İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Ticari Otomasyon Sistemi

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Bitirme Projesi

Buse Ceyda EKREN

Y210240083

Proje Danışmanı: Dr. Öğr. Üyesi Serpil Yılmaz

Haziran 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Buse Ceyda EKREN** tarafından hazırlanan **Ticari Otomasyon Sistemi** başlıklı bu çalışma tarafımca okunmuş olup, kapsam ve nitelik açısından başarılı bulunarak tarafımdan YÜKSEK LİSANS BİTİRME PROJESİ olarak kabul edilmiştir.

ONAYLAYANLAR:

Bitirme Projesi Danışmanı: **Dr. Öğretim Üyesi Serpil Yılmaz**
İzmir Kâtip Çelebi Üniversitesi

Yazarlık Beyanı

Ben, **Buse Ceyda EKREN**, başlığı **Ticari Otomasyon Sistemi** olan bu bitirme projemin ve projenin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Tezsiz Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bitirme projesinin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Projemin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Bitirme projesinde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 11.06.2023

TİCARİ OTOMASYON SİSTEMİ

Öz

E-ticaret Otomasyona basit bir şekilde değinmek gerekirse, sürekli tekrarlanan işlerin bir kodlama yapısı ile dinamik hale getirilmesi yani bu işleri çevrimiçi sistemlere ve yazılımlara yaptırılması olarak ifade edilebilmektedir.

Otomasyon, teknoloji veya yapay zeka insan hatalarını minimum seviyeye indirgese bile e-ticaret gibi sistemleri komple yazılıma/sisteme bırakamayız bir insan kontrolü bu sistemlerde kaçınılmazdır. Fakat standart akışların kontrolü için otomasyonlara ihtiyaç duyulmaktadır. Otomasyonlar sayesinde, şirketler hem zaman kazanıp hem de eleman maliyetlerini de düşürebilmektedirler. Yine otomasyonlar sayesinde insandan kaynaklı olabilecek hataları da azaltabilmektedirler böylece şirketler işlerini çok daha hızlı ve kontrollü büyütebilmektedirler.

E-ticaret otomasyonları günümüzde pek çok alanda kullanılabilmektedir. Müşteri hizmetleri, envanter alma, depolama, teslimat ve daha birçok alanda artık otomasyonlar kullanılmaktadır.

Teknolojini gelişmesiyle ve e-ticarete olan yönelimin artması ile birlikte ticari otomasyonlar hayatımızın büyük bir parçası haline geldiler. Gün geçtikçe daha çok gelişip yeni özellikler kazandılar.

Bu çalışmada, sadece elektronik ürünlerin satımı ve takibini gerçekleştiren bir ticari otomasyon geliştirilmesi amaçlanmıştır. Kullanan firma, satım, personel, departman ve müşteri takiplerini kolaylıkla sağlayabilecektir.

Anahtar Sözcükler: Otomasyon, E-ticaret, Teknoloji, Ticari Otomasyon

COMMERCIAL AUTOMATION SYSTEM

Abstract

To put it simply, e-commerce Automation can be expressed as making repetitive works dynamic with a coding structure, that is, having these works done by online systems and software. Even if automation, technology or artificial intelligence minimizes human errors, e-commerce We cannot leave systems such as commerce to the complete software/system, a human control is inevitable in these systems. But automations are needed for the control of standard flows. Thanks to automation, companies can both save time and reduce personnel costs. Again, thanks to automation, they can reduce human-induced errors, so companies can grow their businesses much faster and in a controlled manner.
 E-commerce automations can be used in many areas today. Automation is now used in customer service, inventory taking, storage, delivery and many more areas.
With the development of technology and the increasing orientation to e-commerce, commercial automations have become a big part of our lives. Day by day, they developed more and gained new features.

In this study, it is aimed to develop a commercial automation that only sells and monitors electronic products. The user will be able to easily follow up the company, sales, personnel, department and customer.

Keywords: Automation, E-commerce, Technology, Commercial Automation

İçindekiler

Şekiller Listesi

Tablolar Listesi

Kısaltmalar Listesi

IDE	Integrated Development Environment
RDBMS	Relational Database Management System
MYSQL	My Structured Query Language
SQL	Structured Query Language
MSSQL	Microsoft Structured Query Language
ENIAC	Electronic Numerical Integrator And Computer
CLI	Command Line Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language

Semboller Listesi

//	Yorum Satırı
==	Denktir
>=	Küçük veya Eşittir

1. Giriş

Günümüzde otomasyon hemen hemen her sektörde baş göstermiş bir halde hayatımızda yer almaktadır. Mağazalarda, kütüphanelerde, ticarette, hastanelerde ve daha birçok yerde otomasyon sistemleri aktif bir biçimde kullanmaktadır. Otomasyonun hayatımıza bu kadar sağlam ve hızlı adapte olmasındaki en büyük etken verileri kolay bir şekilde depolayıp işleyebilmesidir Depolanan ve işlenen bu verilerin izlenebilirliği de bir o kadar kolaydır. Bununla birlikte zamandan çok ciddi bir tasarruf sağlanmaktadır. Ayrıca insan hatası da minimum seviyeye indirgenmektedir. Sistemde, süreçte hata var ise erken aşamalarda tespit edilir. Bu da hata maliyetini minimum seviyeye indirgemektedir. Bazı görüşler otomasyonu, teknolojiyi veya yapay zekayı savunmasa bile hayatımızdaki pozitif etkileri de göz ardı edilememektedir. Ancak otomasyonun bu kadar avantajına karşılık oldukça büyük bir dezavantajı da bulunmaktadır. Otomasyonun kurulum aşaması oldukça maliyetlidir.

Otomasyon sistemlerinde kullanılan programlama dillerine, araçlara değinecek olursak, birçok programlama dili ve araçla uyumlu çalışabilmekte ve internet üzerinde birden çok kaynağa erişim sağlanabilmektedir. Kullanılan programlama dillerine örnek verecek olursak; Java, C# ve C/C++ gibi programlama dillerini listeleyebiliriz. Ancak bizim Ticari Satış Otomasyon'unda kullanacağımız programlama dili C#asp.net olacaktır. C# dilinin avantajlarına değinecek olursak kullanımı ve kaynağı oldukça fazladır. Beraberinde cevap verebildiği soru çeşidi de diğer programlama dillerine kıyasla oldukça fazladır. Günümüzde otomasyon, web, oyun programlama gibi birçok alanda kullanılmaktadır. Ve piyasa üzerinde popular olan araçlarla da entegrasyonu bulunmaktadır. Fakat bir eksi özelliği ise diğer programlama dillerine göre biraz yavaş kalmaktadır. Ayrıca yapılan her değişiklikten sonra derleme işlemine ihityaç duymaktadır [1].

C#asp.net 'e gelecek olursak, HTML, CSS ve JavaScript kullanarak web siteleri ve web uygulamaları oluşturmaya yönelik ücretsiz bir web çerçevesidir. Ayrıca Web

API'leri oluşturabilir ve Web Yuvaları gibi gerçek zamanlı teknolojilerin kullanılmasına olanak sağlayabilmektedir.

Ticari Satış Otomasyon'unda kullanılacak bir diğer teknoloji ise Visual Studio'dur. Kodları yazıp derlediğimiz ve çıktıları, hataları gözlemlediğimiz bir platformdur. Kısaca Visual Studio teknolojisine değinmek gerekirse; birçok programlama dilini desteklemektedir. Ve kulanıcıların masaüstü veya web uygulamaları geliştirmesine olanak sağlamaktadır. Bununla beraber oyun tasarımlarında da Unity ve C# ile entegre çalışabilmektedir. Kullanımı olduçta kolaydır ve piyasada oldukça fazla kaynağa sahiptir. Kod tamamlama ve kodu yeniden düzenlemeyi destekleyen bileşenleri de mevcuttur. Kullanıcıya piyasada aktif olarak kullanılan programla dilleri ile geliştirme yapma imkanı sunar. Bu dillere örnek verecek olursak; C, C++, C#, CLI, VB.NET, Ruby, Node.js, Python gibi programlama dilleri örnek verilebilir.

Son olarak Ticari Otomasyon Sisteminde kullanılan veritabanına değinecek olursak; MSSQL tercih edilmiştir. Veritabanını veriler için bir depolama alanı olarak düşünebiliriz. Veritabanını; verilerin silinebildiği, güncellenebildiği, depoya yeni verilerin eklenebildiği bir sistem olarak tanımlayabiliriz. Otomasyon sistemlerinde sık sık karşımıza çıkan veritabanlarına örnek olarak; Oracle, Microsoft SQL Server, MySQL, Informix ve Interbase gibi veritabanları verilebilir. Yukarıda da bahsedildiği gibi bu otomasyon sisteminde kullanılan veritabanı MSSQL 'dir. MSSQL'e kısa bir bakış atacak olursak, işletmeler için kritik önem taşıyan verileri şifreleme, verilere erişim sağlayan kişileri gözlemleme ve erişim kısıtlamaları tanımlama gibi güvenlik özellikleri sayesinde kullanıcılara kapsamlı bir denetim kapasitesi sunmaktadır. Veri işleme, depolama, indeksleme, sorgulama, raporlama, veritabanı yönetimi, veri akışı optimizasyonu gibi birçok işlemi yapmamıza olanak sağlamaktadır [2].

