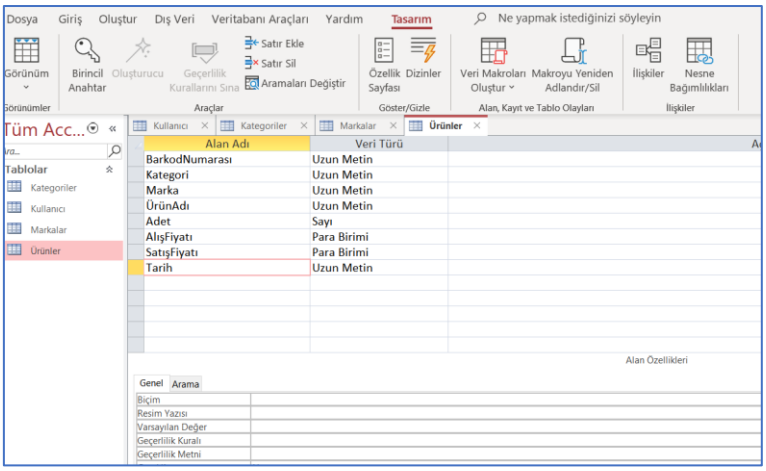
Kategoriler tablosu içindeki ilk hücreye "Kategori" yazılıp veri türü "Uzun Metin" olarak değiştirilerek alan adı kaydedilir. Daha sonra bu alan adı altında, programda kaydedilecek ürünlerin kategorileri kodlarla tanımlanarak depolanacaktır. *Resim 7*



Resim 8

Resim 4 'deki adımlar tekrarlanarak programda eklenecek ürün markalarının depolanacağı tablo oluşturulup *Resim 2* 'deki adımlar tekrarlanarak tablo adı "Markalar" olarak değiştirilir.

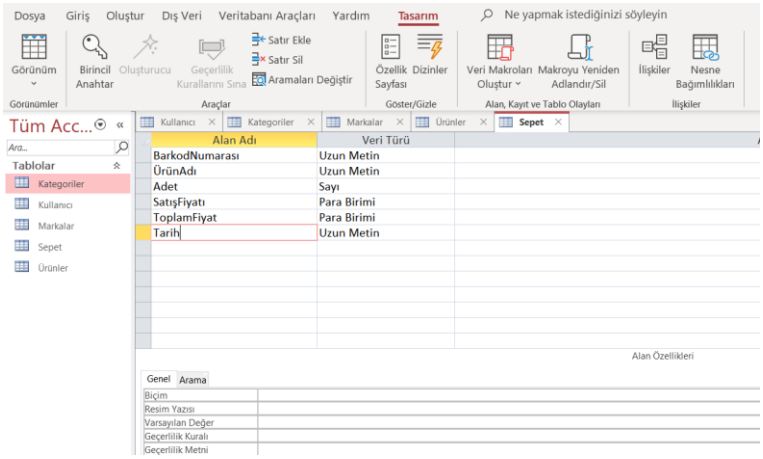
Markalar tablosuna "Kategori" ve "Marka" olmak üzere iki alan adı eklenerek veri türleri "Uzun Metin" olarak değiştirilir. *Resim 8*



Resim 9

Resim 4 'deki adımlar tekrarlanarak programda eklenecek ürün barkod bilgilerinin depolanacağı tablo oluşturulup ardından *Resim 2* 'deki adımlar tekrarlanarak tablo adı "Ürünler" olarak değiştirilir.

Ürünler tablosuna sırasıyla "BarkodNumarası"(veri türü "Uzun Metin") , "Kategori" (veri türü "Uzun Metin") , "Marka" (veri türü "Uzun Metin") , "ÜrünAdı" (veri türü "Uzun Metin") , "Adet" (veri türü "Sayı") , "AlışFiyatı" (veri türü "Para Birimi") , "SatışFiyatı" (veri türü "Para Birimi") , "Tarih" (veri türü "Uzun Metin") olmak üzere sekiz alan adı eklenir. Bu alanlar ürünlerin barkod bilgilerinin saklanacağı kısımlardır. *Resim 9*

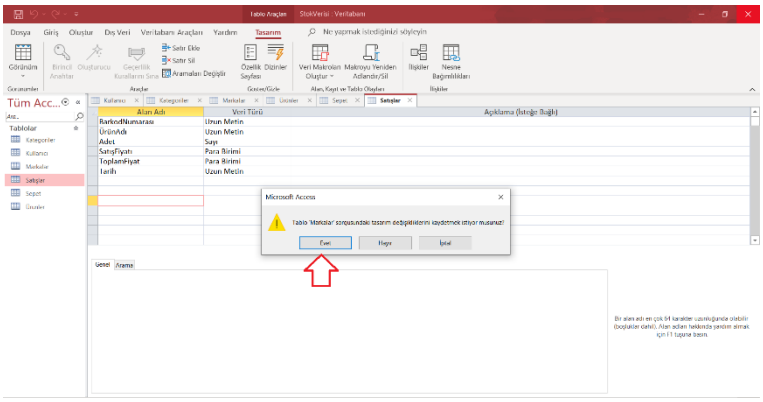


Resim 10

Resim 4 'deki adımlar tekrarlanarak programda satılacak ürünlerin sepet eklenen ürün bilgilerinin tutulacağı tablo oluşturulup ardından *Resim 2* 'deki adımlar tekrarlanarak tablo adı "Sepet" olarak değiştirilir.

Sepet tablosuna sırasıyla "BarkodNumarası"(veri türü "Uzun Metin") , "ÜrünAdı" (veri türü "Uzun Metin") , "Adet" (veri türü "Sayı") , "SatışFiyatı" (veri türü "Para Birimi") ,

“ToplamFiyat” (veri türü “Para Birimi”) , “Tarih” (veri türü “Uzun Metin”) olmak üzere altı alan adı eklenir. *Resim 10*



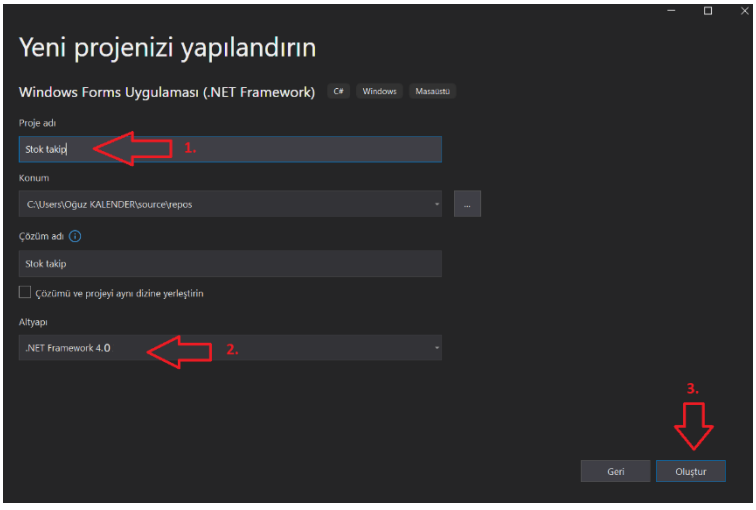
Resim 11

Son olarak; *Resim 4* ‘deki adımlar tekrarlanarak programda satılan ürünlerin kaydının tutulacağı tablo oluşturulup ardından *Resim 2* ‘deki adımlar tekrarlanarak tablo adı “Satışlar” olarak değiştirilir.

Satışlar tablosuna sırasıyla “BarkodNumarası”(veri türü “Uzun Metin”) , “ÜrünAdı” (veri türü “Uzun Metin”) , “Adet” (veri türü “Sayı”) , “SatışFiyatı” (veri türü “Para Birimi”) , “ToplamFiyat” (veri türü “Para Birimi”) , “Tarih” (veri türü “Uzun Metin”) olmak üzere altı alan adı eklenir ve X (çarpı) butonuna tıklanarak gelen uyarıda “Evet” butonu seçilip veri tabanı kaydedilerek programda kullanılacak veri tabanı oluşturma işlemi bitirilir. *Resim 11*

4.4.2.Yeni Proje Oluşturma

Program açıldıktan sonra projeyi oluşturmak için “Yeni bir proje oluştur” seçeneği tıklanır. Projede Windows Forms pencereleri ve Access veri tabanı kullanılacağından arama çubuğuna “forms” yazılarak “Windows Forms Uygulaması (.NET Framework)” seçilip “Sonraki” butonuna tıklanarak yeni proje oluşturulur.



Resim 1

Resim 1 1. Kodlanacak projeye verilen ad.

Resim 1 2. Projenin çalışacağı altyapı yazılımı ".NET Framework 4.0".

Resim 1 3. Proje oluşturma işlemi bitirilir.

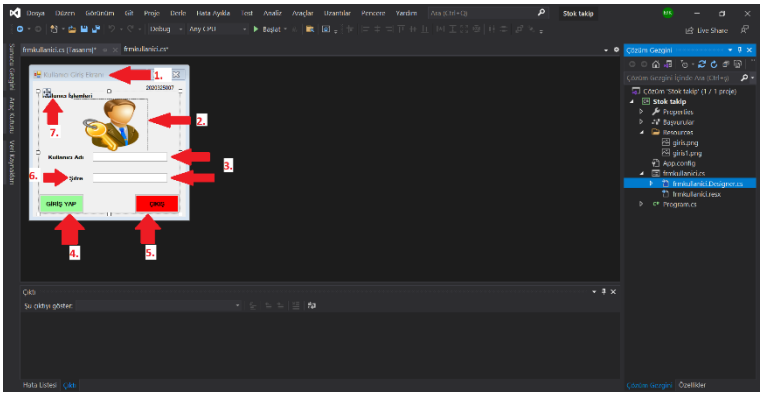
4.4.3.Projenin Tasarımı Ve Kodlanması

Program 64 bit işletim sistemlerinde çalışacağı için Projenin Çözüm Gezgini sekmesinde Properties (Özellikler) açılarak Derleme sekmesi kısmında Platform hedefi x64 seçilir.

Ardından üst menüden Derle menüsüne tıklanarak Yapılandırma Yöneticisi açılır. Açılan Yeni Çözüm Platformu penceresinde Yeni Platformu yaz veya seç kısmından x64 seçilerek tamam butonuna tıklanır. Son olarak Yapılandırma Yöneticisi kapatılır.

