**T.C. CELAL BAYAR ÜNİVERSİTESİ**

**KIRKAĞAÇ MESLEK YÜKSEKOKULU**

**BİLGİSAYAR PROGRAMCILIĞI BÖLÜMÜ**

metin, grafik, yazı tipi, grafik tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Personel Vardiya Otomasyonu**

**(SİSTEM ANALİZİ VE TASARIM DERSİ)**

**Hazırlayanlar**

**221809032 Muzaffer KABADAYI**

**221809010 İlkercan CİHAN**

**221809084 Efe DEMİR**

**211809024 Cansu HAN**

**Danışman**

**Öğr.Gör. Murat ALBAYRAK**

**Manisa**

**2024**

İçindekiler

[Kısa Özet 3](#_Toc156943771)

[**1.** **LOGİN EKRANI** 3](#_Toc156943772)

[1.1. Giriş Butonu 4](#_Toc156943773)

[2. YÖNETİCİ ANA MENÜSÜ 5](#_Toc156943774)

[3. KULLANICILAR BUTONU 6](#_Toc156943775)

[3.1. Kullanıcılar Tablosu 6](#_Toc156943776)

[3.2. Ekle Butonu 6](#_Toc156943777)

[3.3. Güncelle Butonu 6](#_Toc156943778)

[3.4. Sil Butonu 6](#_Toc156943779)

[4. MANUEL VARDİYA BUTONU 7](#_Toc156943780)

[5. OTOMATİK VARDİYA BUTONU 8](#_Toc156943781)

[6. GÜNCELLE BUTONU 8](#_Toc156943782)

[7. SİL BUTONU 8](#_Toc156943783)

[8. TÜMÜNÜ SİL BUTONU 8](#_Toc156943784)

[9. YAZDIR BUTONU 9](#_Toc156943785)

[10. ROLLER BUTONU 9](#_Toc156943786)

[10.1. Ekle Butonu 9](#_Toc156943787)

[10.2. Güncelle Butonu 9](#_Toc156943788)

[10.3. Sil Butonu 9](#_Toc156943789)

[11. RESMİ TATİLLER BUTONU 10](#_Toc156943790)

[11.1. Ekle Butonu 10](#_Toc156943791)

[11.2. Güncelle Butonu 10](#_Toc156943792)

[11.3. Sil Butonu 10](#_Toc156943793)

[12. İZİN GÜNLERİ BUTONU 11](#_Toc156943794)

[12.1. Ekle Butonu 11](#_Toc156943795)

[12.2. Sil Butonu 11](#_Toc156943796)

[13. VARDİYA TABLOSU 12](#_Toc156943797)

[14. PROGRAM HAZIRLANMAYA BAŞLANDIĞINDA YAZILAN KODLAR 13](#_Toc156943798)

[14.1. Veri Tabanı Bağlantı Ekranı 13](#_Toc156943799)

[14.2. Giriş Ekranı 13](#_Toc156943800)

[14.3. Menü Ekranı 15](#_Toc156943801)

[14.4. Kullanıcı Ekleme Ekranı 25](#_Toc156943802)

[14.5. Manuel Vardiya Ekleme Ekranı 28](#_Toc156943803)

[14.6. Otomatik Vardiya Ekleme Ekranı 37](#_Toc156943804)

[14.7. Roller Ekranı 43](#_Toc156943805)

[14.8. Resmi Tatiller Ekranı 48](#_Toc156943806)

[14.9. İzin Ekranı 52](#_Toc156943807)

# Kısa Özet

Personel vardiya otomasyonu, bir işletmenin personel yönetimi süreçlerini daha etkili ve verimli bir şekilde yönetmek amacıyla kullanılan bir sistemdir. Bu otomasyon sistemleri, personel vardiyalarını planlama, takip etme, değişiklikleri bildirme, raporlama ve diğer ilgili görevleri kolaylaştırmak için tasarlanmıştır. Temel amaç, işletmelerin personel kaynaklarını daha iyi yönetmelerine, vardiyaları düzenli bir şekilde planlamalarına ve iş gücü verimliliğini artırmalarına yardımcı olmaktır. Bu otomasyon sistemleri genellikle çeşitli sektörlerde, özellikle perakende, sağlık, üretim ve hizmet sektörlerinde yaygın olarak kullanılmaktadır.

# **LOGİN EKRANI**

Login ekranı yöneticilerin ve personellerin programa girmesi için sisteme giriş yaptığı ekrandır.

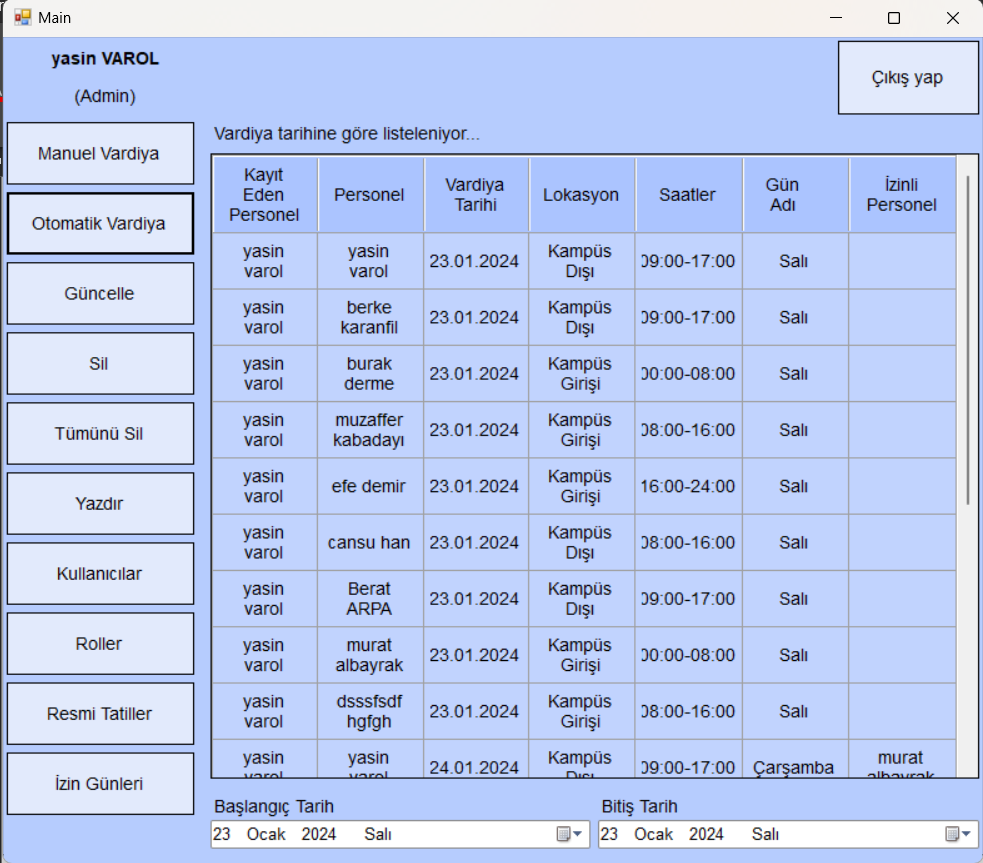
**metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu**