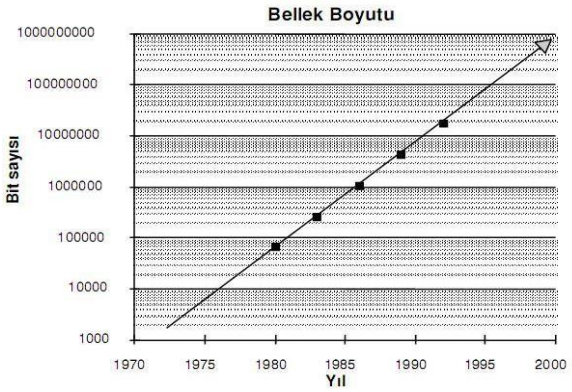
2. Ticari Otomasyon Sistemi

2.1 Bilgisayarın Gelişimi

Teknolojinin ilerleyişi ile birlikte bilgisayarda çağımızın vazgeçilmezleri arasında yerini aldı. Hemen hemen hayatımızın her alanında bilgisayarı aktif olarak kullanıyor ve geliştirmeye devam ediyoruz. Bilgisayarın tarihçesini irdeleyecek olursak günümüz bilgisayarlarının temelleri ilk olarak 1945 senesinde ENIAC ismi ile atıldı. İnsan boyundan bile daha uzun ve 30 tandan daha ağır bir yapıda tasarlanan bu bilgisayar top atışlarının hesaplanmasında kullanılırdı.

1965 yılında ise daha modern, günümüz bilgisayarlarının üretimlerine başlandı. Üretilen bu bilgisayarlar daha küçük ve hafif bir yapıya bürünmeye başladı. Bellek boyutu, klavye, mouse ve birçok bilgisayar parçası da küçüldü ve daha estetik bir yapıya büründü. Günümüz örgütlerinde artan karmaşıklaşma, yönetim için ihtiyaç duyulan bilginin gün geçtikçe artmasına neden olmuştur. Bilginin toplanması, işlenmesi, saklanması ve tekrar bilgi olarak sunulması sürekli gelişen bilgi teknolojisinin gereğidir. Yazılım, donanım ve iletişim teknolojilerini kapsayan bu teknoloji, günümüz örgütlerini 21. yüzyıla hızla taşımıştır [3].

Aşağıdaki grafikte 1970-2000 yılları arasındaki bellek teknolojisindeki gelişmeler yansıtılmaktadır.



yıl	boyut (Mbit)
1980	0,0625
1983	0,25
1986	1
1989	4
1992	16
1996	64
1998	128
2000	256
2002	512

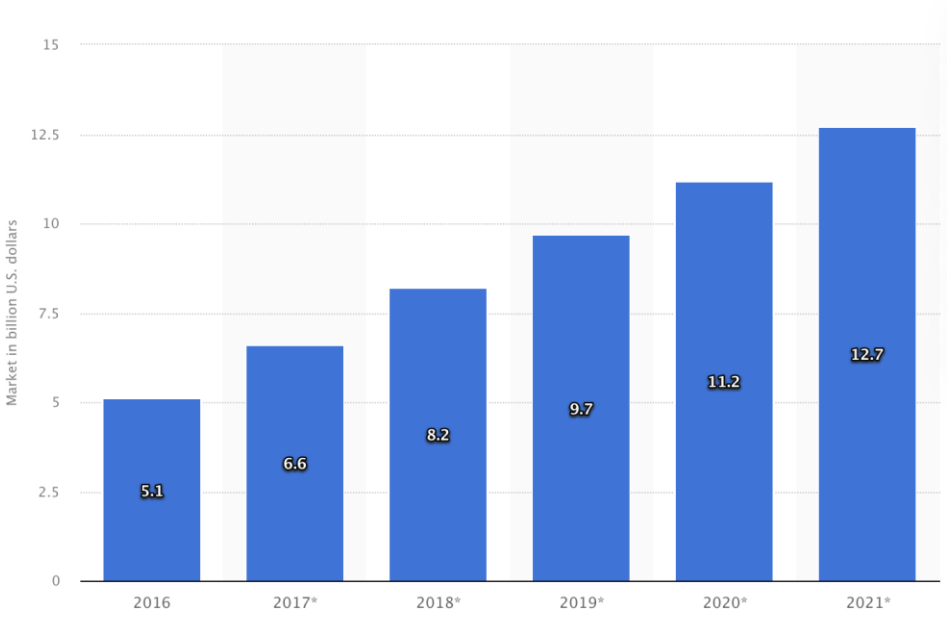
Tablo 1: 1970-2000 yılları arasında bellek teknolojisindeki gelişmeler [4]

2.2 Otomasyon Nedir ve Otomasyonun Ticari Hayat Üzerindeki Etkisi Nedir?

Otomasyon, belirli bir sürecin veya görevin insan müdahalesi olmadan otomatik olarak gerçekleştirilmesi anlamına gelmektedir. Birçok sektörde otomasyon teknolojileri kullanılarak tekrarlayan, zaman alıcı veya monoton işlerin otomatikleştirilmesi hedeflenmektedir. Bu sayede verimlilik artar, hata oranı azalır ve insanların daha değerli görevlere odaklanması sağlanılır. Günümüzde birçok alanda otomasyonun izlerine rastlayabilmekteyiz; ulaşım, sağlık, eğitim, ticaret, güvenlik vb. gibi birçok alanda hayatımızın vazgeçilmezleri arasında yerini almaktadır.

Ticari otomasyonlara ve hayatımızdaki etkilerine değinecek olursak; Otomasyonun ticari hayata etkisi oldukça önemli ve geniş kapsamlıdır. İşletmeler, otomasyon teknolojilerini kullanarak birçok süreci otomatikleştirerek verimlilik, maliyet tasarrufu ve rekabet gücü avantajı elde etmektedirler. Ticari hayatta otomasyon; verimlilik artışı sağlar, tekrar gerektiren ve zaman alan işleri insan müdahalesi olmadan gerçekleştirir. Maliyet tasarrufu sağlar, işletmeler otomasyon sayesinde iş gücü maliyetlerini minimum seviyeye indirgeyebilirler. Hızlı ve esnek üretim imkanı sağlar, otomatik sistemler süreçleri kesintisiz ve kolay bir şekilde yürütebilmektedirler [5].

Aşağıdaki grafikte 2016-2021 yılları arasında dünya çağında iş süreci otomasyon pazarının artış grafiği gösterilmektedir.



Tablo 2: 2016-2021 otomasyon pazarının boyut artışı [6]

2.3 Ticari Otomasyon Sistemi Nedir?

Ticari Otomasyon Sistemi, temelde her türlü elektronik ürünün satım ve takip işlemini gerçekleştiren bir otomasyon sistemidir. İçerisinde kategoriler, ürünler, departmanlar, cariler, personeller, admin, faturalar, fatura ve kalemler, giderler, hareketler ve istatistikler gibi birçok işlemi gerçekleştirmek için sekmeler bulunmaktadır. Ayrıca sayfa üzerinde direkt istenen bilgiye gidebilmek adına bir search box da bulunmaktadır.

Kategoriler menüsünde kullanıcı ilgili ürünlerin kategorilerini görebilir, bu kategoriler altında silme ve güncelleme işlemleri yapabilir. Ayrıca kullanıcı yeni bir ürün kategorisi de oluşturabilir.

Ürünler sekmesinde kullanıcı, daha önceden eklenmiş ürünleri listeleyebilir/güncelleyebilir/silebilir veya belirli bir kategori altına yeni bir ürün ekleyebilir.

Departmanlar sekmesinde ise kullanıcı o firmaya ait departmanları listeleyebilir ve o departmana ait çalışanları, personel numaraları gibi detayları görüntüleyebilir.

Bununla beraber kullanıcı yeni bir departman oluşturabilir veya var olan departmanlar üzerinden güncellemeler yapabilir.

Cariler menüsünde ise kullanıcı, müşterilerini listeleyebilir. Daha öncesinde satış yapmış oldukları müşteriler üzerinde güncellemeler yapabilir veya satış yapacağı yeni müşteri bilgilerini sisteme ekleyebilir.

Personeller sayfasında ise kullanıcı, bünyesinde bulunan personellerin takibini bu sekmeden yapabilir. Personellerin bilgilerini güncelleyebilir veya artık bünyesinde bulunmayan personel bilgilerini sistem üzerinden kaldırabilir. Eğer bünyesine yeni bir personel almış ise buradan yeni personel bilgilerini de ekleyebilir.

Admin sayfasında ise, bu otomasyon sistemindeki en yüksek erişim düzeyine sahip rolün kullanabileceği bir sekmedir. Buradan yönetici araç çubuğundaki tüm ögelere erişilebilir veya sistem üzerinde bir değişiklik yapılabilir. Bu sekmeye her kullanıcının erişimi bulunmamaktadır.

Faturalar kısmında ise, satılan ürünlerin faturaları burada tutulur. İstenen faturalar buradan kaldırılabilir. Yeni ürün satışı gerçekleştikçe ürünün faturası faturalar sekmesine düşecektir.

Giderler kısmında ise o ayın toplam giderleri, harcamaları görüntülenebilecek ve bu gelir/gider oranı istatistikler kısmında toplu grafiklerle takip edilebilecek [7].