Önceki adımda oluşturulan veri tabanı dosyası StokVerisi.accdb Stok takip proje dosyasının bulunduğu klasöre taşınır ve Visual Studio 'da çözüm gezgini penceresi içerisinde projeye dahil edilir. Dosya yolu: source\repos\Stok takip

Kullanıcı Giriş Penceresi Görsel Tasarımı frmkullanici.cs



Resim 2

Resim 2 1. Kullanıcı penceresinin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmkullanici" dir. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 2 2. Pencere içerisine eklenen "PitcureBox". (Kullanıcı giriş Simgesi)

Resim 2 3. Kullanıcı adı ve şifre için text girişi alınacak iki adet "TextBox".

Resim 2 4. Girilen kullanıcı adı ve şifrenin onaylanıp eğer doğruysa programa giriş yapılacak, değilse hata mesajı gösterilecek giriş butonu.

Resim 2 5. Programdan çıkış için çıkış butonu.

Resim 2 6. Son kullanıcıya kullanıcı adı ve şifre girilmesi gerektiğini belirtmek için iki adet "Label".

Resim 2 7. Penceredeki diğer nesnelerin içinde bulunduğu Kullanıcı işlemleri "GroupBox".

Kullanıcı Giriş Penceresi İşlev Kodları frmkullanici.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
namespace StokTakip
{
    public partial class frmkullanici : Form
    {
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
```

Programın kullanacağı sağlayıcıların tanımlanması, Access veri tabanı kullanılacağı için using System.Data.OleDb; eklendi. **Bu kısım diğer pencerelerin kodlamasında da kullanılacaktır.**

Access veri tabanı dosyasını programa bağlamak için baglanti olarak id tanımlanan komut. Dosya, derlenen projenin .exe dosyasının bulunduğu klasörde bulunmalı. Access database 2016 kullanılacağı için OLEDB.16.0 kullanıldı.

```
        OleDbCommand komut;
        OleDbDataReader read;
```

Kullanici penceresinde veri tabanı içindeki veriler okunacağı için DataReader ve Command komutları oluşturuldu.

```
        public OleDbDataReader kullanici(TextBox kullaniciadi, TextBox sifre)
        {
```

Kullanıcı adı ve şifre için kullanıcıdan veri almayı sağlayan TextBox parametreleri tanımlanan DataReader metodu. Girilen text lerin veri tabanında bulunup bulunmadığını kontrol edecek.

```
            baglanti.Open();
```

veri tabanına komutlar vermek ya da veri okumak için baglanti parametresi açılır. Ve komut kodları içerisine yazılır.

```
            komut = new OleDbCommand();
            komut.Connection = baglanti;
```

Komut adı verilen parametre veri tabanı dosya yolu ve adı belirtilen baglanti komutuna bağlandı.

```
            komut.CommandText = "select *from Kullanıcı where Kullanıcıadı='" +
kullaniciadi.Text + "'";
```

Komut parametresi içerisinde, veri tabanı dosyası içeriğinde Kullanıcı tablosu içindeki KullanıcıAdı ve Şifre Alan adlarındaki bilgiler kontrol edilir.

```
            read = komut.ExecuteReader();
```

Komut parametresinde belirtilen veri tabanı bilgileri ExecuteReader ile okunur.

```
            if (read.Read() == true)
        {       Eğer girilen veri, veri tabanındaki ile aynıysa
                if (sifre.Text == read["Şifre"].ToString())
```

Eğer girilen Şifre veri tabanından okunan şifre ile aynıysa kullanıcı programın ilk penceresi olan Satış Ekranı'na yönlendirilir.

```
            {
                    Formsatis Satışlar = new Formsatis();
```

```
                Satışlar.ShowDialog();
            }
             else
            {
                MessageBox.Show("Şifrenizi Kontrol Ediniz.", "HATA!");
            }
```

Girilen kullanıcı adı doğru, sadece şifre yanlışsa kullanıcıya Şifrenizi Kontrol Ediniz uyarı penceresi gösterilerek doğru şifre istenir.

```
        }
         else
        {
            MessageBox.Show("Kullanıcı Adınızı ve Şifrenizi Kontrol Ediniz.",
"HATA!");
        }
```

Kullanıcı adı ve şifrenin her ikiside yanlışsa kullanıcıya Kullanıcı Adınızı ve Şifrenizi Kontrol Ediniz uyarı penceresi gösterilir.

```
            baglanti.Close();
```

baglanti parametresi kapatılır.

```
        return read;
```

Okuma verisi döndürülür.

```
    }
        public frmkullanici()
        {
            InitializeComponent();
        }
        private void btncikisyap_Click(object sender, EventArgs e)
        {
```

Çıkış Yap butonuna tıklandığında işlenecek komut.

```
        Application.Exit();
```

Programdan çıkma komutu.

```
        }
        private void button1_Click(object sender, EventArgs e)
        {
```

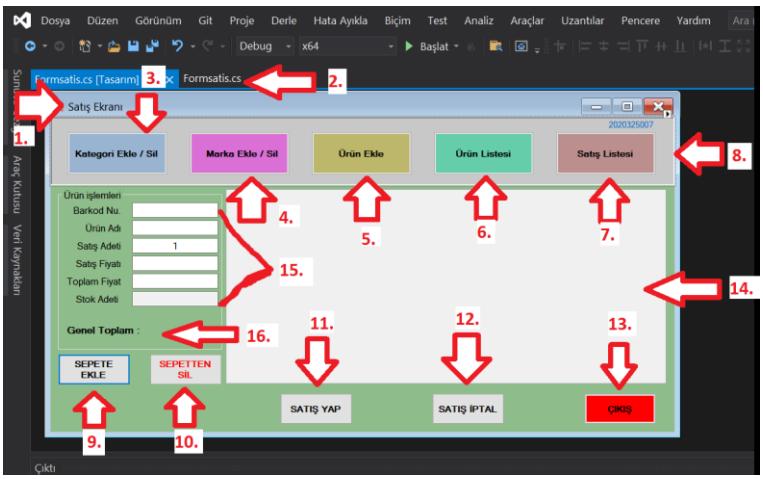
Giriş yap butonuna tıklandığında yapılacak işlem.

```
        kullanici(txtkullaniciadi,txtsifre);
```

Kullanıcı metodu tanımlanarak kullanılır.

```
        }
    }
}
```

Satış Penceresi Görsel Tasarımı Formsatis.cs



Resim 3

Resim 3 1. Kullanıcının Göreceği satış penceresinin görünen adı.

Resim 3 2. Satış penceresinin benzersiz nesne adı.

Resim 3 3. Kategori ekleme veya silme butonu. Bu butona tıklandığında Formsatis.cs kodlama alanında tanımlanan kodlar ile kullanıcı daha sonra oluşturulan frmkategori penceresini açar.

Resim 3 4. Marka ekleme veya silme butonu. Bu butona tıklandığında Formsatis.cs kodlama alanında tanımlanan kodlar ile kullanıcı daha sonra oluşturulan frmmarka penceresini açar.

Resim 3 5. Ürün ekleme butonu. Bu butona tıklandığında Formsatis.cs kodlama alanında tanımlanan kodlar ile kullanıcı daha sonra oluşturulan frmürünekle penceresini açar.

Resim 3 6. Ürün listeleme butonu. Bu butona tıklandığında Formsatis.cs kodlama alanında tanımlanan kodlar ile kullanıcı daha sonra oluşturulan frmürünlistele penceresini açar.

Resim 3 7. Satış listeleme butonu. Bu butona tıklandığında Formsatis.cs kodlama alanında tanımlanan kodlar ile kullanıcı daha sonra oluşturulan frmsatislistesi penceresini açar.

Resim 3 8. Panel. Panel dock=top işlemiyle Satış penceresinin üstüne yerleştirilir. Programın pencerelerini çalıştıran butonlar Panel içerisine dizilir.

Resim 3 9. Ürün işlemleri GridBox ın içinde barkod numarasıyla araması yapılan ve satış adeti girilen ürünü veri tabanındaki sepet tablosuna ekleme butonu.

Resim 3 10. Veri tabanı Sepet tablosuna eklenen ürün ve adet bilgilerinden fare ile seçileni sepetten silme butonu.

Resim 3 11. Veri tabanı Sepet tablosundaki ürünleri veri tabanındaki Satışlar tablosuna ekleyerek eklenen ürünlerin stok adetini düşürerek satış yapma butonu.

Resim 3 12. Veri tabanı Sepet tablosundaki ürünlerin tümünü silerek satışı iptal etme butonu.

Resim 3 13. Programı kapatma butonu. Bu butona tıklandığında kullanıcı, programı kapatmak isteyip istemediğini soran evet, hayır seçenekleri olan bir uyarı ekranıyla karşılaşır.

Resim 3 14. DataGridview. Veri tabanı Sepet tablosuna eklenen verilerin listeleneceği tablo.

Resim 3 15. Barkod numarası girilerek ürün araması yapılacak TextBox. Girilen barkod numarası kontrol edilerek veri tabanından ürün adı, satış fiyatı, stok adeti (sadece okuma) bilgilerinin getirileceği TextBox'lar ve kullanıcının getirilen üründen kaç adet satılacağını girmesi gereken TextBox (Satılacak adetin Varsayılan değeri 1 dir.)

Resim 3 16. Sepetteki ürünlerin toplam fiyatının hesaplanıp kullanıcıya gösterileceği iki adet label. İlk label ekranda görünecek Genel toplam yazısı için. İkincisi ise hemen sağında boş bırakılarak, kodlama aşamasında sepetteki ürünlerin fiyat toplama işleminin yapılıp tanımlanacağı label.

Satış Penceresi Kodları Formsatis.cs

```
namespace StokTakip
{
    public partial class Formsatis : Form
    {
        public Formsatis()
        {
            InitializeComponent();
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
```

Access veri tabanı dosyasını programa bağlamak için baglanti olarak id tanımlanan komut. Dosya, derlenen projenin .exe dosyasının olduğu yerde bulunmalı. Access database 2016 kullanılacağı için OLEDB.16.0 kullanıldı.