# Giriş Butonu

Girişe tıkladığında sisteme giriş yapmaktadır.

# YÖNETİCİ ANA MENÜSÜ

****

Ana menüde Manuel Vardiya, Otomatik Vardiya, Güncelle, Sil, Tümünü Sil, Yazdır, Kullanıcılar, Roller, Resmi Tatiller ve İzin Günlerine erişilmektedir.

# KULLANICILAR BUTONU

Kullanıcılar butonundan personel bilgileri ve düzenleme bölümüne ulaşılır.

metin, ekran görüntüsü, ekran, görüntüleme, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kullanıcılar ekranında personellerin bilgilerini ekleme, ekledikten sonra tekrar güncelleme, silme işlemlerini, haftalık tatil günleri ekleme işlemini yapar.

# Kullanıcılar Tablosu

Personeller tablosunda personellere ait bilgilere ulaşılır.

# Ekle Butonu

Yeni bir personel bilgilerini girdikten sonra personeli sisteme ekleyen butondur.

# Güncelle Butonu

Eklenen personelde herhangi bir düzenleme yapıldığı zaman sistemden tekrar değiştirmek ve güncellemek için kullanılan butondur.

# Sil Butonu

Eğer bir personeli programdan çıkarmak için personelin üstüne tıklayıp sil butonuna tıklanması yeterlidir.

# MANUEL VARDİYA BUTONU

Manuel bir şekilde vardiya ekleme bölümüne ulaşılır.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Vardiya ekleme arayüzünde eklenecek vardiya tarihi seçilir. Vardiya konumu ve saati de seçilir. Son olarak vardiya eklenecek kişi seçilir ve devam et butonuna basılır. Sonuç olarak manuel vardiya oluşturulmuş olur.

# OTOMATİK VARDİYA BUTONU

Otomatik bir şekilde vardiya ekleme bölümüne ulaşılır.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Otomatik eklenecek vardiyanın başlangıç ve bitiş tarihleri seçilir. Daha sonra ekle butonuna basılır ve vardiya oluşturulur.

Çıkış butonuna basılır ve arayüz kapanır.

# GÜNCELLE BUTONU

Personel vardiyaları üzerinde değişiklik yapabildiğimiz bölüme ulaşılır (Vardiya Ekleme).

# SİL BUTONU

Tek bir vardiyayı siler.

# TÜMÜNÜ SİL BUTONU

Tüm vardiyaları siler.

# YAZDIR BUTONU

Mevcut vardiyaların çıktısını almaya yarar.

# ROLLER BUTONU

Kullanıcı yetkilerini düzenlememizi sağlar.

metin, ekran görüntüsü, ekran, görüntüleme, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Ekle Butonu

Ekle butonuna basarak yeni rol eklenir.

# Güncelle Butonu

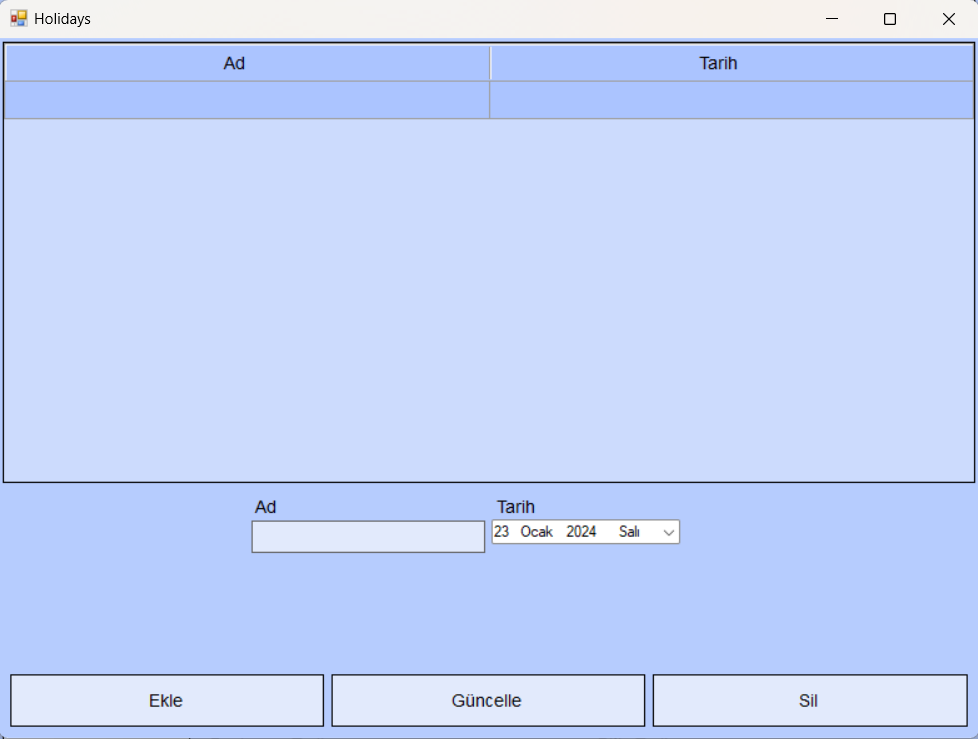
Güncelle butonuna basarak mevcut rol üzerinde yetkileri değiştirebiliriz.