3. Ticari Satış Otomasyonunda Kullanılan Teknolojiler

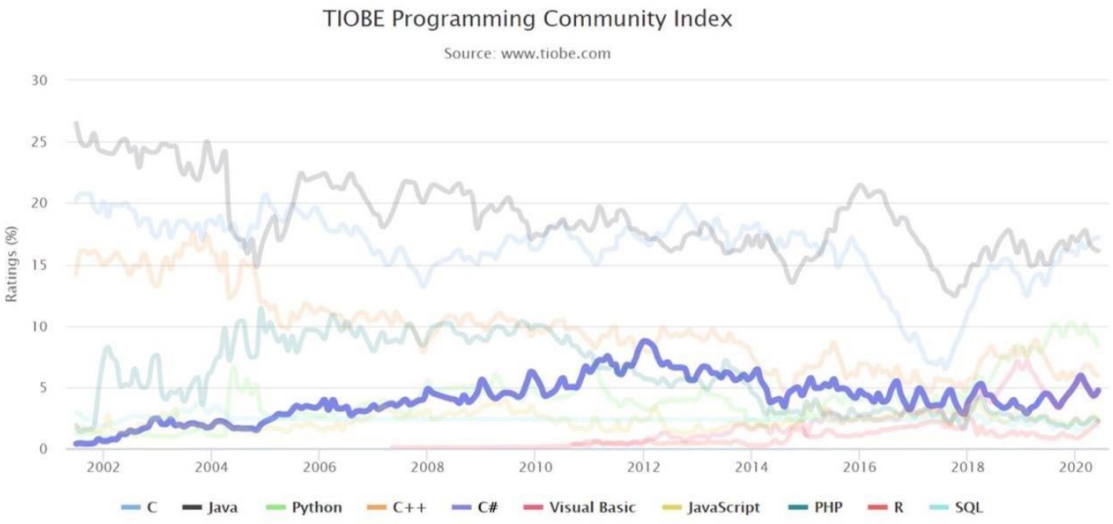
Ticari Satış Otomasyonu yapilirken bircok farkli arac ve teknolojilerden faydalanildi. Program Visual Studio uzerinden C#asp.net programlama dili ile yazildi. Veri tabani olarak Microsoft SQL Server tercih edildi.

3.1 C# Asp.net Programlama Dili

Microsoft tarafından geliştirilen, modern ve nesne yönelimli bir programlama dilidir. Günümüzde C# programcılar tarafından sıkça tercih edilen bir programlama dilidir. Bunun temel sebeplerini sıralayacak olursak; öğrenimi oldukça kolay, internet üzerinde oldukça fazla kaynağı bulunmakta, birçok kod yazma aracı ile uyumlu çalışabilir, C#, derlenen bir dil olduğu için yüksek performanslı uygulamalar geliştirmek için uygundur. Ancak biraz yavaş bir proglamlara dilidir. Çok hızlı çalışma gerektiren uygulamalarda kullanımı uygun değildir.

C# ASP.NET, Microsoft tarafından geliştirilen bir web uygulama çatısıdır. ASP.NET, C# programlama dili ve .NET Framework veya .NET Core platformlarıyla birlikte kullanılarak dinamik ve etkileşimli web siteleri, web uygulamaları ve web hizmetleri geliştirmek için kullanılır. ASP.NET, geliştiricilere zengin bir geliştirme ortamı ve araç seti sunar. Visual Studio IDE (Integrated Development Environment) gibi araçlar, kolay ve verimli bir şekilde ASP.NET uygulamaları geliştirmeyi destekler. Genel olarak, ASP.NET, C# programlama dilinin web uygulamaları için kullanıldığı güçlü bir web çatısıdır [8].

Aşağıdaki grafikte 2002-2020 yılları arsında C# programlama dilinin kullanım oranı paylaşılmıştır.



Tablo 3: 2002-2020 yılları arasında C# kullanım oranı [9]

3.2 Microsoft Visual Studio

Ticari Satış Otomasyon'unda kullanilan bir digger teknoloji de Microsoft Visual Studiodur [10]. Microsoft Visual Studio, Microsoft tarafından geliştirilen bir entegre geliştirme ortamıdır (Integrated Development Environment - IDE). Yazılım geliştiricilerin uygulama oluşturma, hata ayıklama, derleme, dağıtım ve diğer geliştirme süreçlerini yönetmelerine yardımcı olan kapsamlı bir araç setini içerir. Visual Studio, genellikle C#, C++, Visual Basic, F# ve diğer Microsoft teknolojileriyle birlikte kullanılan bir IDE olarak bilinir. Bununla birlikte, çeşitli programlama dillerini ve platformları destekler. Web uygulamaları, masaüstü uygulamaları, mobil uygulamalar, oyun geliştirme ve veri analizi gibi çeşitli yazılım projelerini geliştirmek için kullanılabilir [11].

3.3 Microsoft SQL Server

Microsoft SQL Server, Microsoft tarafından geliştirilen ve yönetilen ilişkisel veritabanı yönetim sistemidir (RDBMS). SQL Server, verilerin depolanması, yönetilmesi, sorgulanması ve güvenli bir şekilde erişilmesi için kullanılan güçlü bir veritabanı platformudur. SQL Server, işletmelerin veri tabanlı uygulamalarını

desteklemek için geniş bir özellik yelpazesi sunar. Veri işleme, depolama, güncelleme, veri takibi, veri raporlama gibi imkanları kullanıcıya sunar. Bununla birlikte kullanıcıya yedekleme ve kurtarma mekanizması da sunmaktadır [12].

4. Ticari Otomasyon Sistemi İçeriği

Ticari Satış Otomasyonu projesinde aşağıda belirtilen Kategori sayfasında; yeni kategoriler oluşturulabilir, var olan kategorilerde slime veya güncelleme işlemleri yapılabilmektedir.

4.1 Kategoriler

```
        public ActionResult KategoriGetir(int id)
        {
            var kategori = c.Kategoris.Find(id);
            return View("KategoriGetir", kategori);
        }
```

```
@using MVCOnlineTicariOtomasyon.Models.Siniflar
@model List<Kategori>
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}

<table class="table table-bordered" style="margin-top:20px;">

    <tr>
        <th>ID</th>
```

```
        <th>Kategori Adı</th>
        <th>Sil</th>
        <th>Güncelle</th>
    </tr>

    @foreach (var k in Model)
    {
        <tr>
            <td>
                @k.KategoriID
            </td>
            <td>
                @k.KategoriAd
            </td>
            <td><a href="/Kategori/KategoriSil/@k.KategoriID" class="btn btn-danger">Sil</a></td>
              <td><a href="/Kategori/KategoriGetir/@k.KategoriID" class="btn btn-success">Güncelle</a></td>
        </tr>
    }
</table>
<a href="/Kategori/KategoriEkle" class="btn btn-primary">Kategori Ekle</a>
```

Kategori güncelleme için oluşturduğum sayfanın düzenleme işlemlerini gerçekleştirdim. Bu sayfa da güncellemek istediğim kategorinin güncelle butonuna tıklatıldıktan sonra gözükecek sayfa da kategorinin hangi değerlerinin gözükeceğini belirledim.

```
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Kategori
@{
    ViewBag.Title = "KategoriGetir";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
```

```
<br/>
<h2>Kategori Güncelleme</h2>
<br/>
@using (Html.BeginForm("KategoriGuncelle","Kategori",FormMethod.Post))
{
    <div class="form-group">
        @Html.LabelFor(x => x.KategoriAd)
        @Html.TextBoxFor(x => x.KategoriAd, new { @class = "form-control" })
    </div>
    <button class="btn btn-warning">Güncelle</button>

}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCOnlineTicariOtomasyon.Models.Siniflar;
namespace MVCOnlineTicariOtomasyon.Controllers
{
    public class KategoriController : Controller
    {
        // GET: Kategori
        Context c = new Context();
        public ActionResult Index()
        {

            var degerler = c.Kategoris.ToList();
            return View(degerler);
```

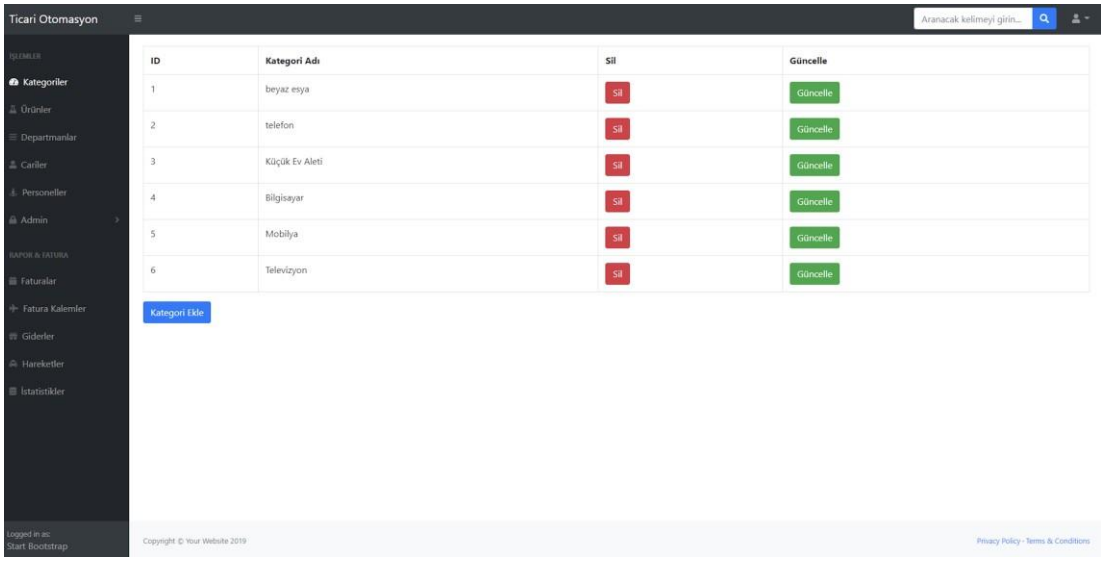
```
}

[HttpGet]
public ActionResult KategoriEkle()
{
    return View();
}

[HttpPost]
public ActionResult KategoriEkle(Kategori k)
{
    c.Kategoris.Add(k);
    c.SaveChanges();
    return RedirectToAction("Index");

}

public ActionResult KategoriSil(int id)
{
    var ktg = c.Kategoris.Find(id);
    c.Kategoris.Remove(ktg);
    c.SaveChanges();
    return RedirectToAction("Index");
}

public ActionResult KategoriGetir(int id)
{
    var kategori = c.Kategoris.Find(id);
    return View("KategoriGetir", kategori);
}
```

```
        public ActionResult KategoriGuncelle (Kategori k)
        {
            var ktgr = c.Kategoris.Find(k.KategoriID);
            ktgr.KategoriAd = k.KategoriAd;
            c.SaveChanges();
            return RedirectToAction("Index");
        }




    }
}
```