```
        DataSet daset = new DataSet();
```

Kullanıcının sepete eklediği ürünlerin, daha sonra datagridview de kullanılarak listenebilmesi için data deposu oluşturuldu.

```
        private void sepetlistesi()
        {
            baglanti.Open();
```

veri tabanı bağlantısı açıldı.

```
            OleDbDataAdapter adtr = new OleDbDataAdapter("select *from Sepet",baglanti);
```

DataSet verileri almak ve kaydetmek için Dataadapter oluşturulup, string komutuna baglanti adı verilen veri tabanındaki Sepet tablosundaki herşey komutu verildi.

```
            adtr.Fill(daset,"Sepet");
```

Sepetteki belirli daset satırlarını veri kaynağında olanlarla eşleşecek şekilde ekler veya yeniler.

```
            dataGridView1.DataSource = daset.Tables["Sepet"];
```

Sepet tablosundaki daset verileri dataGridview tablosu içinde gösterilir.

```
            baglanti.Close();
```

veri tabanı bağlantısı kapatıldı.

```
        }
        private void Formsatis_Load(object sender, EventArgs e)
        {
            sepetlistesi();
```

sepetlistesi metodu Formsatis sayfası yüklenirken oluşturulması için tanımlanır.

```
        }
        private void hesapla()
        {
```

Sepetteki ürünlerin satış fiyatları toplamı görüntülenebilmesi ve Toplam fiyat labeline tanımlanması için geri dönüşsüz metod oluşturulur.

```
            try
            {
                baglanti.Open();
                OleDbCommand komut = new OleDbCommand("select sum(ToplamFiyat) from
Sepet", baglanti);
```

Komut stringinde Sum fonksiyonu ile Sepet tablosundaki ürünlerin ToplamFiyat alan adındaki değerlerinin toplamını bulmak için kullanılır.

```
                lblgeneltoplam.Text = komut.ExecuteScalar() + " TL";
```

Komut stringi içindeki veri tabanı tablosundan Toplam Fiyatı tek bir değer olarak almak için kullanılır ve değerin yanına TL eklenerek Türk lirası olduğu kullanıcıya belirtilir.

```
                baglanti.Close();
            }
            catch (Exception)
            {
                ;
            }
        }
        private void btncikisyap_Click(object sender, EventArgs e)
        {
```

Çıkış yap butonuna tıklandığında işlenecek kodlar yazılır.

```
            DialogResult onayla = MessageBox.Show("Programdan Çıkmak İstediğinize Emin
misiniz?","UYARI!",MessageBoxButtons.YesNo);
```

Kullanıcı Çıkış Yap butonuna tıkladığında Programı kapatmak isteyip istemediğini soran bir evet, hayır seçenekli uyarı ekranı kodlanır.

```
            if (onayla==DialogResult.Yes)
            {
```

Kullanıcı eğer eveti seçerse program kapatılır.

```
                Application.Exit();
            }
        }
        private void btnurunekle_Click(object sender, EventArgs e)
        {
```

Kullanıcı Ürün ekle butonuna tıkladığında işlenecek kodlar.

```
            frmürünekle ekle = new frmürünekle();
            ekle.ShowDialog();
```

Kullanıcı ürün ekleme sayfasına yönlendirilir.

```
        }
        private void button1_Click(object sender, EventArgs e)
        {
```

Kullanıcı Kategori Ekle/Sil butonuna tıkladığında işlenecek kodlar.

```
            frmkategori Kategori = new frmkategori();
            Kategori.ShowDialog();
```

Kullanıcı Kategori ekleme-silme sayfasına yönlendirilir.

```
        }
        private void button2_Click(object sender, EventArgs e)
        {
```

Kullanıcı Marka Ekle/Sil butonuna tıkladığında işlenecek kodlar.

```
            frmmarka Marka = new frmmarka();
            Marka.ShowDialog();
```

Kullanıcı Marka ekleme-silme sayfasına yönlendirilir.

```
        }
```

```
        private void btnurunliste_Click(object sender, EventArgs e)
        {
```
Kullanıcı Ürünlerin listele butonuna tıkladığında işlenecek kodlar.
```
            frmürünlistele listele = new frmürünlistele();
            listele.ShowDialog();
```
Kullanıcı Ürünlerin listeleneceği frmürünlistele sayfasına yönlendirilir.
```
        }
        private void txtbarkodno_TextChanged(object sender, EventArgs e)
        {
```
Barkod numarası girilen textBox da veri değiştiğinde işlenecek kodlar.
```
            yenile();
```
yenile metodu tanımlanır.
```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Ürünler where
BarkodNumarası like '" + txtbarkodno.Text + "'", baglanti);
```
Komut string içerisinde, baglanti parametresinde, kullanıcının barkod numarası girerek ürün bilgilerine ulaşacağı TextBox kontrol edilerek tanımlanan veri tabanı Ürünler tablosundaki BarkodNumarası alan adındaki verilerin hepsiyle eşleşmesi aranır.
```
            OleDbDataReader read = komut.ExecuteReader();
```
Komut stringinde belirtilen veri tabanındaki bilgiler okunur.
```
            while (read.Read())
            {
```
Veri tabanından read ile okunan verilerin parametresini içeren While döngüsü.
```
                txturunadi.Text = read["ÜrünAdı"].ToString();
                txtsatisfiyat.Text = read["SatışFiyatı"].ToString();
                if (txtbarkodno.Text == "")
                {
                    foreach (Control item in groupBox2.Controls)
                    {
```
GroupBox2 içindeki öğeleri kontrol etmek için Foreach döngüsü.
```
                        if (item is Label)
                        {
```
Eğer öğe Label ise
```
                            if (item != txtstok)
                            {
```
Eğer öğe txtstok daki veriye eşit değilse.
```
                                item.Text = "";
```
Öğe texti boş tanımlanır.
```
                            }
                        }
                    }
                }
                txtstok.Text = read["Adet"].ToString();
```
Veri tabanından okunan tablonun Adet alan adı altındaki veri, Stok text içine tanımlanır. Burada ürünün Stokta kaç adet kaldığı gösterilir.
```
            }
            baglanti.Close();
        }
        private void yenile()
```
Barkod numarası alanı boş olduğunda diğer textBoxların değerlerini temizlemek için yenile metodu.

```
{
    if (txtbarkodno.Text == "")
    { Eğer Barkod numarası TextBox ı boş ise
        foreach (Control item in groupBox2.Controls)
        { Döngü ile GroupBox2 içerisindeki öğeleri kontrol etmek için.
            if (item is TextBox)
            { Eğer GroupBox2 deki öğeler Textbox ise
                if (item != txtstokadet)
                { Eğer öğeler txtstokadet e eşit değilse
                    item.Text = "";
                   Öğelerin içeriği temizlenir.
                }
            }
        }
    }
}
bool durum;
```

Durum adında Boolean değişkeni tanımlanır.

```
private void barkodkontrol()
{
```

Kullanıcı barkod numarası girdiğinde işlenecek Barkodkontrol metodu tanımlanır.

```
    durum = true;
```

durum değişkenine true değeri verilir.

```
    baglanti.Open();
    OleDbCommand komut = new OleDbCommand("select *from Sepet",baglanti);
```

Komut Stringinde, baglanti değişkeninde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilerek tablonun verilerine ulaşılır.

```
    OleDbDataReader read = komut.ExecuteReader();
```

Komut stringinde belirtilen veri tabanındaki bilgiler okunur.

```
    while (read.Read())
    {
```

Veri tabanından read ile okunan verilerin parametresini içeren While döngüsü.

```
        if (txtbarkodno.Text==read["BarkodNumarası"].ToString())
        {
```

Eğer girilen barkod numarası, read ile veri tabanından okunan değere eşitse. Köşeli parantez içindeki değer; okunan tablonun içerisinde BarkodNumarası alan adı altındaki verileri okumak için.

```
            durum = false;
```

durum değişkeni false değerine çekilir.

```
        }
    }
    baglanti.Close();
}
private void btnekle_Click(object sender, EventArgs e)
{
```

Kullanıcı Sepete Ekle butonuna tıkladığında çalıştırılacak komutlar.

```
    barkodkontrol();
```

barkodkontrol metodu kullanılır.

```
            if (durum==true)
            {
```

Eğer durum değişkeni true ise

```
                baglanti.Open();
                OleDbCommand komut = new OleDbCommand("insert into
Sepet(BarkodNumarası,ÜrünAdı,Adet,SatışFiyatı,ToplamFiyat,Tarih)
values(@BarkodNumarası,@ÜrünAdı,@Adet,@SatışFiyatı,@ToplamFiyat,@Tarih)", baglanti);
```

Komut Stringinde, baglanti değişkeninde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilerek tablodaki BarkodNumarası, ÜrünAdı, Adet, SatışFiyatı, ToplamFiyat, Tarih alan adlarına veri eklemek için insert into komutu yazılır. Bu alan adlarına Values içinde @ ile değer tanımlanır.

```
                komut.Parameters.AddWithValue("@BarkodNumarası", txtbarkodno.Text);,
```

Komut Stringinde tanımlanan veri tabanı tablosunun @BarkodNumarası değerine, kullanıcının Ürün işlemleri GroupBox 1 içinde girdiği barkod numarası değeri eklenir.

```
                komut.Parameters.AddWithValue("@ÜrünAdı", txturunadi.Text);
```

Komut Stringinde tanımlanan veri tabanı tablosunun @ÜrünAdı değerine, read ile okunarak Ürün işlemleri GroupBox 1 içinde gösterilen ürün adı değeri eklenir.

```
                komut.Parameters.AddWithValue("@Adet", int.Parse(txtstokadet.Text));
```

Komut Stringinde tanımlanan veri tabanı tablosunun @Adet değerine, kullanıcının Ürün işlemleri GroupBox 1 içinde girdiği satış adeti değeri int sayı tipine dönüştürülerek eklenir.

```
                komut.Parameters.AddWithValue("@SatışFiyatı",
double.Parse(txtsatisfiyat.Text));
```

Komut Stringinde tanımlanan veri tabanı tablosunun @SatışFiyatı değerine, read ile okunarak Ürün işlemleri GroupBox 1 içinde gösterilen satış fiyatı değeri ondalıklı (double) sayı tipine dönüştürülerek eklenir.