# Sil Butonu

Sil butonuna basarak mevcut rolleri silebiliriz.

# RESMİ TATİLLER BUTONU

Yıl içerisindeki resmi tatilleri eklemeye veya düzenlemeye yarar.



# Ekle Butonu

Tatil adı, tarihi girilerek ekle butonuna basılır ve tatil günü eklenmiş olur.

# Güncelle Butonu

Güncelle butonuna basılarak mevcut tatil gününü düzenleyebiliriz.

# Sil Butonu

Sil butonuna basarak mevcut tatil silinebilir.

# İZİN GÜNLERİ BUTONU

İşçilerin özel izin almasını sağlar.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

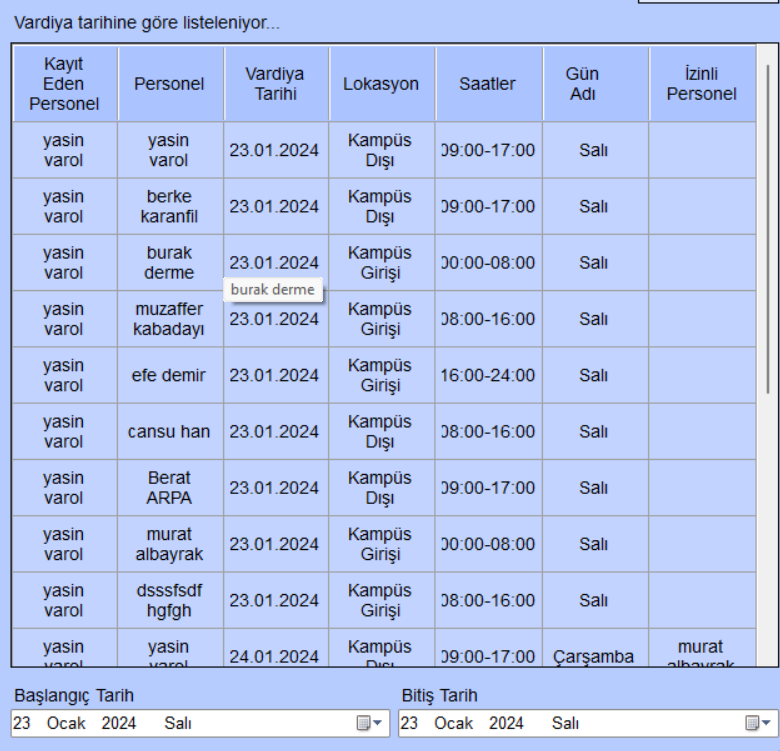
# Ekle Butonu

Personellere izin tarihlerini atadıktan sonra ekle butonundan izin oluşturur.

# Sil Butonu

Yönetici izin silmek isterse personeli seçip sil butonuna basması yeterlidir.

# VARDİYA TABLOSU



Mevcut vardiyaları, tarihlerini, saatlerini, günü ve o gün izinli olan kişiyi listeleyen tablodur.

# PROGRAM HAZIRLANMAYA BAŞLANDIĞINDA YAZILAN KODLAR

# Veri Tabanı Bağlantı Ekranı

using System.Data;

using System.Data.SqlClient;

namespace PersonelVardiyaOtomasyonu.Class.Helper

{

public static class ConnectionStrings

{

public static SqlConnection \_sqlConnection = new SqlConnection(@"Server=.; Initial Catalog=PersonelVardiyaOtomasyonu; Integrated Security=true; MultipleActiveResultSets=true");

public static void ConnectionControl()

{

if (\_sqlConnection.State != ConnectionState.Open)

{

\_sqlConnection.Open();

}

}

}

}

# Giriş Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Class.Helper;

using System;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class Login : Form

{

public Login()

{

InitializeComponent();

}

UserDal \_userDal = new UserDal();

RoleDal \_roleDal = new RoleDal();

private void txt\_IdentificationNumber\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private void btn\_login\_Click(object sender, EventArgs e)

{

if (txt\_IdentificationNumber.Text != "" && txt\_password.Text != "")

{

if (\_userDal.CheckByIdentificationNumberAndPassword(txt\_IdentificationNumber.Text, txt\_password.Text))

{

UserInformation.User = \_userDal.GetByIdentificationNumber(txt\_IdentificationNumber.Text);

UserInformation.Role = \_roleDal.GetById(UserInformation.User.RoleId);

Main main = new Main();

main.Show();

this.Hide();

txt\_IdentificationNumber.Clear();

txt\_password.Clear();

}

else

{

MessageBox.Show("Kimlik Numarasını veya Parolayı doğru girdiğinizden emin olunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void Login\_Load(object sender, EventArgs e)

{

}

}

}

# Menü Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Class.Helper;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Globalization;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

private bool isFormLoaded = false;

ShiftDal \_shiftDal = new ShiftDal();

UserDal \_userDal = new UserDal();

private void Main\_Load(object sender, EventArgs e)

{

isFormLoaded = true;

CheckUserRole();

GetList();

}

public void GetList()

{

dgv\_shifts.Rows.Clear();

List<Shift> shifts;

if (UserInformation.Role.SeeAllShifts)

{

shifts = \_shiftDal.GetAllWithDate(dtp\_startDate.Value, dtp\_endDate.Value);

}

else

{

shifts = \_shiftDal.GetAllWithDateAndEmployee(dtp\_startDate.Value, dtp\_endDate.Value, UserInformation.User.Id);