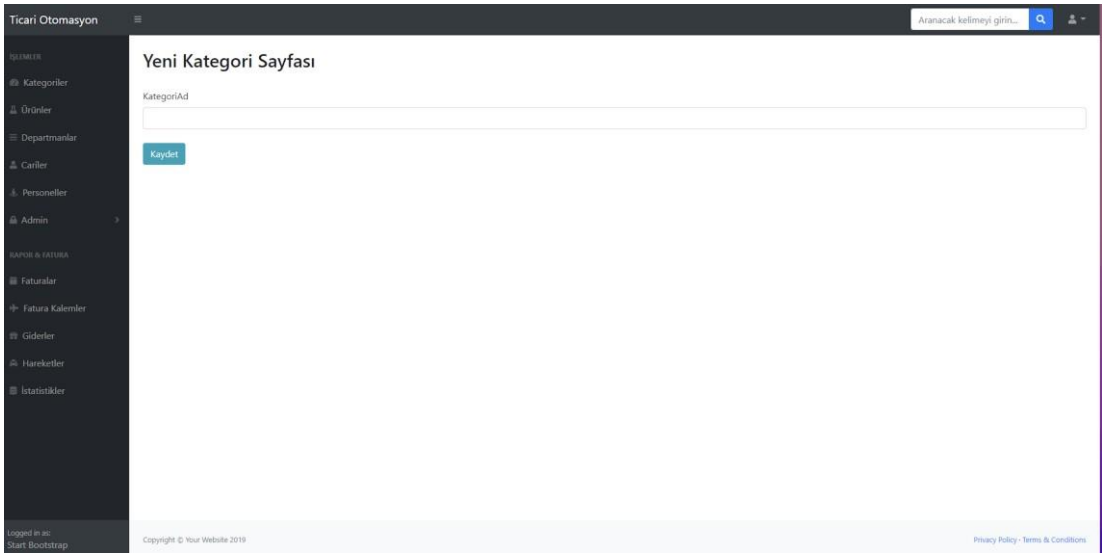
Son olarak kategori getir sayasında kategori id lerini de belirleyip kategori alanını tamamlamış oldum.

```
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Kategori
@{
    ViewBag.Title = "KategoriGetir";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}

<br/>
<h2>Kategori Güncelleme</h2>
<br/>
@using (Html.BeginForm("KategoriGuncelle","Kategori",FormMethod.Post))
{
    <div class="form-group">
        @Html.LabelFor(x => x.KategoriAd)
        @Html.TextBoxFor(x => x.KategoriAd, new { @class = "form-control" })
        @Html.HiddenFor(x => x.KategoriID)
```

```
    </div>
    <button class="btn btn-warning">Güncelle</button>

}
```

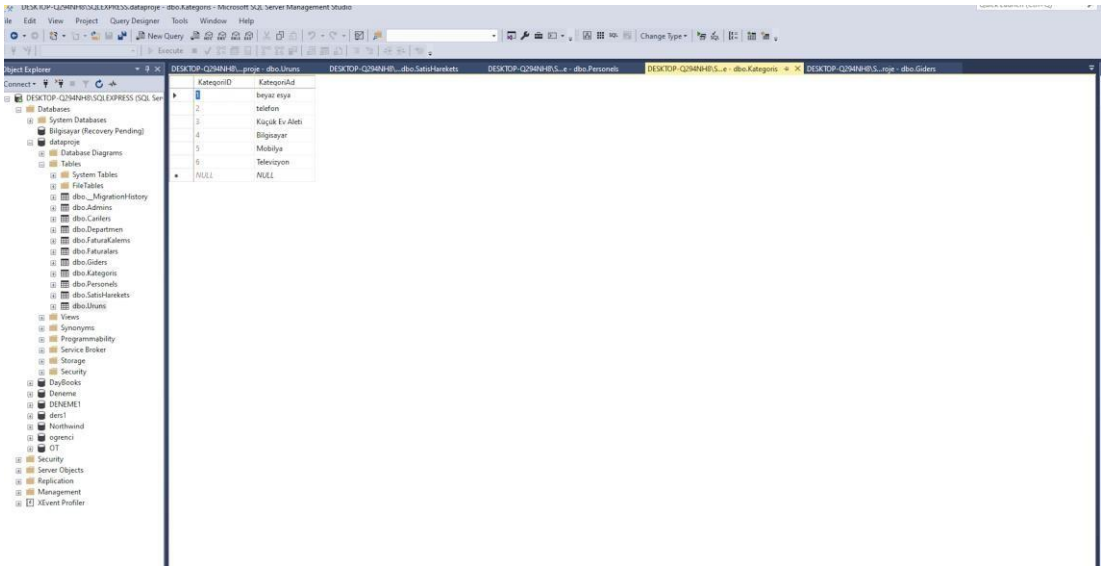


Şekil 1: Kategoriler



Şekil 1.2: Kategorileri Ekle

Veritabanında katagoriler tablosu bulunmaktadır. Otomasyon üzerinden yapılan kategori ekleme,slime,güncelleme gibi işlemler aktif olarak veritabanından da görüntülenebilmketedir.



Şekil 1.3: Kategoriler Veri Tabanı Tablosu

4.2 Ürünler

Kullanıcı ürünler sekmesinden yeni ürün ekleyebilecek veya var olan ürünler üzerinden güncelleme veya slime işlemlerini gerçekleştirebilecekler.

Öncelikle ürünler sayfası için controller oluşturuldu.

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCOnlineTicariOtomasyon.Models.Siniflar;
namespace MVCOnlineTicariOtomasyon.Controllers
{
```

```
    public class UrunController : Controller
    {
        // GET: Urun

        Context c = new Context();
        public ActionResult Index()
        {
            var urunler = c.Uruns.ToList();
            return View(urunler);
        }
    }
}
```

Daha sonrasında bu controllerlara bağlı viewlar eklendi.Bu view ürün sayfasını temsil etmektedir. Bu ürün indexi içinde uruna ait id, ad, marka, stok, alış fiyatı, satış fiyatı, kategori adı ve ürün görseli için tablo oluşturuldu.

```
@using MVCOnlineTicariOtomasyon.Models.Siniflar;
@model List<Urun>
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
<br/>

<table class="table table-bordered">
    <tr>
        <th>ID</th>
        <th>ÜRÜN</th>
        <th>MARKA</th>
        <th>STOK</th>
        <th>ALIŞ FİYAT</th>
```

```
            <th>SATIŞ FİYAT</th>
            <th>KATEGORİ</th>
            <th>GÖRSEL</th>
        </tr>
        @foreach(var u in Model)
        {
            <tr>
                <td>@u.Urunid</td>
                <td>@u.UrunAd</td>
                <td>@u.Marka</td>
                <td>@u.Stok</td>
                <td>@u.AlisFiyat</td>
                <td>@u.SatisFiyat</td>
                <td>@u.Kategori.KategoriAd</td>
                <td>@u.UrunGorsel</td>
            </tr>
        }
</table>
```

Ürün eklemek için, UrunController'ına YeniUrun adlı bir actionresult oluşturuldu. Bu actiona bir view eklendi. Bu view web sayfasında yeni ürün butonuna tıklatıldığında gösterilecek sayfayı temsil etmektedir.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCOnlineTicariOtomasyon.Models.Siniflar;
namespace MVCOnlineTicariOtomasyon.Controllers
{
    public class UrunController : Controller
```

```
    {
        // GET: Urun

        Context c = new Context();
        public ActionResult Index()
        {
            var urunler = c.Uruns.ToList();
            return View(urunler);
        }

        [HttpGet]
        public ActionResult YeniUrun()
        {
            return View();
        }
        [HttpPost]
        public ActionResult YeniUrun(Urun p)
        {
            c.Uruns.Add(p);
            c.SaveChanges();
            return RedirectToAction("Index");

        }
    }
}
```

Ürün ekleme işlemi için ındex içinde buton tanımlaması yaptım.

```
...
<a href="/Urun/YeniUrun/" class="btn btn-info">Yeni Ürün</a>
...
```