```
                komut.Parameters.AddWithValue("@ToplamFiyat",
double.Parse(txttoplamfiyat.Text));
```

Komut Stringinde tanımlanan veri tabanı tablosunun @ToplamFiyat değerine, Sepete eklenen ürünün toplam fiyatı Ürün işlemleri GroupBox 1 içinde gösterilen Toplam fiyat değeri ondalıklı (double) sayı tipine dönüştürülerek eklenir.

```
                komut.Parameters.AddWithValue("@Tarih", DateTime.Now.ToString());
```

Komut Stringinde tanımlanan veri tabanı tablosunun @ToplamFiyat değerine, Ürünün sepete eklendiği tarih ve saat eklenir.

```
                komut.ExecuteNonQuery();
```

veri tabanının yapısını sorgulamak veya tabloların içine veri eklemek için kullanılır.

```
                baglanti.Close();
            }
             Else
```

Eğer durum true değilse işlenecek kodlar tanımlanır.

```
            {
```

```
                baglanti.Open();
                OleDbCommand komut1 = new OleDbCommand("update Sepet set
Adet=Adet+'"+int.Parse(txtstokadet.Text)+ "' where BarkodNumarası='" + txtbarkodno.Text
+ "'", baglanti);
```

Bir ürünün Barkod numarası baz alınarak mevcut stok sayısının üstüne kullanıcının eklemek istediği sayı değeri eklenip veri güncellemesi için, komut1 Stringinde, baglanti değişkeni içerisinde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilerek tablodaki Adet alan adındaki sayının üstüne eklenecek veriyi txtstokadet textboxından alarak Barkod numarası alanında yazılı olan veriye göre var olan ürün adetini güncellemek için update komutu yazılır.

```
                komut1.ExecuteNonQuery();
                OleDbCommand komut2 = new OleDbCommand("update Sepet set
ToplamFiyat=Adet*SatışFiyatı where BarkodNumarası='"+txtbarkodno.Text+"'", baglanti);
```

Bir ürünün Barkod numarası baz alınarak kullanıcının girdiği adet ile ürünün satış fiyatı çarpım işlemi yapılıp ToplamFiyat verisinin güncellemesi için, komut2 Stringinde, baglanti parametresi verilip update komutları yazılır.

```
                komut2.ExecuteNonQuery();
                baglanti.Close();
            }
            txtstokadet.Text = "1";
```

Ürünün satılacağı adet varsayılan değeri 1 olarak tanımlanır.

```
            daset.Tables["Sepet"].Clear();
```

Sepet tablosu temizlenir.

```
            sepetlistesi();
```

sepetlistesi metodu kullanılır.

```
            hesapla();
```

hesapla metodu kullanılır.

```
            txtbarkodno.Clear();
```

Son olarak ürün işlemleri kısmındaki bilgilerin temizlenmesi için barkod numarası alanı temizlenir.

```
        }
        private void txtstokadet_TextChanged(object sender, EventArgs e)
        {
            try
            {
                txttoplamfiyat.Text = (double.Parse(txtstokadet.Text) *
double.Parse(txtsatisfiyat.Text)).ToString();
            }
            catch (Exception)
            {
                ;
            }
        }
        private void txtsatisfiyat_TextChanged(object sender, EventArgs e)
        {
            try
            {
                txttoplamfiyat.Text = (double.Parse(txtstokadet.Text) *
double.Parse(txtsatisfiyat.Text)).ToString();
```

Ürünün satılacak adeti ile ürün satış fiyatını çarpıp toplam fiyat Textbox ına tanımlanır.

```
            }
            catch (Exception)
```

```
            {
                ;
            }
        }
        private void btnsil_Click(object sender, EventArgs e)
        {
```
Kullanıcı Sepetten Sil butonuna tıkladığında işlenecek kodlar.
```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("delete from Sepet where
BarkodNumarası='"+dataGridView1.CurrentRow.Cells["BarkodNumarası"].Value.ToString()+"'
",baglanti);
```
Bir ürünün Barkod numarası baz alınarak sepetteki ürün bilgilerini silmek için, komut Stringinde, baglanti değişkeni içerisinde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilerek Datagridview1 ve veri tabanının Sepet tablosundaki listeden seçili ürünün Barkod numarası baz alınarak ürün bilgilerini silmek için delete komutları kullanılır.
```
            komut.ExecuteNonQuery();
            baglanti.Close();
            MessageBox.Show("Ürün Silme Başarılı.","ONAY");
```
Kullanıcıya onay mesaj ekranı gösterilerek kayıt silme işleminin başarılı olduğu bildirilir.
```
            daset.Tables["Sepet"].Clear();
```
Sepet tablosundaki tabloda listelenen veriler temizlenir.
```
            sepetlistesi();
```
sepeti güncellemek için sepetlistesi metodu kullanılır.
```
            hesapla();
```
güncellenen sepet listesindeki ürünlerin toplam fiyatını güncellemek için hesapla metodu kullanılır.
```
        }
        private void btnsatisiptal_Click(object sender, EventArgs e)
        {
```
Kullanıcı Satış İptal butonuna tıkladığında işlenecek kodlar.
```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("delete from Sepet ", baglanti);
```
komut Stringinde, baglanti değişkeni içerisinde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilip içindekiler delete komutuyla silinir.
```
            komut.ExecuteNonQuery();
            baglanti.Close();
            MessageBox.Show("Satış İptali Başarılı.","ONAY");
```
Kullanıcıya onay mesaj ekranı gösterilerek Satış iptal etme işleminin başarılı olduğu bildirilir.
```
            daset.Tables["Sepet"].Clear();
```
Sepet tablosu temizlenir.
```
            sepetlistesi();
```
sepeti güncellemek için sepetlistesi metodu kullanılır.
```
            hesapla();
```
güncellenen sepet listesindeki ürünlerin toplam fiyatını güncellemek için hesapla metodu kullanılır.
```
        }
        private void btnsatisliste_Click(object sender, EventArgs e)
        {
```
Kullanıcı Satış Listeleme butonuna tıkladığında işlenecek kodlar.
```
            frmsatislistesi listele = new frmsatislistesi();
            listele.ShowDialog();
```
Satış listeleme ekranı (frmsatislistesi) penceresi açılır.

```
}
private void btnsatisyap_Click(object sender, EventArgs e)
{ Kullanıcı Satış Yap butonuna tıkladığında işlenecek kodlar.
    for (int i = 0; i < dataGridView1.Rows.Count-1; i++)
    { Sepette olan Datagridview listesindeki kayıtları Satışlar tablosuna aktarmak için
```
For döngüsü oluşturulur. dataGridView1.Rows.Count-1; komutu listedeki kayıtların tümü için -1 ifadesinin amacı listelenen tabloda varsayılan olarak her zaman bir adet boş sütün olduğundan "i" değişkenini de 0 dan başlatıldığı için bir sütun eksik alması.

```
        baglanti.Open();
        OleDbCommand komut = new OleDbCommand("insert into
Satışlar(BarkodNumarası,ÜrünAdı,Adet,SatışFiyatı,ToplamFiyat,Tarih)
values(@BarkodNumarası,@ÜrünAdı,@Adet,@SatışFiyatı,@ToplamFiyat,@Tarih)", baglanti);
```
Komut Stringinde, baglanti değişkeninde tanımlanan veri tabanına ulaşılıp Satışlar tablosundaki alan adları olan BarkodNumarası, ÜrünAdı, Adet, SatışFiyatı, ToplamFiyat, Tarih alan adlarına veri eklemek için insert into komutu yazılır. Bu alan adlarına Values içinde @ ile id tanımlanır.

```
        komut.Parameters.AddWithValue("@BarkodNumarası",
dataGridView1.Rows[i].Cells["BarkodNumarası"].Value.ToString());
```
Datagridview listesindeki i dizini barkod numarası hücresi komut stringinde Values ile id tanımlanan Satışlar tablosundaki @BarkodNumarası alan adına eklenir.

```
        komut.Parameters.AddWithValue("@ÜrünAdı",
dataGridView1.Rows[i].Cells["ÜrünAdı"].Value.ToString());
```
Datagridview listesindeki i dizini Ürün Adı hücresi komut stringinde Values ile id tanımlanan Satışlar tablosundaki @ÜrünAdı alan adına eklenir.

```
        komut.Parameters.AddWithValue("@Adet",
int.Parse(dataGridView1.Rows[i].Cells["Adet"].Value.ToString()));
```
Datagridview listesindeki i dizini Adet hücresi komut stringinde Values ile id tanımlanan Satışlar tablosundaki @Adet alan adına eklenir.

```
        komut.Parameters.AddWithValue("@SatışFiyatı",
double.Parse(dataGridView1.Rows[i].Cells["SatışFiyatı"].Value.ToString()));
```
Datagridview listesindeki i dizini SatışFiyatı hücresi komut stringinde Values ile id tanımlanan Satışlar tablosundaki @SatışFiyatı alan adına double (ondalık) sayıya dönüştürülerek eklenir.

```
        komut.Parameters.AddWithValue("@ToplamFiyat",
double.Parse(dataGridView1.Rows[i].Cells["ToplamFiyat"].Value.ToString()));
```
Datagridview listesindeki i dizini ToplamFiyat hücresi komut stringinde Values ile id tanımlanan Satışlar tablosundaki @ToplamFiyat alan adına double (ondalık) sayıya dönüştürülerek eklenir.

```
        komut.Parameters.AddWithValue("@Tarih", DateTime.Now.ToString());
```
komut stringinde Values ile id tanımlanan Satışlar tablosundaki @Tarih alan adına eklendiği tarih ve saat verileri eklenir.