}

foreach (Shift shift in shifts)

{

var registrantUser = \_userDal.GetById(shift.RegistrantId);

var user = \_userDal.GetById(shift.EmployeeId);

string durum = shift.IsNew ? "Aktif Vardiya" : "Pasif Vardiya";

dgv\_shifts.Rows.Add(shift.Id, shift.RegistrantId, registrantUser.NameSurname, shift.EmployeeId, user.NameSurname,

shift.DateOfRegistration.ToShortDateString(), shift.Date.ToShortDateString(), shift.Location, shift.Hours, shift.IsNew,

durum, lessonName(shift.Date),shiftUsers(shift.Date));

}

dgv\_shifts.ClearSelection();

}

public static string shiftUsers(DateTime Days)

{

string bugununGunu = Days.ToString("dddd", new CultureInfo("tr-TR"));

string name = "";

string surname = "";

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT Name, Surname FROM Users WHERE leave\_day\_1=@day or leave\_day\_2=@day",

ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@day", bugununGunu);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

name = sqlDataReader["Name"].ToString();

surname = sqlDataReader["Surname"].ToString();

}

}

}

ConnectionStrings.\_sqlConnection.Close();

// Veritabanındaki sütunlar ne tür ise ona göre işlem yapın

// Örneğin, eğer veritabanında nvarchar sütunlarsa, tip dönüşümüne gerek yoktur.

// Eğer int sütunlarsa, int.TryParse() gibi bir yöntem kullanmalısınız.

// Örneğin, int sütunlar olarak düşünüyorsanız:

// int leaveDay1, leaveDay2;

// int.TryParse(leave\_day\_1, out leaveDay1);

// int.TryParse(leave\_day\_2, out leaveDay2);

return name + " " + surname;

}

public static string lessonName(DateTime tarih) {

string bugununGunu = tarih.ToString("dddd", new CultureInfo("tr-TR"));

return bugununGunu;

}

private void dtp\_endDate\_ValueChanged(object sender, EventArgs e)

{

GetList();

}

public void CheckUserRole()

{

int collapse = flp\_buttons.Controls.Count;

lbl\_nameSurname.Text = UserInformation.User.Name + " " + UserInformation.User.Surname.ToUpper();

lbl\_roleName.Text = "(" + UserInformation.Role.Name + ")";

if (!UserInformation.Role.Add)

{

btn\_add.Visible = false;

collapse--;

}

if (!UserInformation.Role.Update)

{

btn\_update.Visible = false;

collapse--;

}

if (!UserInformation.Role.Delete)

{

btn\_delete.Visible = false;

collapse--;

}

if (!UserInformation.Role.Print)

{

btn\_print.Visible = false;

collapse--;

}

if (!UserInformation.Role.UsersManagement)

{

btn\_users.Visible = false;

collapse--;

}

if (!UserInformation.Role.RolesManagement)

{

btn\_roles.Visible = false;

collapse--;

}

if (collapse == 0)

{

splitContainer1.Panel1Collapsed = true;

}

}

private void btn\_logout\_Click(object sender, EventArgs e)

{

Login loginCheck = (Login)Application.OpenForms["Login"];

if (loginCheck != null)

{

loginCheck.Show();

}

else

{

Login login = new Login();

login.Show();

}

this.Close();

}

private void Main\_SizeChanged(object sender, EventArgs e)

{

// Sadece form ilk kez yüklendiğinde SplitterDistance'ı sabitle

if (isFormLoaded)

{

// SplitterDistance sınırları kontrol et

int minSize = splitContainer1.Panel1MinSize;

int maxSize = this.Width - splitContainer1.Panel2MinSize;

// Geçerli SplitterDistance değerini al

int currentDistance = splitContainer1.SplitterDistance;

// Değer sınırlar içinde değilse, düzelt

if (currentDistance < minSize)

{

splitContainer1.SplitterDistance = minSize;

}

else if (currentDistance > maxSize)

{

splitContainer1.SplitterDistance = maxSize;

}

}

}

#region LeftMenu

private void btn\_roles\_Click(object sender, EventArgs e)

{

Roles roles = new Roles();

roles.ShowDialog();

}

private void btn\_add\_Click(object sender, EventArgs e)

{

ShiftAdd shiftAdd = new ShiftAdd();

shiftAdd.ShowDialog();

}

private void btn\_update\_Click(object sender, EventArgs e)

{

if (dgv\_shifts.SelectedRows.Count != 0)

{

ShiftUpdate shiftUpdate = new ShiftUpdate(Convert.ToInt32(dgv\_shifts.CurrentRow.Cells[0].Value));

shiftUpdate.ShowDialog();

}

else

{

MessageBox.Show("Güncellenecek vardiyayı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_users\_Click(object sender, EventArgs e)

{

Users users = new Users();

users.ShowDialog();

}

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (dgv\_shifts.SelectedRows.Count != 0)

{

if (MessageBox.Show("Vardiyayı silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

\_shiftDal.Delete(Convert.ToInt32(dgv\_shifts.CurrentRow.Cells[0].Value));

GetList();

dgv\_shifts.ClearSelection();

}

}

else

{

MessageBox.Show("Silinecek vardiyayı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

#endregion

private void btn\_print\_Click(object sender, EventArgs e)

{

DataTable dataTable = new DataTable();

// Add necessary columns

dataTable.Columns.Add("Personel");

dataTable.Columns.Add("Vardiya Tarihi");

dataTable.Columns.Add("Lokasyon");

dataTable.Columns.Add("Saatler");

dataTable.Columns.Add("Gün Adı");

dataTable.Columns.Add("Izinli Personel");

for (int i = 0; i < dgv\_shifts.Rows.Count; i++)

{

DataRow dataRow = dataTable.NewRow();