Daha sonra yeni ürün ekleme için oluşturduğum sayfa üzerinde düzenlemeler yaptım.

```
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Urun

@{

    ViewBag.Title = "YeniUrun";

    Layout = "~/Views/Shared/AdminLayout.cshtml";

}

<br/>

<h2>Yeni Ürün Sayfası</h2>

<br/>

<form class="form-group" method="post">

    @Html.LabelFor(x => x.UrunAd)

    @Html.TextBoxFor(x => x.UrunAd, new {@class="form-control" })

</form>
```

```
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Urun

@{

    ViewBag.Title = "YeniUrun";

    Layout = "~/Views/Shared/AdminLayout.cshtml";

}

<br/>

<h2>Yeni Ürün Sayfası</h2>

<br/>

<form class="form-group" method="post">

    @Html.LabelFor(x => x.UrunAd)

    @Html.TextBoxFor(x => x.UrunAd, new { @class = "form-control" })

    <br />

    @Html.LabelFor(x => x.Marka)

    @Html.TextBoxFor(x => x.Marka, new { @class = "form-control" })

    <br />

    @Html.LabelFor(x => x.Stok)
```

```
    @Html.TextBoxFor(x => x.Stok, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.AlisFiyat)
    @Html.TextBoxFor(x => x.AlisFiyat, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.SatisFiyat)
    @Html.TextBoxFor(x => x.SatisFiyat, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Kategori)
    @Html.TextBoxFor(x => x.Kategoriid, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.UrunGorsel)
    @Html.TextBoxFor(x => x.UrunGorsel, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Durum)
    @Html.TextBoxFor(x => x.Durum, new { @class = "form-control" })
    <br />
    <button class="btn btn-primary"> Kaydet</button>
</form>
```

Ürün slime işlemleri için şöyle bir yapı oluşturuldu; Bu silme işleminde sil butonuna tıklatıldığında ürünler silinmiyor onun yerine bu ürünlerin durum değerleri false olarak değiştiriliyor. Böylece ürünün güncellenmesi durumunda da bu işlem true yapılarak kolayca güncel ürün durumu belirlenmiş olabilir hale geliyor.Öncelikle silme işlemi için UrunController'ı içinde UrunSil adında bir aciton oluşturuldu.

```
public ActionResult UrunSil(int id)
        {
            var deger = c.Uruns.Find(id);
            deger.Durum = false;
            c.SaveChanges();
            return RedirectToAction("Index");
```

```
@using MVCOnlineTicariOtomasyon.Models.Siniflar;
@model List<Urun>
@{
    ViewBag.Title = “Index”;
    Layout = “~/Views/Shared/AdminLayout.cshtml”;
}
<br/>

<table class=”table table-bordered”>
    <tr>
        <th>ID</th>
        <th>ÜRÜN</th>
        <th>MARKA</th>
        <th>STOK</th>
        <th>ALIŞ FİYAT</th>
        <th>SATIŞ FİYAT</th>
        <th>KATEGORİ</th>
        <th>GÖRSEL</th>
        <th>SİL</th>
    </tr>

    @foreach(var u in Model)
    {
        <tr>
            <td>@u.Urunid</td>
            <td>@u.UrunAd</td>
            <td>@u.Marka</td>
            <td>@u.Stok</td>
            <td>@u.AlisFiyat</td>
            <td>@u.SatisFiyat</td>
```

```
            <td>@u.Kategori.KategoriAd</td>
            <td>@u.UrunGorsel</td>
            <td>
                <a href="/Urun/UrunSil/@u.Urunid" class="btn btn-danger">Sil</a>
            </td>
        </tr>
    }
</table>
<a href="/Urun/YeniUrun/" class="btn btn-info">Yeni Ürün</a>
```

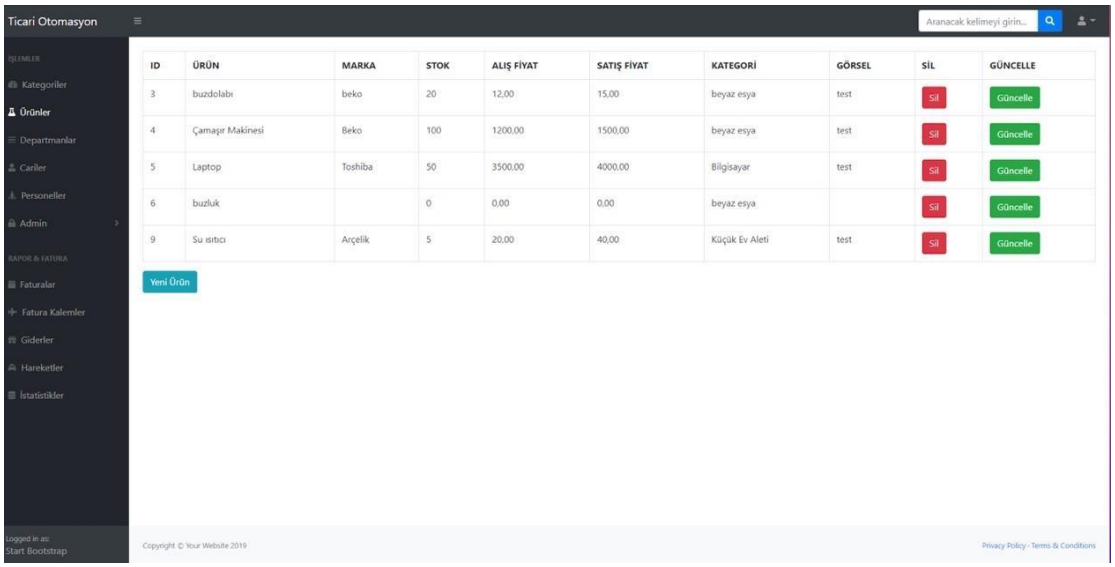
Ürün güncelleme sayfası içinse;

```
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Urun
@{
    ViewBag.Title = "UrunGetir";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
<br />
<h2>Ürün Güncelleme Sayfası</h2>
<br />

@using (Html.BeginForm("UrunGuncelle", "Urun", FormMethod.Post))
{
    @Html.LabelFor(x => x.UrunAd)
    @Html.TextBoxFor(x => x.UrunAd, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Marka)
    @Html.TextBoxFor(x => x.Marka, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Stok)
    @Html.TextBoxFor(x => x.Stok, new { @class = "form-control" })
    <br />
```

```
    @Html.LabelFor(x => x.AlisFiyat)

    @Html.TextBoxFor(x => x.AlisFiyat, new { @class = "form-control" })

    <br />

    @Html.LabelFor(x => x.SatisFiyat)

    @Html.TextBoxFor(x => x.SatisFiyat, new { @class = "form-control" })

    <br />

    @Html.LabelFor(x => x.Kategori.KategoriID)

    @Html.DropDownListFor(x                    =>                    x.Kategoriid,
(List<SelectListItem>)ViewBag.dgr1, new { @class = "form-control" })

    @*@Html.TextBoxFor(x => x.Kategoriid, new { @class = "form-control" })*@

    <br />

    @Html.LabelFor(x => x.UrunGorsel)

    @Html.TextBoxFor(x => x.UrunGorsel, new { @class = "form-control" })

    <br />

    @Html.LabelFor(x => x.Durum)

    @Html.TextBoxFor(x => x.Durum, new { @class = "form-control" })

    <br />

    <button class="btn btn-warning"> Güncelle</button>

}

      public ActionResult UrunGuncelle(Urun p)

        {

            var urn = c.Uruns.Find(p.Urunid);

            urn.AlisFiyat = p.AlisFiyat;

            urn.Durum = p.Durum;

            urn.Kategoriid = p.Kategoriid;

            urn.Marka = p.Marka;

            urn.SatisFiyat = p.SatisFiyat;

            urn.Stok = p.Stok;

            urn.UrunAd = p.UrunAd;

            urn.UrunGorsel = p.UrunGorsel;

            c.SaveChanges();
```

```
            return RedirectToAction("Index");
        }
<br />
<h2>Ürün Güncelleme Sayfası</h2>
<br />

@using (Html.BeginForm("UrunGuncelle", "Urun", FormMethod.Post))
{
    @Html.HiddenFor(x => x.Urunid)
    @Html.LabelFor(x => x.UrunAd)
    @Html.TextBoxFor(x => x.UrunAd, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Marka)
    @Html.TextBoxFor(x => x.Marka, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Stok)
    @Html.TextBoxFor(x => x.Stok, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.AlisFiyat)
    @Html.TextBoxFor(x => x.AlisFiyat, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.SatisFiyat)
    @Html.TextBoxFor(x => x.SatisFiyat, new { @class = "form-control" })
    <br />
    @Html.LabelFor(x => x.Kategori.KategoriID)
    @Html.DropDownListFor(x                    =>                    x.Kategoriid,
(List<SelectListItem>)ViewBag.dgr1, new { @class = "form-control" })
    @*@Html.TextBoxFor(x => x.Kategoriid, new { @class = "form-control" })*@
    <br />
    @Html.LabelFor(x => x.UrunGorsel)
    @Html.TextBoxFor(x => x.UrunGorsel, new { @class = "form-control" })
```