```
        komut.ExecuteNonQuery();
```

```
                OleDbCommand komut1 = new OleDbCommand("update Ürünler set Adet=Adet-'"
+    int.Parse(dataGridView1.Rows[i].Cells["Adet"].Value.ToString())    +    "'    where
BarkodNumarası='"  +  dataGridView1.Rows[i].Cells["BarkodNumarası"].Value.ToString()  +
"'", baglanti);
```

Sepet Datagridview1 listesindeki bir ürünün Barkod numarası baz alınarak mevcut stok sayısından satılan ürün adeti kadar düşülüp Ürünler tablosundaki adet verisinin güncellemesi için, komut1 Stringinde, baglanti değişkeni içerisinde tanımlanan veri tabanına ulaşılıp Ürünler tablosu seçilerek tablodaki Adet alan adındaki sayıdan satılan adet sayısı kadar düşülerek var olan ürün adetini güncellemek için update komutları yazılır.

```
                komut1.ExecuteNonQuery();
                baglanti.Close();
            }
            baglanti.Open();
            OleDbCommand komut2 = new OleDbCommand("delete from Sepet ", baglanti);
```

Komut2 Stringinde, baglanti değişkeni içerisinde tanımlanan veri tabanına ulaşılıp Sepet tablosu seçilip içindekiler delete komutuyla silinir.

```
            komut2.ExecuteNonQuery();
            baglanti.Close();
            daset.Tables["Sepet"].Clear();
```

Sepet tablosu temizlenir.

```
            sepetlistesi();
```

sepeti güncellemek için sepetlistesi metodu kullanılır.

```
            hesapla();
```

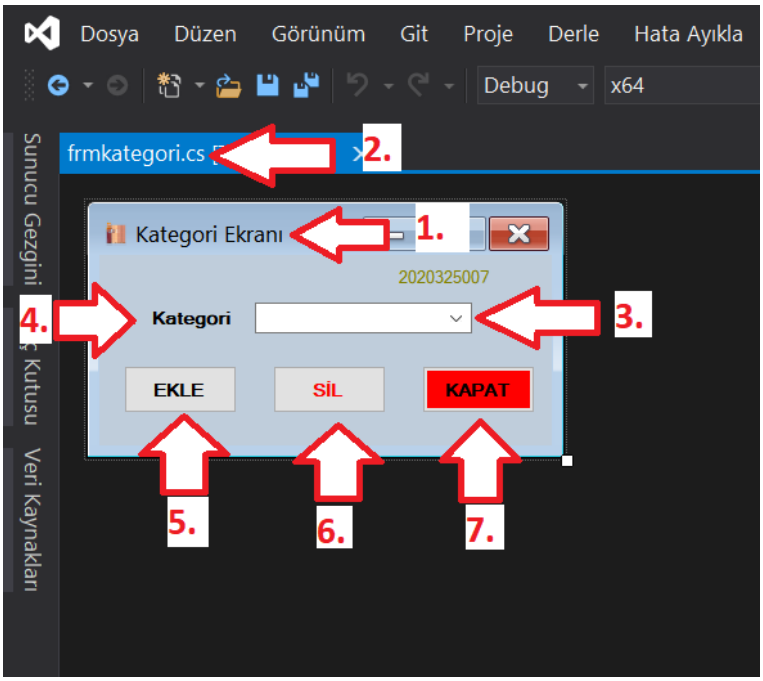
güncellenen sepet listesindeki ürünlerin toplam fiyatını güncellemek için hesapla metodu kullanılır.

```
            MessageBox.Show("Satış Yapıldı.","ONAY");
```

Kullanıcıya Satış Yapma işleminin başarılı olduğunu gösteren bir onay penceresi gösterilir.

```
        }
    }
}
```

Kategori Ekleme Ve Silme Penceresi Görsel Tasarım



Resim 4

Resim 4 1. Kategori Ekleme veya silme penceresinin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmkategori" dir. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 4 2. Pencerenin tanımlanan değişmez adı.

Resim 4 3. Kategori text girişi alınacak bir "ComboBox". (Kullanıcının veri tabanındaki kategorileri görebilmesini sağlamak için ComboBox kullanıldı.)

Resim 4 4. Kullanıcıya kategori girilmesi gerektiğini belirtmek için bir "Label".

Resim 4 5. Girilen kategori textini, veri tabanındaki Kategoriler tablosuna ekleyecek giriş butonu.

Resim 4 6. Kullanıcı veri tabanında bulunan herhangi bir kategoriyi silmek istediğinde ComboBox da seçilen kategori texti baz alınarak veri tabanından ilgili kategoriyi silme butonu.

Resim 4 7. Kategori penceresini kapatıp ana ekrana dönmek için kapat butonu.

Kategori Ekleme Veya Silme Penceresi Kodları frmkategori.cs

```
namespace StokTakip
{
    public partial class frmkategori : Form
    {
        public frmkategori()
        {
            InitializeComponent();
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
```

Veri tabanı dosyası sağlayıcısı ve konumu belirtilir.

```
        bool durum;
```

Boolean veri türünde durum değişkeni tanımlanır.

```
        private void kategorikontrol()
        { 
```

Kategorinin veri tabanında olup olmadığını ya da kategori ekleme ve silme işlemlerini kodlama metodu.

```
            durum = true;
```

durum değişkeni true olarak tanımlanır.

```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from
Kategoriler",baglanti);
```

Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Kategoriler tablosu içindeki bütün bilgiler seçilir.

```
            OleDbDataReader read = komut.ExecuteReader();
```

Komutlarda veri tabanından veri okunacağı için kullanılır.

```
            while (read.Read())
            { 
```

While döngüsü ile Kategori comboBox a veri girldiğinde veri tabanındaki Kategori tablosu içindeki verilerin kontrolü için okuma yapılır.

```
                if (comboBox1.Text==read["Kategori"].ToString()|| comboBox1.Text=="")
```

```
                {  Eğer Giriş alınan comboBox verisi Kategori tablosundaki okunan
verilerden birine eşitse veya comboBox boş bırakılmışsa durum false olur.
                    durum = false;
                }
            }
            baglanti.Close();
        }
        private void kategorigetir()
        { Kullanıcı mevcut kategorileri görmek için ComboBox ı tıkladığında kategorilerin
listelenmesi için metod oluşturulur.
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Kategoriler",
baglanti);
```

Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Kategoriler tablosu içindeki bütün bilgiler seçilir.

```
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            { Okunan verilere ComboBox içindeki verinin eklenmesi için döngü oluşturulur.
                comboBox1.Items.Add(read["Kategori"].ToString());
```

ComboBox daki girilen veriler okuma yapılan veri tabanında Kategoriler tablosu içerisine eklenir.

```
            }
            baglanti.Close();
        }
        private void button2_Click(object sender, EventArgs e)
        { Kullanıcı Kapat butonuna tıkladığında çalıştırılacak komut.
            Close();   Aktif olan pencere kapatılır. (Program kapatılmaz)
        }
        private void frmkategori_Load(object sender, EventArgs e)
        {
            kategorigetir();
```

Kategorilerin pencere açıldığında yüklenmesi için kategorigetir metodu çağırılır.

```
        }
        private void button1_Click(object sender, EventArgs e)
        { Kullanıcı ekle butonuna tıkladığında çalıştırılacak kodlar.
            kategorikontrol();
```

Butona tıklandığında öncelikle girilen kategorinin veri tabanında olup olmadığını kontrol amacıyla kategorikontrol metodu çağırılır.

```
            if (durum==true)
            { Eğer kategorikontrol metodundaki durum değişkeni true ise
                baglanti.Open();
                OleDbCommand komut = new OleDbCommand("insert into
Kategoriler(Kategori) values('" + comboBox1.Text + "')", baglanti);
```

Komut stringinde, baglanti değişkenindeki veri tabanı Kategoriler tablosundaki Kategori alan adına, kullanıcının kategori adını yazdığı ComboBox eklenerek Values içinde girilen metin değeri verilir.

```
                komut.ExecuteNonQuery();
                baglanti.Close();
                MessageBox.Show("Kategori Ekleme Başarılı.","ONAY");
```

Kullanıcıya Kategori Ekleme işleminin başarılı olduğunu gösteren Onay penceresi gösterilir.

```
            }
            else
            { Değilse (Durum false ise) işlenecek kodlar.
                MessageBox.Show("Girdiğiniz Kategori Zaten Var","HATA! ");
```

Kullanıcıya Kategori Ekleme işleminin başarılı olduğunu gösteren Onay penceresi gösterilir.

```
            }
            comboBox1.Text = "";          Son olarak ComboBox temizlenir.

        }
        private void btnsil_Click(object sender, EventArgs e)
        { Kullanıcı sil butonuna tıkladığında çalıştırılacak kodlar.
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("delete from Kategoriler where
Kategori='" + comboBox1.Text + "'", baglanti);
```

Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Kategoriler tablosu içindeki kategori alan adı baz alınarak silinmesi için delete komutları kullanılır.

```
            komut.ExecuteNonQuery();
            baglanti.Close();
            MessageBox.Show("Kategori Silme Başarılı.", "ONAY");
```

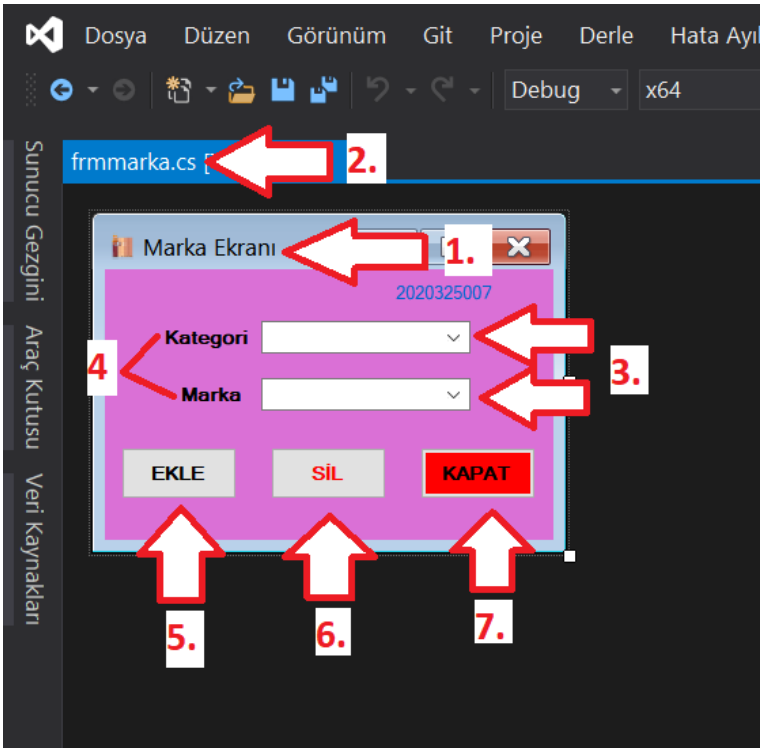
Kullanıcıya Seçilen kategorinin başarıyla silindiğini gösteren onay penceresi gösterilir.