// Populate necessary columns

dataRow["Personel"] = dgv\_shifts.Rows[i].Cells[4].Value;

dataRow["Vardiya Tarihi"] = dgv\_shifts.Rows[i].Cells[6].Value;

dataRow["Lokasyon"] = dgv\_shifts.Rows[i].Cells[7].Value;

dataRow["Saatler"] = dgv\_shifts.Rows[i].Cells[8].Value;

dataRow["Gün Adı"] = dgv\_shifts.Rows[i].Cells[11].Value;

dataRow["Izinli Personel"] = dgv\_shifts.Rows[i].Cells[12].Value;

dataTable.Rows.Add(dataRow);

}

PrintPDFBuilder.Print(dataTable, dtp\_startDate.Value, dtp\_endDate.Value);

}

private void button1\_Click(object sender, EventArgs e)

{

otomasyon oto = new otomasyon();

oto.ShowDialog();

}

private void button2\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Vardiyaları silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

ConnectionStrings.ConnectionControl();

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("DELETE FROM Shifts", ConnectionStrings.\_sqlConnection))

{

sqlCommand.ExecuteNonQuery();

}

ConnectionStrings.\_sqlConnection.Close();

Main main = (Main)Application.OpenForms["Main"];

main.GetList();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Holidays holidays = new Holidays();

holidays.ShowDialog();

}

private void button4\_Click(object sender, EventArgs e)

{

Leaves leaves = new Leaves();

leaves.ShowDialog();

}

}

}

# Kullanıcı Ekleme Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class Users : Form

{

public Users()

{

InitializeComponent();

}

UserDal \_userDal = new UserDal();

RoleDal \_roleDal = new RoleDal();

private void Users\_Load(object sender, EventArgs e)

{

GetRoleList();

GetList();

}

public void GetRoleList()

{

cb\_roles.DisplayMember = "Name";

cb\_roles.ValueMember = "Id";

cb\_roles.DataSource = \_roleDal.GetAll();

}

public void GetList()

{

dgv\_users.Rows.Clear();

var users = \_userDal.GetAll();

foreach (var user in users)

{

var role = \_roleDal.GetById(user.RoleId);

dgv\_users.Rows.Add(

user.Id,

user.RoleId,

role.Name,

user.IdentificationNumber,

user.Name,

user.Surname,

user.Password,

user.Address,

user.EMail,

user.PhoneNumber,

user.Status,

user.leave\_day\_1,

user.leave\_day\_2

);

}

dgv\_users.ClearSelection();

}

private void btn\_add\_Click(object sender, EventArgs e)

{

if (cb\_roles.Text != "" && txt\_identificationNumber.Text != "" && txt\_name.Text != "" && txt\_surname.Text != "" && txt\_password.Text != "" && txt\_address.Text != "" && txt\_email.Text != "" && txt\_phoneNumber.Text != "" && cb\_leave\_day\_1.Text != "" && cb\_leave\_day\_2.Text != "")

{

if (!\_userDal.CheckByIdentificationNumber(txt\_identificationNumber.Text))

{

User user = new User

{

RoleId = Convert.ToInt32(cb\_roles.SelectedValue),

IdentificationNumber = txt\_identificationNumber.Text,

Name = txt\_name.Text,

Surname = txt\_surname.Text,

Password = txt\_password.Text,

Address = txt\_address.Text,

EMail = txt\_email.Text,

PhoneNumber = txt\_phoneNumber.Text,

leave\_day\_1 = cb\_leave\_day\_1.Text,

leave\_day\_2 = cb\_leave\_day\_2.Text,

Status = Convert.ToBoolean(dgv\_users.CurrentRow.Cells[10].Value),

};

\_userDal.Add(user);

GetList();

Clear();

}

else

{

MessageBox.Show("Aynı T.C. Kimlik Numaralı personel bulunmaktadır!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_update\_Click(object sender, EventArgs e)

{

if (dgv\_users.SelectedRows.Count != 0)

{

if (cb\_roles.Text != "" && txt\_identificationNumber.Text != "" && txt\_name.Text != "" && txt\_surname.Text != "" && txt\_password.Text != "" && txt\_address.Text != "" && txt\_email.Text != "" && txt\_phoneNumber.Text != "" && cb\_leave\_day\_1.Text != "" && cb\_leave\_day\_2.Text != "")

{

User user = new User

{

Id = Convert.ToInt32(dgv\_users.CurrentRow.Cells[0].Value),

RoleId = Convert.ToInt32(cb\_roles.SelectedValue),

IdentificationNumber = txt\_identificationNumber.Text,

Name = txt\_name.Text,

Surname = txt\_surname.Text,

Password = txt\_password.Text,

Address = txt\_address.Text,

EMail = txt\_email.Text,

PhoneNumber = txt\_phoneNumber.Text,

leave\_day\_1 = cb\_leave\_day\_1.Text,

leave\_day\_2 = cb\_leave\_day\_2.Text,

Status = Convert.ToBoolean(dgv\_users.CurrentRow.Cells[10].Value),

};

\_userDal.Update(user);

GetList();

Clear();

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Güncellenecek kullanıcıyı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (dgv\_users.SelectedRows.Count != 0)

{

if (MessageBox.Show("Kullanıcıyı silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

\_userDal.Delete(Convert.ToInt32(dgv\_users.CurrentRow.Cells[0].Value));

GetList();

Clear();

}

}

else

{

MessageBox.Show("Silinecek kullanıcıyı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void dgv\_users\_CellClick(object sender, DataGridViewCellEventArgs e)

{

cb\_roles.Text = dgv\_users.CurrentRow.Cells[2].Value.ToString();

txt\_identificationNumber.Text = dgv\_users.CurrentRow.Cells[3].Value.ToString();

txt\_name.Text = dgv\_users.CurrentRow.Cells[4].Value.ToString();

txt\_surname.Text = dgv\_users.CurrentRow.Cells[5].Value.ToString();

txt\_password.Text = dgv\_users.CurrentRow.Cells[6].Value.ToString();

txt\_address.Text = dgv\_users.CurrentRow.Cells[7].Value.ToString();

txt\_email.Text = dgv\_users.CurrentRow.Cells[8].Value.ToString();

txt\_phoneNumber.Text = dgv\_users.CurrentRow.Cells[9].Value.ToString();

cb\_leave\_day\_1.Text = dgv\_users.CurrentRow.Cells[11].Value.ToString();

cb\_leave\_day\_2.Text = dgv\_users.CurrentRow.Cells[12].Value.ToString();