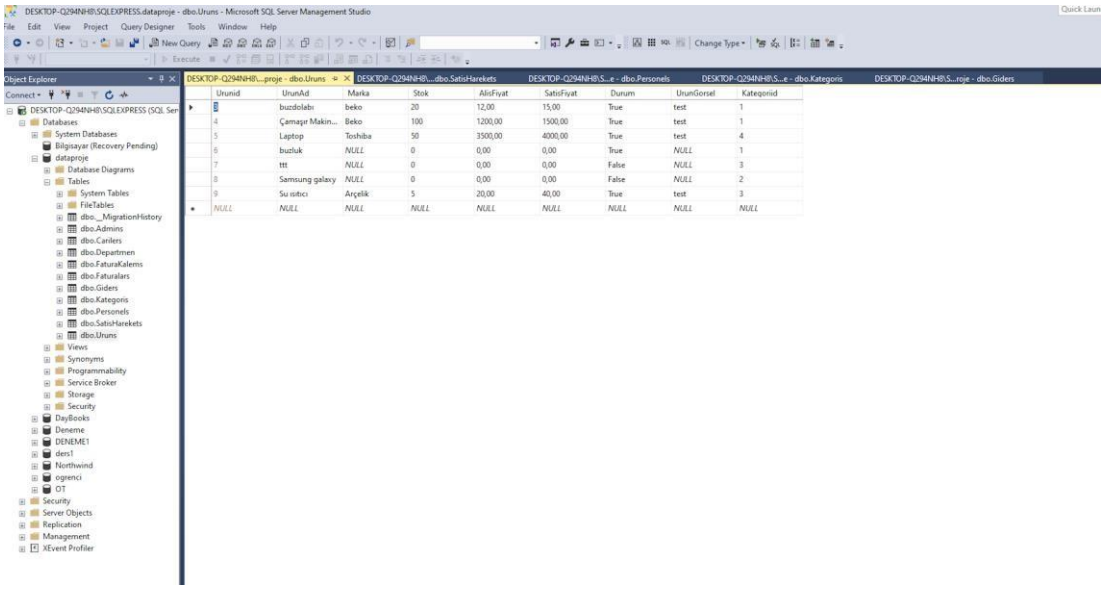
```
    <br />
    @Html.LabelFor(x => x.Durum)
    @Html.TextBoxFor(x => x.Durum, new { @class = "form-control" })
    <br />
    <button class="btn btn-warning"> Güncelle</button>
}
```



Şekil 2: Ürünler

Şekil 2.1: Ürün Ekle

Proje veritabanında bulunan ürünler tablosuna otomasyon üzerinden yapılan ürün ekleme,güncelleme,silme işlemleri aktif olarak yansımaktadır.



Şekil 2.2: Ürün Veri Tabanı Tablosu

4.3 Departmanlar

Bu bölümde kullanıcı firmaya ait departmanları listeleyebilir ve departman içerisinde çalışan personel bilgilerine ulaşabilmektedir. Sisteme yeni departmanlar eklenebilir veya var olan departmanlara güncelleme, silme işlemleri yapılabilmektedir.

```
@using MVCOnlineTicariOtomasyon.Models.Siniflar

@model List<Departman>

@{

    ViewBag.Title = "Index";

    Layout = "~/Views/Shared/AdminLayout.cshtml";

}

<table class="table table-bordered" style="margin-top:20px;">

    <tr>

        <th>ID</th>

        <th>Departman Adı</th>
```

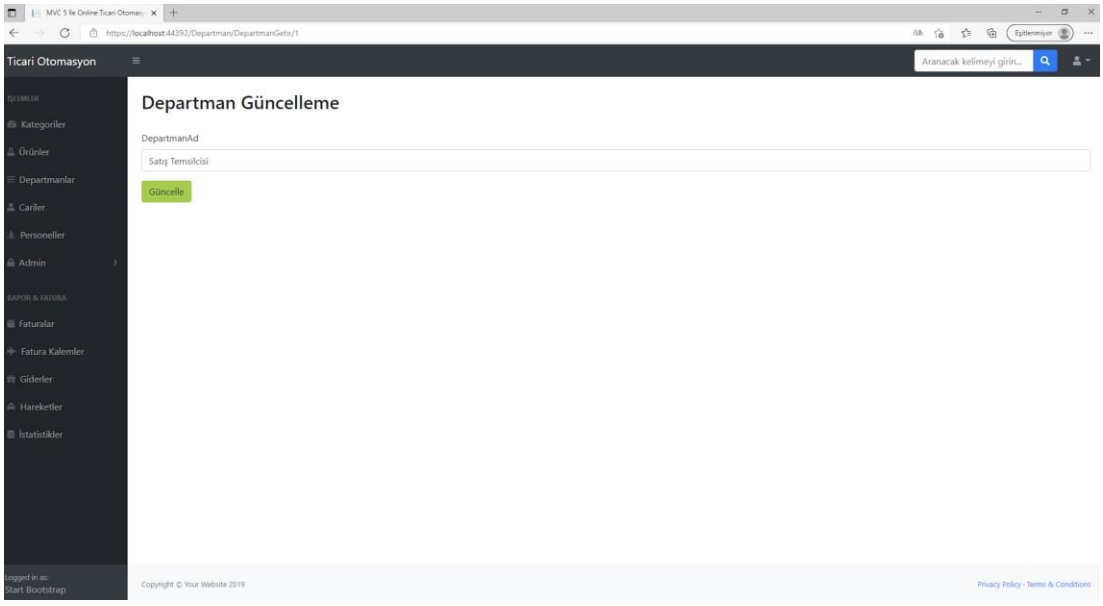
```
            <th>Sil</th>
            <th>Güncelle</th>
            <th>Detaylar</th>
        </tr>
        @foreach (var k in Model)
        {
<tr>
        <td>
            @k.Departmanid
        </td>
        <td>
            @k.DepartmanAd
        </td>
        <td><a   href="/Departman/DepartmanSil/@k.Departmanid"   class="btn   btn-danger">Sil</a></td>
        <td><a   href="/Departman/DepartmanSil/@k.Departmanid"   class="btn   btn-success">Güncelle</a></td>
        <td><a   href="/Departman/DepartmanDetay/@k.Departmanid"   class="btn   btn-default" style="background-color:lightcoral; color:white">Detaylar </a></td>
</tr>
        }
</table>
<a href="/Departman/DepartmanEkle" class="btn btn-info">Departman Ekle</a>
{
    public class DepartmanController : Controller
    {
        // GET: Departman
        Context c = new Context();
        public ActionResult Index()
        {
            var degerler = c.Departmans.ToList();
            return View(degerler);
```

```
        }
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Departman
@{
    ViewBag.Title = "DepartmanEkle";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
<br />
<h2>Yeni Departman Sayfası</h2>
<br />
<form class="form-group" method="post">
    @Html.LabelFor(x => x.DepartmanAd)
    @Html.TextBoxFor(x => x.DepartmanAd, new { @class = "form-control" })
    <br />
    <button class="btn btn-info"> Kaydet</button>
</form>
        [HttpGet]
        public ActionResult DepartmanEkle()
        {
            return View();
        }
        [HttpPost]
        public ActionResult DepartmanEkle(Departman d)
        {
            c.Departmans.Add(d);
            c.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Departman
```

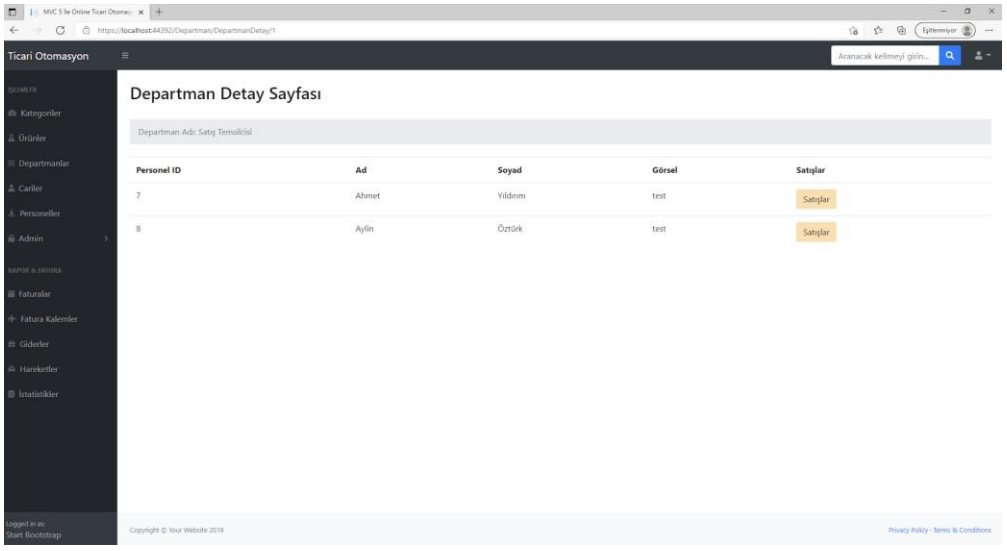
```
@{
    ViewBag.Title = "DepartmanEkle";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
<br />
<h2>Yeni Departman Sayfası</h2>
<br />

<form class="form-group" method="post">
    @Html.LabelFor(x => x.DepartmanAd)
    @Html.TextBoxFor(x => x.DepartmanAd, new { @class = "form-control" })
    <br />
    <button class="btn btn-info"> Kaydet</button>
</form>
@model MVCOnlineTicariOtomasyon.Models.Siniflar.Departman
@{
    ViewBag.Title = "DepartmanGetir";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}

<br />
<h2>Departman Güncelleme</h2>
<br />
@using (Html.BeginForm("DepartmanGuncelle", "Departman", FormMethod.Post))
{
    <div class="form-group">
        @Html.LabelFor(x => x.DepartmanAd)
        @Html.TextBoxFor(x => x.DepartmanAd, new { @class = "form-control" })
        @Html.HiddenFor(x => x.Departmanid)
```

```
    </div>
    <button class="btn btn-warning">Güncelle</button>
}
@{
    ViewBag.Title = "DepartmanDetay";
    Layout = "~/Views/Shared/AdminLayout.cshtml";
}
<br/>
<h2>Departman Detay Sayfası</h2>
<br/>
<div style="background-color:darkgray">
    deneme
</div>
<br/>
<table class="table table-hover">
    <tr>
        <th>Personel ID</th>
        <th>Ad</th>
        <th>Soyad</th>
        <th>Görsel</th>
        <th>Satışlar</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Ali</td>
        <td>Bulut</td>
        <td>Test</td>
        <td><a    href="#"    class="btn    btn-default"    style="background-
color:orangered">Satışlar</a></td>
    </tr>
    <tr>
```