```
            comboBox1.Text = "";
```

ComboBox1 içindeki veri temizlenir

```
        }
    }
}
```

Marka Ekleme Ve Silme Penceresi Görsel Tasarım



Resim 4

Resim 4 1. Marka Ekleme veya silme penceresinin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmmarka" dır. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 4 2. Pencerenin tanımlanan değişmez adı.

Resim 4 3. Kullanıcının Kategori seçmesi ve Marka text girişi yapması için iki adet "ComboBox". (Kullanıcının veri tabanında mevcut kategorileri ve seçili kategorideki markaları görebilmesini sağlamak için ComboBox kullanıldı.)

Resim 4 4. Kullanıcıya kategori seçmesi ve marka bilgisi girmesi gerektiğini belirtmek için iki adet "Label".

Resim 4 5. Seçilen kategori ve girilen marka textini, veri tabanındaki Markalar tablosuna ekleyecek giriş butonu.

Resim 4 6. Kullanıcı veri tabanında bulunan herhangi bir kategoriye ait markayı silmek istediğinde ComboBox da seçilen kategori ve marka baz alınarak veri tabanından ilgili markayı silme butonu.

Resim 4 7. Marka penceresini kapatıp ana ekrana dönmek için kapat butonu.

Marka Ekleme Ve Silme Penceresi Kodları frmmarka.cs

```
namespace StokTakip
{
    public partial class frmmarka : Form
    {
        public frmmarka()
        {
            InitializeComponent();
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
        bool durum;
```

Boolean veri türünde durum değişkeni tanımlanır.

```
        private void markakontrol()
        {
```

Girilen Markanın veri tabanında olup olmadığını kontrol etme işlemlerinin kodlanacağı metod.

```
            durum = true;
```

Durum değişkenine true değeri tanımlanır.

```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Markalar", baglanti);
```

Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Markalar tablosu içindeki bütün bilgiler seçilir.

```
            OleDbDataReader read = komut.ExecuteReader();
```

Komut Stringinde belirtilen dataların okunması için.

```
            while (read.Read())
            {
```

Okunan verilere ComboBox içindeki verinin eklenmesi için döngü oluşturulur.

```
                if (combokategori.Text==read["Kategori"].ToString() && combomarka.Text
== read["Marka"].ToString() || combokategori.Text=="" || combomarka.Text == "")
```

{ Eğer girilen kategori reader ile okunan Kategori tablosundaki veri ile eşitse ve girilen marka reader ile okunan Marka verisiyle eşitse ya da marka alanı boş ise ve kategori alanı boş ise.

```
                    durum = false;
```

Durum false değerini alır.

```
                }
            }
            baglanti.Close();
        }
        private void button2_Click(object sender, EventArgs e)
```

{ Kullanıcı Kapat butonuna bastığında çalıştırılacak kod.

```
            Close();
```

Pencereyi kapatma komutu.

```
        }
        private void button1_Click(object sender, EventArgs e)
```

{ Kullanıcı Ekle butonuna tıkladığında çalıştırılacak kod.

```
            markakontrol();
```

Öncelikle girilen markayı kontrol etmek için markakontrol metodu çağırılır.

```
            if (durum==true)
```

{ Eğer markakontrol metodundaki durum değişkeni true ise çalıştırılacak kod.

```
                baglanti.Open();
                OleDbCommand komut = new OleDbCommand("insert into
Markalar(Kategori,Marka) values('" + combokategori.Text + "','" + combomarka.Text +
"')", baglanti);
```

Komut stringinde, baglanti değişkenindeki veri tabanı Kategoriler tablosundaki Kategori ve Marka alan adları altına, kullanıcının kategori seçtiği ve marka girişi yaptığı ComboBox lar eklenerek Values içinde kullanıcının ComboBox lara girdiği metnin değeri verilir.

```
                komut.ExecuteNonQuery();
                baglanti.Close();
                MessageBox.Show("Marka Ekleme Başarılı.", "ONAY");
```

İşlem tamamlandiktan sonra kullanıcıya marka ekleme işleminin başarılı olduğu onay penceresi ile gösterilir.

```
            }
            else
```

{ Eğer metod da durum false döndüyse

```
                MessageBox.Show("Girdiğiniz Kategori veya Marka Zaten Var.","HATA! ");
```

Kullanıcıya Hata mesajı gösterilir.

```
            }
            combomarka.Text = "";
            combokategori.Text = "";
```

değerleri temizlenir

```
        }
        private void kategorigetir()
```

{ Kullanıcı mevcut kategorileri görmek için ComboBox ı tıkladığında kategorilerin listelenmesi için metod oluşturulur.

```
            baglanti.Open();
```

```
            OleDbCommand komut = new OleDbCommand("select *from Kategoriler",
baglanti);
```
Veri tabanından Kategoriler tablosu tüm içeriği seçilir.

```
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            {
                combokategori.Items.Add(read["Kategori"].ToString());
```
Veri tabanı okunması için

Kategorileri okumak için döngü

Veri tabanında içeriği okunacak alan adı ve nerede gösterileceği.

```
            }
            baglanti.Close();
        }
        private void frmmarka_Load(object sender, EventArgs e)
        {
            kategorigetir();
```
Marka ekleme penceresi açıldığında yüklenecek işlevler

kategorigetir metodu çağırılır.

```
        }
        private void combokategori_SelectedIndexChanged(object sender, EventArgs e)
        {
            combomarka.Items.Clear();
            combomarka.Text = "";
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Markalar where
Kategori='" + combokategori.SelectedItem + "'", baglanti);
```
Kullanıcı kategori comboBox ında seçim yaptığında işlenecek kodlar.

Marka alanı temizlenir.

Komut stringinde, baglanti değişkenindeki veri tabanı Markalar tablosu tüm içeriği seçilerek Kullanıcının seçtiği kategori gösterilir.

```
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            {
                combomarka.Items.Add(read["Marka"].ToString());
Marka comboBox ınında veri tabanında okunan Marka alan adı gösterilir.
            }
            baglanti.Close();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("delete from Markalar where Marka='"
+ combomarka.Text + "'", baglanti);
```
Veri tabanı Kullanıcı Sil Butonuna tıkladığında işlenecek kodlar.

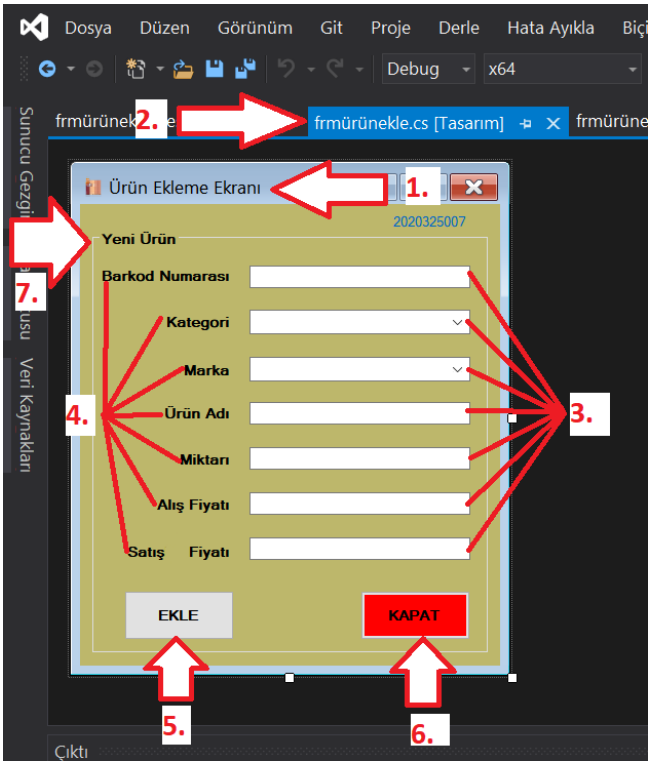
Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Markalar tablosu içindeki Marka verisinden girilen marka bilgisinin silinmesi için delete komutları kullanılır.

```
            komut.ExecuteNonQuery();
            baglanti.Close();
            MessageBox.Show("Kategori Silme Başarılı.", "ONAY");
```
İşlem tamamlandığında kullanıcıya silme işleminin başarılı olduğu bildirilen bir onay penceresi gösterilir.

```
            combomarka.Text = "";
            combokategori.Text = "";
        }
    }
}
```
Marka alanı temizlenir.

Kategori alanı temizlenir.

Ürün Ekleme Penceresi Görsel Tasarım



Resim 5

Resim 5 1. Ürün ekleme penceresinin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmürünekle" dır. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 5 2. Pencerenin tanımlanan değişmez adı.

Resim 5 3. Kullanıcının barkod numarası, ürün adı, adeti, alış fiyatı, satış fiyatı girmesi ile Kategori ve Marka seçmesi için beş adet "TextBox" ve iki adet "ComboBox".

Resim 5 4. Kullanıcıya barkod numarası, ürün adı, adet, alış fiyatı, satış fiyatı girmesi ile Kategori ve Marka seçmesi gerektiğini belirtmek için yedi adet "Label".

Resim 5 5. Seçilen kategori, marka ile kullanıcının girdiği barkod numarası, ürün adı, adeti, alış fiyatı, satış fiyatı textlerini, veri tabanındaki Ürünler tablosuna ekleyecek giriş butonu.

Resim 5 6. Ürün ekleme penceresini kapatıp ana ekrana dönmek için kapat butonu.

Resim 5 7. Tüm butonlar ve öğeleri yerleştirmek için Yeni ürün yazılı GroupBox.

Ürün Ekleme Penceresi Kodları

```
namespace StokTakip
{
    public partial class frmürünekle : Form
    {
        public frmürünekle()
        {
            InitializeComponent();
```

```
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
```
Veri tabanı bağlantı komutu

```
        bool durum;
```
Boolean veri türünde durum değişkeni tanımlanır.