}

public void Clear()

{

cb\_roles.SelectedIndex = -1;

txt\_identificationNumber.Clear();

txt\_name.Clear();

txt\_surname.Clear();

txt\_password.Clear();

txt\_address.Clear();

txt\_email.Clear();

txt\_phoneNumber.Clear();

cb\_leave\_day\_1.SelectedIndex = -1;

cb\_leave\_day\_2.SelectedIndex = -1;

}

private void dgv\_users\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

# Manuel Vardiya Ekleme Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Class.Helper;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Data.SqlClient;

using System.Globalization;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class ShiftAdd : Form

{

public ShiftAdd()

{

InitializeComponent();

}

ShiftDal \_shiftDal = new ShiftDal();

UserDal \_userDal = new UserDal();

private void ShiftAdd\_Load(object sender, EventArgs e)

{

GetUserList();

}

public void GetUserList()

{

cb\_users.DisplayMember = "NameSurname";

cb\_users.ValueMember = "Id";

cb\_users.DataSource = \_userDal.GetAll();

}

private void cb\_locations\_TextChanged(object sender, EventArgs e)

{

cb\_hours.Items.Clear();

switch (cb\_locations.Text)

{

case "Kampüs Girisi":

cb\_hours.Items.Add("00:00 - 08:00");

cb\_hours.Items.Add("08:00 - 16:00");

cb\_hours.Items.Add("16:00 - 24:00");

break;

case "Kampüs Içi":

cb\_hours.Items.Add("08:00 - 16:00");

cb\_hours.Items.Add("09:00 - 17:00");

break;

default:

cb\_hours.Items.Add("Saat bulunmamaktadır!");

break;

}

}

private void btn\_continue\_Click(object sender, EventArgs e)

{

if (cb\_locations.Text != "" && cb\_hours.Text != "")

{

Shift shift = new Shift

{

RegistrantId = UserInformation.User.Id,

EmployeeId = Convert.ToInt32(cb\_users.SelectedValue),

DateOfRegistration = Convert.ToDateTime(DateTime.Now.ToShortDateString()),

Date = Convert.ToDateTime(dtp\_date.Value.ToShortDateString()),

Location = cb\_locations.Text,

Hours = cb\_hours.Text,

IsNew = true,

};

if (vardiyaKontrol(shift.EmployeeId, shift.Date)) {

MessageBox.Show("İşçi o gün zaten çalışıyor!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (ÖzelİzinGünü(shift.Date, shift.EmployeeId))

{

DialogResult result = MessageBox.Show("Resmi Tatil Günü. Yine de yeni vardiya eklemek istiyor musunuz?", "Uyarı",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.No)

{

return;

}

}

if (!HaftalıkİzinSorgulama(shift.EmployeeId, shift.Date))

{

DialogResult result2 = MessageBox.Show("Haftalık izin günü. Yine de yeni vardiya eklemek istiyor musunuz?", "Uyarı",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result2 == DialogResult.No)

{

return;

}

}

if (!IzinSorgulama(shift.EmployeeId, shift.Date))

{

DialogResult result3 = MessageBox.Show("Kullanıcı o gün izinli. Yine de yeni vardiya eklemek istiyor musunuz?", "Uyarı",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result3 == DialogResult.No)

{

return;

}

}

\_shiftDal.DeactivateAllNewShiftsTheEmployee(shift.EmployeeId);

\_shiftDal.Add(shift);

Main main = (Main)Application.OpenForms["Main"];

main?.GetList();

this.Close();

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private static bool vardiyaKontrol(double Sicil, DateTime Days)

{

int Count = 0;

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT COUNT(\*) FROM Shifts WHERE Date = @Tarih And EmployeeId = @Sicil", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Tarih", Days); // Corrected parameter name

sqlCommand.Parameters.AddWithValue("@Sicil", Sicil);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

// Increment the Count variable

Count = sqlDataReader.GetInt32(0);

}

}

}

ConnectionStrings.\_sqlConnection.Close();

// If Count is 0, return true; otherwise, return false

return Count != 0;

}

private static bool IzinSorgulama(double Sicil, DateTime Days)

{

DateTime? baslangic = null;

DateTime? bitis = null;

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT start\_date, end\_date FROM Leaves WHERE sicil\_no=@Id", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Id", Sicil);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

// Veritabanından alınan değerleri DateTime türüne dönüştür

if (DateTime.TryParse(sqlDataReader["start\_date"].ToString(), out DateTime startDate))

{

baslangic = startDate;

}

if (DateTime.TryParse(sqlDataReader["end\_date"].ToString(), out DateTime endDate))

{

bitis = endDate;

}

}

}

}

ConnectionStrings.\_sqlConnection.Close();

if (baslangic.HasValue && bitis.HasValue)

{

TimeSpan gunFarki = bitis.Value - baslangic.Value;

int gunSayisi = gunFarki.Days;

for (int i = 0; i <= gunSayisi; i++)

{

DateTime kontrolTarihi = baslangic.Value.AddDays(i);

if (kontrolTarihi.Date == Days.Date)

{

// Eğer Days, izin tarihleri arasında ise izin verme (false döndürme)

return false;

}

}

}

// İzin sorgulama işlemi başarılıysa true döndür

return true;

}

private static bool ÖzelİzinGünü(DateTime Days, int Sicil)

{

int holidayCount = 0;

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT COUNT(\*) FROM Holidays WHERE Date = @Tarih", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Tarih", Days); // @Tarih parametresini düzelt

holidayCount = (int)sqlCommand.ExecuteScalar();