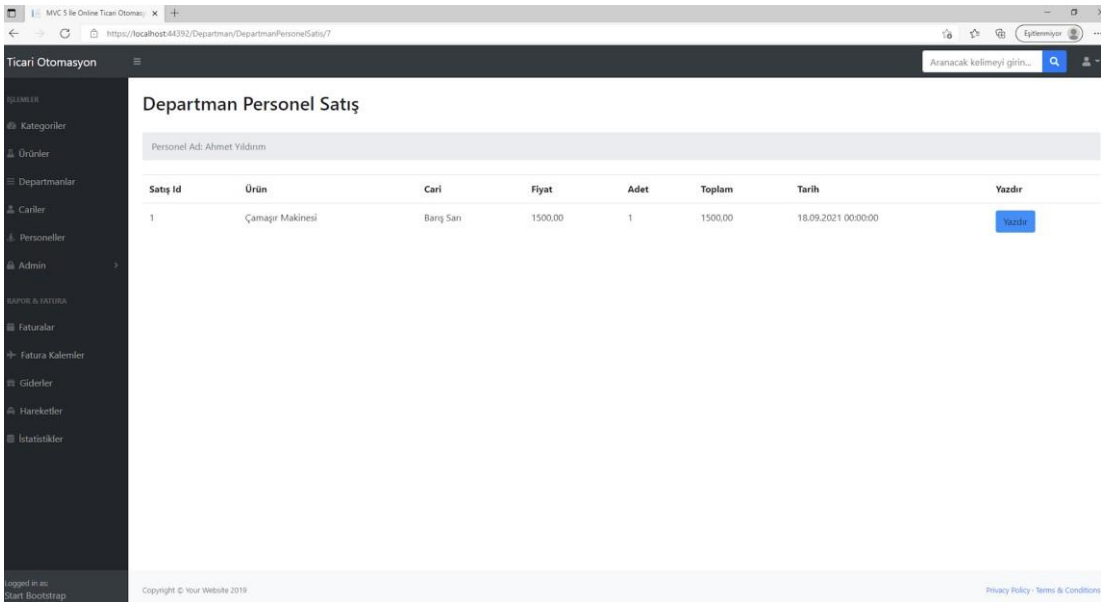
```
        <td>2</td>
       <td>Hasan</td>
        <td>Kaya</td>
        <td>Test</td>
        <td><a href="#" class="btn btn-default" style="background-color:orangered">Satışlar</a></td>
    </tr>
</table>
```