```
        private void barkodkontrol()
        {
```
Kullanıcı barkod numarası girdiğinde işlenecek Barkodkontrol metodu tanımlanır.

```
            durum = true;
```
durum değişkeni true olarak tanımlanır.

```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Ürünler", baglanti);
```
Komut stringinde, veri tabanı yolu tanımlanan baglanti dosyasından Ürünler tablosu içindeki tüm veri seçilir.

```
        OleDbDataReader read = komut.ExecuteReader();
```
Veri tabanından okuma yapmak için

```
            while (read.Read())
            {
```
barkodno textbox ı içindeki verinin veri tabanındaki BarkodNumarası alan adından okunması için döngü oluşturulur.

```
                if (txtbarkodno.Text == read["BarkodNumarası"].ToString() ||
txtbarkodno.Text == "")
                {
```
Eğer girilen barkod numarası Veri tabanındaki barkod numaralararından biri ile eşitse ya da barkod numarası alanı boş ise

```
                    durum = false;
```
Durum değişkeni false olur.

```
                }
            }
            baglanti.Close();
        }
        private void button1_Click(object sender, EventArgs e)
        {
```
Kullanıcı Kapat butonuna tıkladığında işlenecek kodlar.

```
            Close();
```
Ürün ekleme penceresi kapatılır.

```
        }
        private void kategorigetir()
        {
```
Kullanıcı kategori ComboBox ını tıkladığında Kategorilerin listelenmesi için metod oluşturulur.

```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Kategoriler",
baglanti);
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            {
                combokategori.Items.Add(read["Kategori"].ToString());
            }
            baglanti.Close();
        }
        private void frmürünekle_Load(object sender, EventArgs e)
        {
```
Ürün ekleme penceresi açıldığında çalıştırılacak kodlar.

```
            kategorigetir();
```
kategorigetir metodu çağırılır.

```
        }
        private void combokategori_SelectedIndexChanged(object sender, EventArgs e)
        {
```
Kullanıcı Kategori seçtiğinde işlenecek kodlar.

```
            combomarka.Items.Clear();
            combomarka.Text = "";
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Markalar where
Kategori='" + combokategori.SelectedItem + "'", baglanti);
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            {
                combomarka.Items.Add(read["Marka"].ToString());
            }
            baglanti.Close();
        }
        private void btnyeniekle_Click(object sender, EventArgs e)
        {
            barkodkontrol();
            if (durum == true)
            {
                baglanti.Open();
                OleDbCommand komut = new OleDbCommand("insert into
Ürünler(BarkodNumarası,Kategori,Marka,ÜrünAdı,Adet,AlışFiyatı,SatışFiyatı,Tarih)
values(@BarkodNumarası,@Kategori,@Marka,@ÜrünAdı,@Adet,@AlışFiyatı,@SatışFiyatı,@Tarih
)", baglanti);
                komut.Parameters.AddWithValue("@BarkodNumarası", txtbarkodno.Text);
                komut.Parameters.AddWithValue("@Kategori", combokategori.Text);
                komut.Parameters.AddWithValue("@Marka", combomarka.Text);
                komut.Parameters.AddWithValue("@ÜrünAdı", txturunadi.Text);
                komut.Parameters.AddWithValue("@Adet", int.Parse(txtmiktari.Text));
                komut.Parameters.AddWithValue("@AlışFiyatı", txtalisfiyati.Text);
                komut.Parameters.AddWithValue("@SatışFiyatı", txtsatisfiyati.Text);
                komut.Parameters.AddWithValue("@Tarih", DateTime.Now.ToString());
                komut.ExecuteNonQuery();
                baglanti.Close();
                MessageBox.Show("Ürün Ekleme Başarılı", "ONAY");
            }
            else
            {
                MessageBox.Show("Girdiğiniz Barkod Numarası Zaten Var.", "HATA!");
            }
            combomarka.Items.Clear();
            foreach (Control item in groupBox1.Controls)
            {
                if (item is TextBox)
                {
                item.Text = "";
                }
                if (item is ComboBox)
                {
                    item.Text = "";
                }
            }
        }
    }
}
```

Marka alanı Temizlenir.

Seçilen kategoriye göre Markalar tablosundaki verilerin tümü seçilir.

Kullanıcı ekle butonuna tıkladığında işlenecek kodlar.

öncelikle barkod kontrolü için metod çağırılır.

Eğer barkodkontrol metodunda durum değişkeni true ise

Kullanıcının girdiği ürün verileri Ürünler tablosundaki alan adlarına eklenir

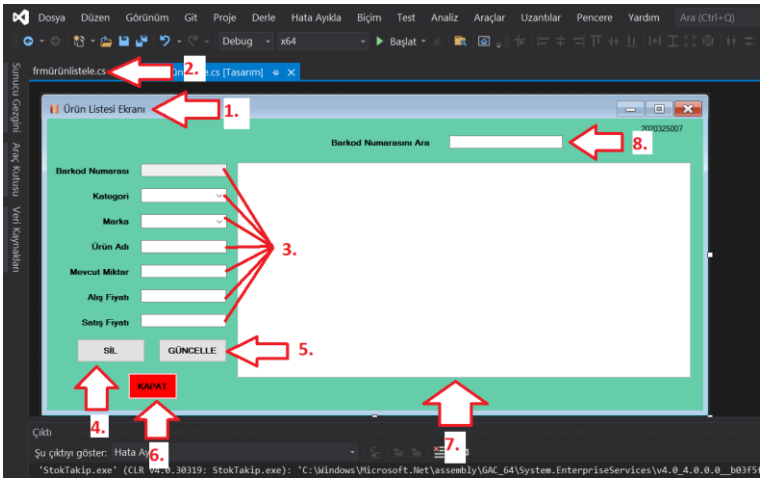
Son olarak onay penceresi ile kullanıcıya işlemin başarılı olduğu gösterilir.

Eğer barkodkontrol metodunda durum değişkeni true DEĞİL ise

Kullanıcıya Hata mesajı gösterilir.

Son olarak tüm ComboBox ve TextBox alanları temizlenir.

Ürün Listeleme Penceresi Görsel Tasarım



Resim 7

Resim 7 1. Ürün listeleme penceresinin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmürünlistele" dir. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 7 2. Pencerenin tanımlanan değişmez adı.

Resim 7 3. Kullanıcının barkod numarası, ürün adı, adeti, alış fiyatı, satış fiyatı girmesi ile Kategori ve Marka seçmesi için beş adet "TextBox" ve iki adet "ComboBox".

Resim 7 4. Kullanıcının seçtiği ürünün Ürünler tablosundaki verilerini silme butonu.

Resim 7 5. Kullanıcının seçtiği ürünün Ürünler tablosundaki verilerini güncelleme butonu.

Resim 7 6. Ürün listeleme penceresini kapatıp ana ekrana dönmek için kapat butonu.

Resim 7 7. Ürünleri listelemek için DataGridview.

Resim 7 8. Kullanıcıdan, ürün araması yapmak için barkod numarası girdisi alarak aranan ürünün bilgilerini datagridview de görüntülemek için TextBox. (Arama çubuğu)

Ürün Listeleme Penceresi Kodları

```
namespace StokTakip
{
    public partial class frmürünlistele : Form
    {
        public frmürünlistele()
        {
            InitializeComponent();
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
        DataSet daset = new DataSet();
        private void btncikis1_Click(object sender, EventArgs e)
        {Kullanıcı Kapat butonuna tıkladığında işlenecek kod.

            Close(); Pencere kapatılır.
        }
        private void kategorigetir()
```

```
        { Kullanıcının listeden seçtği ürün kategorisini ComboBox da göstermek için.
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Kategoriler",
baglanti); Kategoriler tablosu içindekilerin tümünü seçme komutu
            OleDbDataReader read = komut.ExecuteReader(); Okuma komutu için.
            while (read.Read())
            { verileri okuma döngüsü
                comboBox1.Items.Add(read["Kategori"].ToString());
            } Okunan veriyi kategori alanında göstermek için
            baglanti.Close();
        }
        private void frmürünlistele_Load(object sender, EventArgs e)
        { Ürün listeleme penceresi açıldığında işlenecek kodlar.
            urunlistele();     Ürün listeleme metodu çağırılır.
            kategorigetir();   kategori okuma metodu çağırılır.
        }
        private void urunlistele()
        { Kullanıcının datagridview deki listeden seçtiği ürün baz alınarak bilgileri
ComboBox ve TextBoxlarda göstermek için metod.
            baglanti.Open();
            OleDbDataAdapter adtr = new OleDbDataAdapter("select *from Ürünler",
baglanti); Ürünler tablosunun tüm içeriği seçilir.
            adtr.Fill(daset, "Ürünler");
            dataGridView1.DataSource = daset.Tables["Ürünler"];
Ürünler datagridview1 de listelenir ve gösterilir.
            baglanti.Close();
        }
        private void dataGridView1_CellMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        { Kullanıcı Ürün listesinden bir ürüne tıkladığında işlenecek kodlar.
            barkodnotxt.Text =
dataGridView1.CurrentRow.Cells["BarkodNumarası"].Value.ToString();
            comboBox1.Text =
dataGridView1.CurrentRow.Cells["Kategori"].Value.ToString();
            comboBox2.Text = dataGridView1.CurrentRow.Cells["Marka"].Value.ToString();
            urunaditxt.Text =
dataGridView1.CurrentRow.Cells["ÜrünAdı"].Value.ToString();
            txtmiktar.Text = dataGridView1.CurrentRow.Cells["Adet"].Value.ToString();
            txtalisfiyati.Text =
dataGridView1.CurrentRow.Cells["AlışFiyatı"].Value.ToString();
            txtsatisfiyati.Text =
dataGridView1.CurrentRow.Cells["SatışFiyatı"].Value.ToString();
        } Kullanıcı, ürünlerin listelendiği Datagridview1 den ürün seçtiğinde veri
tabanındaki bilgilerin okunarak ComboBox ve Textbox larda gösterilmesi için.
        private void btnguncelle_Click(object sender, EventArgs e)
        { Kullanıcı Güncelle butonuna tıkladığında işlenecek kodlar.
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("update Ürünler set
BarkodNumarası='"+ barkodnotxt.Text +"',Kategori='"+comboBox1.Text+ "',Marka='" +
comboBox2.Text + "',ÜrünAdı='" + urunaditxt.Text + "',Adet='" + txtmiktar.Text +
"',AlışFiyatı='" + txtalisfiyati.Text + "',SatışFiyatı='" + txtsatisfiyati.Text + "'
where BarkodNumarası='" + barkodnotxt.Text + "'", baglanti);
```

```
            komut.ExecuteNonQuery();
            baglanti.Close();
            daset.Tables["Ürünler"].Clear();
            urunlistele();
            MessageBox.Show("Güncelleme Başarılı.","ONAY");
```

Tablo Temizlenir.

Ürünler tekrar Listelenir.

Kullanıcıya işlemin başarılı olduğu, onay penceresi ile gösterilir.