}

ConnectionStrings.\_sqlConnection.Close();

// Eğer tatil günü varsa (holidayCount > 0), özel izin günüdür.

return holidayCount != 0;

}

private static bool HaftalıkİzinSorgulama(double Sicil, DateTime Days)

{

string bugununGunu = Days.ToString("dddd", new CultureInfo("tr-TR"));

string leave\_day\_1 = "";

string leave\_day\_2 = "";

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT Leave\_day\_1, Leave\_day\_2 FROM Users WHERE Id=@Id", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Id", Sicil);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

leave\_day\_1 = sqlDataReader["Leave\_day\_1"].ToString();

leave\_day\_2 = sqlDataReader["Leave\_day\_2"].ToString();

}

}

}

ConnectionStrings.\_sqlConnection.Close();

// Veritabanındaki sütunlar ne tür ise ona göre işlem yapın

// Örneğin, eğer veritabanında nvarchar sütunlarsa, tip dönüşümüne gerek yoktur.

// Eğer int sütunlarsa, int.TryParse() gibi bir yöntem kullanmalısınız.

// Örneğin, int sütunlar olarak düşünüyorsanız:

// int leaveDay1, leaveDay2;

// int.TryParse(leave\_day\_1, out leaveDay1);

// int.TryParse(leave\_day\_2, out leaveDay2);

return !(leave\_day\_1.Equals(bugununGunu) || leave\_day\_2.Equals(bugununGunu));

}

}

}

# Otomatik Vardiya Ekleme Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Class.Helper;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class otomasyon : Form

{

public otomasyon()

{

InitializeComponent();

}

private void otomasyon\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

DateTime baslangıc = DateTime.Parse(dateTimePicker2.Text);

DateTime bitir = DateTime.Parse(dateTimePicker1.Text);

int gunfarki = (int)(bitir - baslangıc).TotalDays;

List<string> kampüs1 = new List<string>();

kampüs1.Add("00:00-08:00");

kampüs1.Add("08:00-16:00");

kampüs1.Add("16:00-24:00");

List<string> kampüs2 = new List<string>();

kampüs2.Add("08:00-16:00");

kampüs2.Add("09:00-17:00");

List<string> kampüs = new List<string>();

kampüs.Add("Kampüs Girişi");

kampüs.Add("Kampüs Dışı");

for (int i = 0; i <= gunfarki + 1; i++)

{

DateTime Days = baslangıc.AddDays(i);

var k1 = 1;

var k2 = 0;

var vardiyaA = 0;

var vardiyaB = 0;

var vardiyaC = 0;

var vardiyaD = 0;

var vardiyaE = 0;

List<int> users = GetAllUserIds();

foreach (double Sicil in users)

{

int Verikampüs = 0;

String saat = "0";

if (IzinSorgulama(Sicil, Days))

{

if (HaftalıkİzinSorgulama(Sicil, Days)) {

if (ÖzelİzinGünü(Days))

{

if (k1 < k2)

{

int enKucukVardiye = Math.Min(vardiyaA, Math.Min(vardiyaB, vardiyaC));

if (enKucukVardiye == vardiyaA)

{

vardiyaA++;

saat = kampüs1[0];

}

else if (enKucukVardiye == vardiyaB)

{

vardiyaB++;

saat = kampüs1[1];

}

else if (enKucukVardiye == vardiyaC)

{

vardiyaC++;

saat = kampüs1[2];

k1++;

}

vardiyaEkleme(Sicil, Days, saat, "Kampüs Girişi");

continue;

}

else

{

Verikampüs = 1;

if (vardiyaD < vardiyaB)

{

vardiyaD++;

saat = kampüs2[0];

}

else

{

vardiyaE++;

saat = kampüs2[1];

k2++;

}

vardiyaEkleme(Sicil, Days, saat, "Kampüs Dışı");

continue;

}

}

if (Verikampüs == 0)

{

if (vardiyaD < vardiyaE)

{

vardiyaD++;

saat = kampüs2[0];

}

else

{

vardiyaE++;

saat = kampüs2[1];

k2++;

Verikampüs = 1;

}

vardiyaEkleme(Sicil, Days, saat, "Kampüs Dışı");

continue;

}

}

}

}

Main main = (Main)Application.OpenForms["Main"];

main.GetList();

}

}

public List<int> GetAllUserIds()

{

List<int> userIds = new List<int>();

try

{

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT Id FROM users", ConnectionStrings.\_sqlConnection))

{

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

int userId = sqlDataReader.GetInt32(0);

userIds.Add(userId);

}

}

}

}

catch (Exception ex)

{

// Hata yönetimi burada yapılabilir.

// Örneğin, hata durumunda bir log oluşturabilir veya uygun bir şekilde işleyebilirsiniz.

}

finally

{

// Bağlantıyı kapatma işlemini finally bloğunda yapmak önemlidir.

ConnectionStrings.\_sqlConnection.Close();

}

return userIds;

}

public static bool HaftalıkİzinSorgulama(double Sicil, DateTime Days)

{

string bugununGunu = Days.ToString("dddd", new CultureInfo("tr-TR"));

string leave\_day\_1 = "";

string leave\_day\_2 = "";

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT Leave\_day\_1, Leave\_day\_2 FROM Users WHERE Id=@Id", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Id", Sicil);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

leave\_day\_1 = sqlDataReader["Leave\_day\_1"].ToString();

leave\_day\_2 = sqlDataReader["Leave\_day\_2"].ToString();

}

}

}

ConnectionStrings.\_sqlConnection.Close();

// Veritabanındaki sütunlar ne tür ise ona göre işlem yapın

// Örneğin, eğer veritabanında nvarchar sütunlarsa, tip dönüşümüne gerek yoktur.

// Eğer int sütunlarsa, int.TryParse() gibi bir yöntem kullanmalısınız.