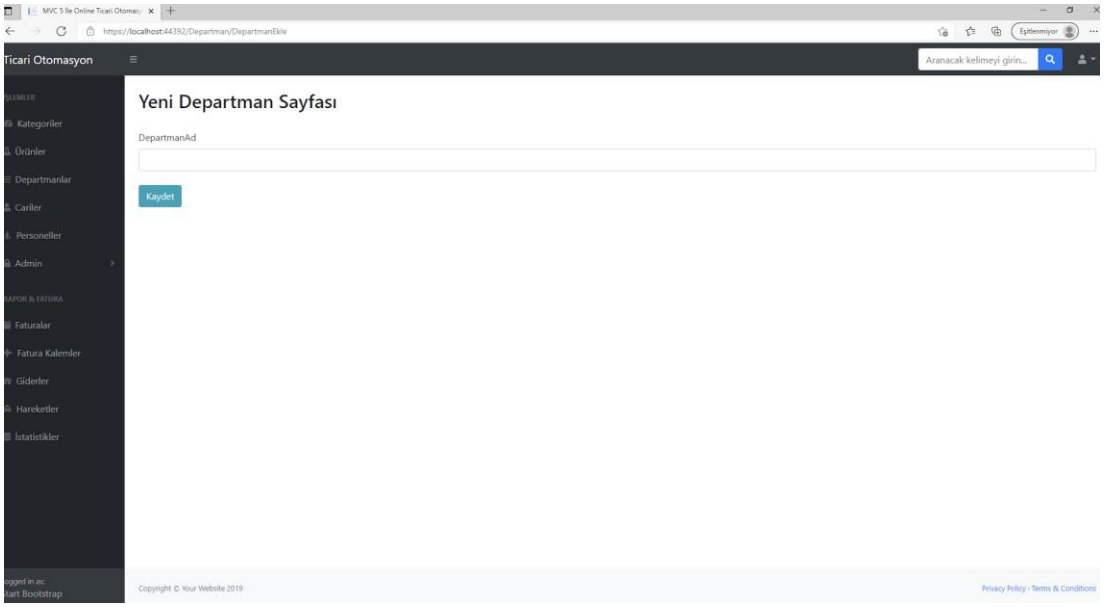
Şekil 3: Departman Güncelleme



Şekil 3.1: Departman Detay Sayfası

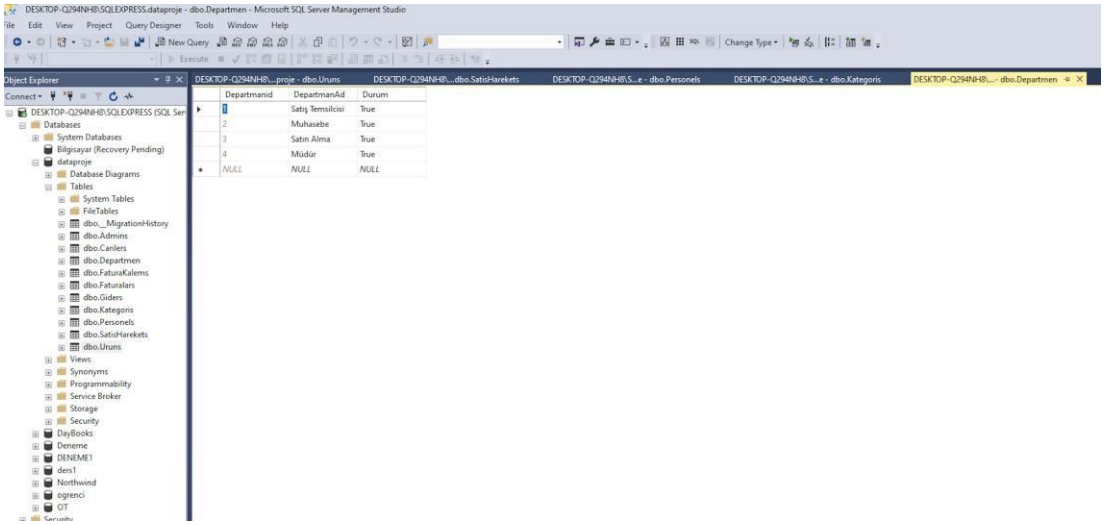


Şekil 3.2: Departman Personel Satış Sayfası

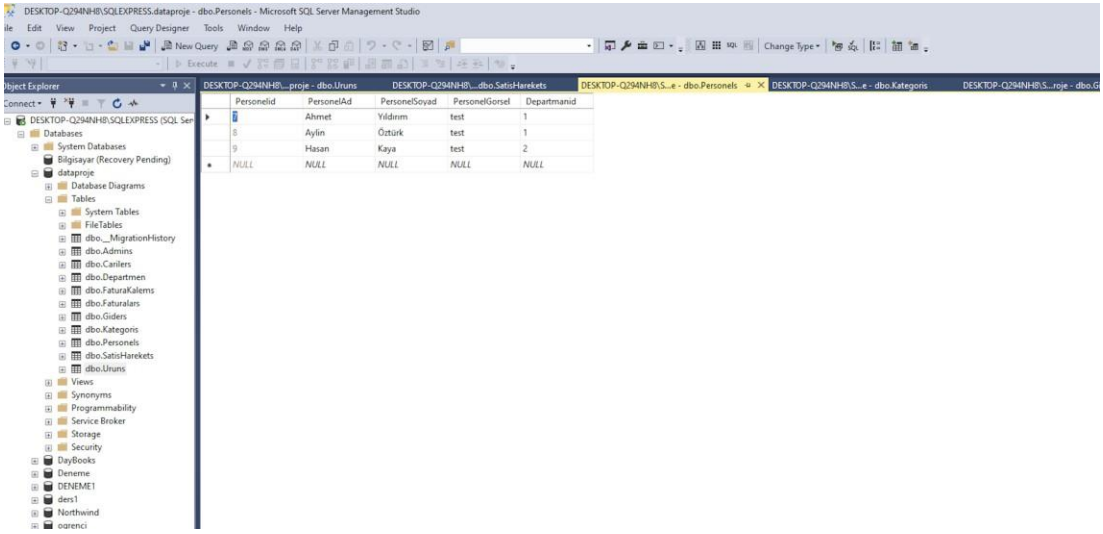


Şekil 3.3: Yeni Departman Ekleme Sayfası

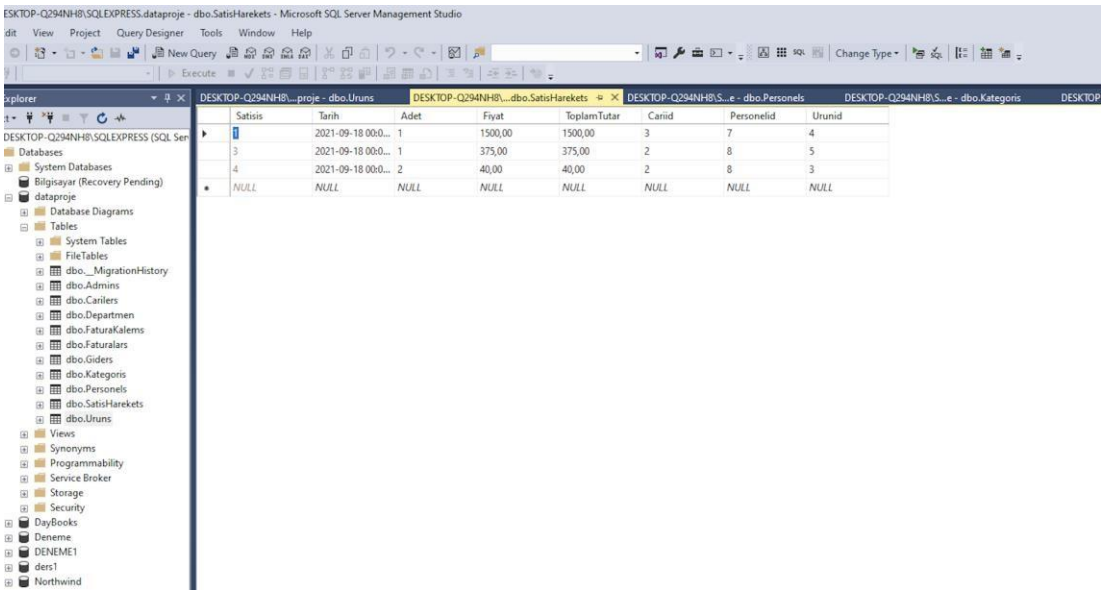
Proje veritabanında bulunan departman tablosuna otomasyon üzerinden yapılan ürün ekleme,güncelleme,silme ve listeleme işlemleri aktif olarak yansımaktadır.



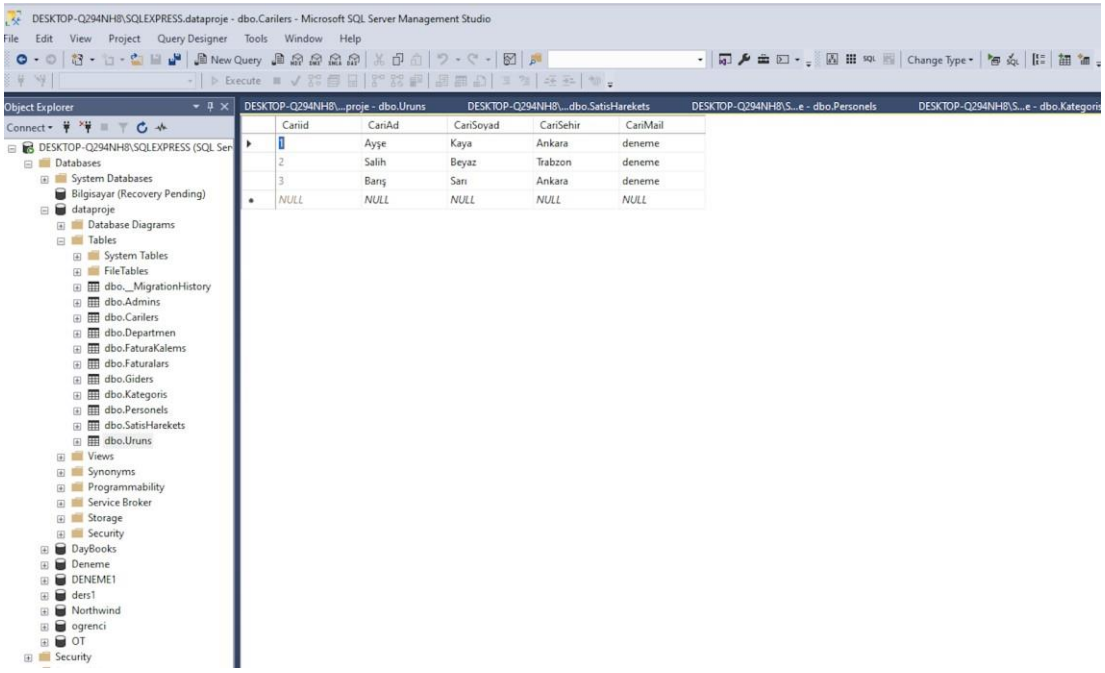
Şekil 3.4: Departman Veri Tabanı tablosu



Şekil 3.5: Personel Veri Tabanı tablosu



Şekil 3.6: Satış Hareket Veri Tabanı tablosu



Şekil 3.7: Müşteri Veri Tabanı Tablosu

5.SONUÇ

Ticari Otomasyon Sistemi piyasa üzerindeki tüm elektronik ürünlerin satımı için kullanılabilecek bir otomasyon sistemidir. Bu otomasyon sistemi ile satım, personel, müşteri, kategori, fatura veya gelir/gider takipleri hızlı ve güvenli bir şekilde yapılabilmektedir. Bu otomasyon ile zamandan ve iş gücünden ciddi miktarda tasarruf edilebilmektedir. Bu otomasyon sistemi kullanılarak insan hatası oranı minimum düzeye indirgenebilmektedir. Ticari Otomasyon Sistemi, diğer otomasyonlar gibi 7/24 çalışabilen bir otomasyon sistemidir. Ancak dezavantajlarına değinecek olursak; ilk kurulum aşamasında yüksek bir maliyet çıkabilir ve bazı donanım eksiklikleri ile karşılaşılabilir. Her ne kadar Ticari Otomasyon Sistemi'nin kullanımı kolay olsa da ileride kullanacak firmalar ilk aşamada adaptasyon sorunları yaşayabilir.

Ticari Otomasyon Sistemi'nin gelecek planlarından bahsedecek olursak; şu an sadece elektronik ürünlerin satan firmalara hizmet eden bir otomasyon sistemidir. Ancak kullanım miktarına ve gelen dönüşlere göre bu sınır ortadan kaldırılabilir ve e-ticaret dünyasındaki diğer ürünlerin satımında da aktif rol alabilir. Şimdilik Ticari Otomasyon Sistemi'nin ara yüzü kullanım kolaylığı, karışıklık olmaması için normalden biraz daha sade tutuldu. Ancak ilerleyen dönemlerde ara yüz konuşunda iyileştirmeler yapılabilir veya kullanan firmaların geri dönüşlerine göre sisteme yeni fonksiyonlar eklenebilir.

Kaynaklar

[1] ShiftDelete.Net, "Otomasyon Nedir, 2019". Erisim: 01.06.2023, https://shiftdelete.net/otomasyon-nedir

[2] Udemy, "Asp.Net+Docker". Erisim: 26.05.2023, https://www.udemy.com/course/aspnet-core-docker/

[3] Sami Acar (2006) Bilgi Teknolojisindeki Gelişmelerin Ofis Sistemleri Üzerindeki Etkisi ve Ofislerde Görsel Otomasyon, Gazi Üniversitesi Sosyal Bilimler EnstitüsüYüksek Lisans Tezi, Ankara

[4] Vikipedi, " Bellek (Bilgisayar]" Erisim: 01.06.2023, https://www.turkcewiki.org/wiki/Bellek_%28bilgisayar%29

[5] Vikipedi, "Bilgisayar Donanımı Tarihi". Erisim: 24.05.2023, https://tr.wikipedia.org/wiki/Bilgisayar_donan%C4%B1m%C4%B1_tarihi#:~:text=1944%2C%20Harvard%20Mark%20I%20isimli,1947%2C%20Transist%C3%B6r%20icat%20edildi.

[6] Statista, "Size of the business automation (BPA) market worldwide from 2016 to 2021" Erisim: 27.05.2023, https://www.statista.com/statistics/740593/worldwide-business-process-automation-market-size/

[7] Tesla Akademi, "Otomasyon Sistemi Nedir?".Erisim: 01.06.2023,https://teslaakademi.com/otomasyon-sistemi-nedir

[8] Microsoft, "C# Guide". Erisim:27.05.2023, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[9] Visual Studio Magazine, "C# Historical" Erisim: 01.06.2023, https://visualstudiomagazine.com/articles/2020/06/02/~/media/ECG/visualstudiomagazine/Images/2020/06/csharp_historical.asxh

[10] Microsoft, "Visual Studio". Erisim: 27.05.2023, https://visualstudio.microsoft.com/

[11] Edureka, "Visual Studio Tutorial". Erisim: 27.05.2023, https://www.edureka.co/blog/visual-studio-tutorial/

[12] Microsoft, "Download SQL Server Management Studio (SSMS)". Erisim: 27.05.2023, https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Özgeçmiş

Adı Soyadı: Buse Ceyda EKREN

E-mail (1): ceydaekrenn@gmail.com

E-mail (2): buseekren@outlook.com

Eğitim:

2016–2021 Kırklareli Üniversitesi, Yazılım Mühendisliği 2022–2023 İzmir Kâtip Çelebi Üniversitesi, Yazılım Mühendisliği

İş Deneyimi:

2021 – 2022 Ericsson Araştırma ve Geliştirme Bilişim Hizmetleri A.Ş

2022 – 2023 HUAWEI

2023- Devam Ediyor KAIZEN TASARIM VE TEKNOLOJI