```
            foreach (Control item in this.Controls)
            {
                if (item is ComboBox)
                {
                    item.Text = "";
                }
                if (item is TextBox)
                {
                    item.Text = "";
                }
            }
        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
```

Son olarak TextBox ve ComboBoxlar Temizlenir

Kullanıcı kategori seçtiğinde işlenecek kodlar.

```
            comboBox2.Items.Clear();
            comboBox2.Text = "";
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("select *from Markalar where
Kategori='" + comboBox1.SelectedItem + "'", baglanti);
            OleDbDataReader read = komut.ExecuteReader();
            while (read.Read())
            {
                comboBox2.Items.Add(read["Marka"].ToString());
            }
            baglanti.Close();
        }
        private void txturunara_TextChanged(object sender, EventArgs e)
        {
```

marka alanı temizlenir

bağlantı açılır

Marka bilgileri Marka tablosundan okunur.

bağlantı kapatılır.

Kullanıcı ürün arama çubuğuna veri girdiğinde işlenecek kodlar.

```
            DataTable tablo = new DataTable();
            baglanti.Open();
            OleDbDataAdapter adtr = new OleDbDataAdapter("select *from Ürünler where
BarkodNumarası like '%" + txturunara.Text + "%'", baglanti);
```

Tablo datası oluşturulur.

Arama çubuğuna yazılan veri ile veri tabanındaki BarkodNumarası alan adı karşılaştırılır ve barkodnumarası olan ürün Ürünler tablosundaki tüm veri çağırılır.

```
            adtr.Fill(tablo);
            dataGridView1.DataSource = tablo;
            baglanti.Close();
        }
        private void btnsil_Click(object sender, EventArgs e)
        {
```

Tablo Listelenir

Listelenen tablo datagridview1 de gösterilir.

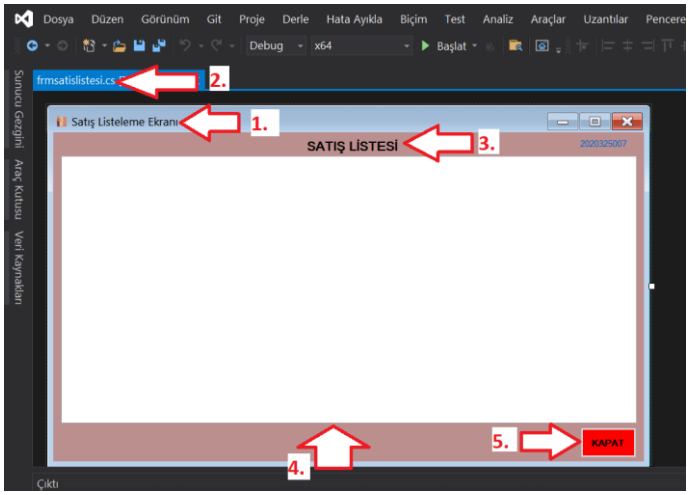
Kullanıcı Sil butonuna tıkladığında işlenecek kod.

```
            baglanti.Open();
            OleDbCommand komut = new OleDbCommand("delete from Ürünler where
BarkodNumarası='" + dataGridView1.CurrentRow.Cells["BarkodNumarası"].Value.ToString()
+ "'", baglanti);
```

Datagridviewde listesinden seçilen ürünün BarkodNumarası alanı baz alınarak, Ürünler tablosundaki verileri delete komutlarıyla silinir.

```
            komut.ExecuteNonQuery();
            baglanti.Close();
            daset.Tables["Ürünler"].Clear(); Tablo Temizlenir.
            urunlistele(); Ürünler tekrar listelenir.
            MessageBox.Show("Ürün Silme Başarılı.","ONAY");
    }
 }Son olarak Kullanıcıya silme işleminin başarılı olduğunu bildiren onay penceresi gösterilir.
}
```

Satış Listeleme Penceresi Görsel Tasarım



Resim 8

Resim 8 1. Satışların listelendiği pencerenin görünen metni. Son kullanıcı pencereyi bu isimle görecektir. Bu ad nesnenin değişmez adından farklıdır. Değişmez adı "frmsatislistesi" dir. Metinsel kodlama kısmında nesne bu değişmez ad ile çağırılacaktır.

Resim 8 2. Pencerenin tanımlanan değişmez adı.

Resim 8 3. Kullanıcıya satışların nerde listeleneceğini göstermek için Label

Resim 8 4. Satışların listeleneceği DataGridview.

Resim 8 5. Satış Listesi penceresini kapatıp ana ekrana dönmek için kapat butonu.

Satış Listeleme Penceresi Kodları

```
namespace StokTakip
{
    public partial class frmsatislistesi : Form
    {
        public frmsatislistesi()
        {
            InitializeComponent();
        }
        OleDbConnection baglanti = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.16.0;Data Source=" +
Application.StartupPath + "\\StokVerisi.accdb");
        DataSet daset = new DataSet();
        private void satislistele()
        { Satılan ürünleri listelemek için metod.
            baglanti.Open();
```

```
            OleDbDataAdapter adtr = new OleDbDataAdapter("select *from Satışlar",
baglanti);
```
Veri tabanından Satışlar tablosunun tüm içeriği seçilir.

```
            adtr.Fill(daset, "Satışlar");
            dataGridView1.DataSource = daset.Tables["Satışlar"];
```
Seçilen tablo listelenir.

Liste datagridview1 e eklenir.

```
            baglanti.Close();
        }
        private void frmsatislistesi_Load(object sender, EventArgs e)
        {
            satislistele();
        }
        private void btncikis1_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```
Satış listesi penceresi açıldığında çalıştırılacak kodlar.

Satışların listelendiği metod çağırılır.

Kullanıcı Kapat butonuna tıkladığında çalıştırılacak komut.

Pencere kapatılır.

4.5. Programı Kullanırken Dikkat Edilmesi Gereken Eksiklikler

1. Satış penceresinde, satılmak üzere seçilen ürün bilgileri ürün işlemleri alanından değiştirilirse Satış listesinde eklenen bilgiler yer alır.

2. Satış penceresinde Sepet listesindeki ürünlerden biri seçilmeden veya listedeki * (yıldız) sütunu seçilerek Sil butonuna tıklanırsa sepete eklenen son ürün silinir.

3. Ürün listeleme penceresinde listeden seçilerek Ürün işlemleri alanına getirilen kategori ve marka alanı değiştirilirse ürünün kategori ve markası listede değişir. Fakat ürün daha önce tanımlanan kategori ve markalara eklenmez.

4.6. Programın Çalışmasını Durduran Kritik Hatalar

1. Satış penceresinde satılacak ürün bilgileri girilmez ise ya da herhangi bir alan boş bırakılarak Sepete Ekle veya Sepetten Sil butonuna tıklanırsa program hata vererek çalışmaz ve yeniden başlatmak gerekir.

2. Satış penceresindeki Satış Adeti, Satış Fiyatı, Toplam Fiyat alanlarına sayı yerine metin girilirse program hata vererek çalışmaz ve yeniden başlatmak gerekir.

3. Ürün Ekleme penceresinde Adet, Alış Fiyatı, Satış Fiyatı alanlarına sayı yerine metin girilip Ekle butonuna tıklanırsa program hata vererek çalışmaz ve yeniden başlatmak gerekir.

4. Ürün Listeleme penceresi mevcut ürün alanında Adet, Alış Fiyatı, Satış Fiyatı alanlarına sayı yerine metin girilip Ekle butonuna tıklanırsa program hata vererek çalışmaz ve yeniden başlatmak gerekir.

SONUÇ

Bu çalışmada Görsel programlama ve C# programlama dili ile ilgili temel bilgiler, Visual Studio 2019 yazılımında görsel programlama öğeleri kullanarak yeni proje oluşturma, Access veri tabanı oluşturarak proje içinde veri tabanından veri çekme, okuma, güncelleme ve silme, Forms öğelerinde görsel tasarım yapma, C# programlama dilinde kod yazma pratikleri yapılarak küçük çaplı işletmeler için Stok Takip isimli stok ve satış takip programı geliştirilip bu alanın giriş seviyesi düzeyinde, yazılım geliştirme ve yazılıma veri tabanı entegre etme becerileri kazandırılmıştır. Proje bölümleri kısmında kodlanan projenin görsel tasarımı ve işlev kodlaması anlatıldığı gibi proje dosyası ve kurulum uygulaması olarak hazırlanmıştır. Kodlanan programın proje dosyası ve Smart Install Maker yazılımıyla hazırlanan, programın son kullanıcıya kurulumu için kurulum dosyası (Stok Takip Kurulum.exe), "2020325007 MustafaKALENDER.rar" dosyasının içerisine eklenmiştir. Programın kurulum talimatları ve sistem gereksinimleri kurulum penceresinde anlatılmıştır.

KAYNAKLAR

https://tr.wikipedia.org/wiki/Microsoft_Access - Microsoft access Veri tabanı

https://www.oracle.com/tr/database/what-is-database/ - Veri Tabanı

https://veriakademi.com/c-sharp-nedir - C#

https://visualstudio.microsoft.com/tr/vs/ - Visual Studio Nedir

https://tr.wikipedia.org/wiki/Veritabanı - Veri Tabanı