// Örneğin, int sütunlar olarak düşünüyorsanız:

// int leaveDay1, leaveDay2;

// int.TryParse(leave\_day\_1, out leaveDay1);

// int.TryParse(leave\_day\_2, out leaveDay2);

return !(leave\_day\_1.Equals(bugununGunu) || leave\_day\_2.Equals(bugununGunu));

}

private static bool ÖzelİzinGünü(DateTime Days)

{

int holidayCount = 0;

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT COUNT(\*) FROM Holidays WHERE Date = @Tarih", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Tarih", Days); // @Tarih parametresini düzelt

holidayCount = (int)sqlCommand.ExecuteScalar();

}

ConnectionStrings.\_sqlConnection.Close();

// Eğer tatil günü varsa (holidayCount > 0), özel izin günüdür.

return holidayCount == 0;

}

public static bool IzinSorgulama(double Sicil, DateTime Days)

{

DateTime? baslangic = null;

DateTime? bitis = null;

ConnectionStrings.ConnectionControl();

using (SqlCommand sqlCommand = new SqlCommand("SELECT start\_date, end\_date FROM Leaves WHERE sicil\_no=@Id", ConnectionStrings.\_sqlConnection))

{

sqlCommand.Parameters.AddWithValue("@Id", Sicil);

using (SqlDataReader sqlDataReader = sqlCommand.ExecuteReader())

{

while (sqlDataReader.Read())

{

// Veritabanından alınan değerleri DateTime türüne dönüştür

if (DateTime.TryParse(sqlDataReader["start\_date"].ToString(), out DateTime startDate))

{

baslangic = startDate;

}

if (DateTime.TryParse(sqlDataReader["end\_date"].ToString(), out DateTime endDate))

{

bitis = endDate;

}

}

}

}

ConnectionStrings.\_sqlConnection.Close();

if (baslangic.HasValue && bitis.HasValue)

{

TimeSpan gunFarki = bitis.Value - baslangic.Value;

int gunSayisi = gunFarki.Days;

for (int i = 0; i <= gunSayisi; i++)

{

DateTime kontrolTarihi = baslangic.Value.AddDays(i);

if (kontrolTarihi.Date == Days.Date)

{

// Eğer Days, izin tarihleri arasında ise izin verme (false döndürme)

MessageBox.Show("Sicil " + Sicil + " Days " + Days);

return false;

}

}

}

// İzin sorgulama işlemi başarılıysa true döndür

return true;

}

public static void vardiyaEkleme(double Sicil, DateTime Days, String saat, String location)

{

ConnectionStrings.ConnectionControl();

SqlCommand sqlCommand = new SqlCommand("Insert into Shifts values(@registrantId,@employeeId,@dateOfRegistration,@date,@location,@hours,@isNew)", ConnectionStrings.\_sqlConnection);

sqlCommand.Parameters.AddWithValue("@registrantId", UserInformation.User.Id);

sqlCommand.Parameters.AddWithValue("@employeeId", Sicil);

sqlCommand.Parameters.AddWithValue("@dateOfRegistration", Days);

sqlCommand.Parameters.AddWithValue("@date", Days);

sqlCommand.Parameters.AddWithValue("@location", location);

sqlCommand.Parameters.AddWithValue("@hours", saat);

sqlCommand.Parameters.AddWithValue("@isNew", false); // true veya false gibi bir boolean değer

sqlCommand.ExecuteNonQuery();

ConnectionStrings.\_sqlConnection.Close();

}

private void baslangıc\_ValueChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

Main MainCheck = (Main)Application.OpenForms["Main"];

if (MainCheck != null)

{

MainCheck.Show();

}

else

{

Main main = new Main();

main.Show();

}

this.Close();

}

}

}

# Roller Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class Roles : Form

{

public Roles()

{

InitializeComponent();

}

RoleDal \_roleDal = new RoleDal();

private void Roles\_Load(object sender, EventArgs e)

{

GetList();

}

public void GetList()

{

dgv\_roles.Rows.Clear();

var roles = \_roleDal.GetAll();

foreach (var role in roles)

{

dgv\_roles.Rows.Add(role.Id, role.Name, role.Add, role.Update, role.Delete, role.Print, role.UsersManagement, role.RolesManagement,role.SeeAllShifts, role.Status);

}

dgv\_roles.ClearSelection();

}

private void dgv\_roles\_CellClick(object sender, DataGridViewCellEventArgs e)

{

txt\_name.Text = dgv\_roles.CurrentRow.Cells[1].Value.ToString();

chb\_add.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[2].Value);

chb\_update.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[3].Value);

chb\_delete.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[4].Value);

chb\_print.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[5].Value);

chb\_usersManagement.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[6].Value);

chb\_rolesManagement.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[7].Value);

chb\_seeAllShifts.Checked = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[8].Value);

}

private void btn\_add\_Click(object sender, EventArgs e)

{

if (txt\_name.Text != "")

{

Role role = new Role

{

Name = txt\_name.Text,

Add = Convert.ToBoolean(chb\_add.Checked),

Update = Convert.ToBoolean(chb\_update.Checked),

Delete = Convert.ToBoolean(chb\_delete.Checked),

Print = Convert.ToBoolean(chb\_print.Checked),

UsersManagement = Convert.ToBoolean(chb\_usersManagement.Checked),

RolesManagement = Convert.ToBoolean(chb\_rolesManagement.Checked),

SeeAllShifts = Convert.ToBoolean(chb\_seeAllShifts.Checked),

Status = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[9].Value)

};

\_roleDal.Add(role);

GetList();

Clear();

}

else

{

MessageBox.Show("Rol adı boş olamaz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_update\_Click(object sender, EventArgs e)

{

if (dgv\_roles.SelectedRows.Count != 0)

{

if (txt\_name.Text != "")

{

Role role = new Role

{

Id = Convert.ToInt32(dgv\_roles.CurrentRow.Cells[0].Value),

Name = txt\_name.Text,

Add = Convert.ToBoolean(chb\_add.Checked),

Update = Convert.ToBoolean(chb\_update.Checked),

Delete = Convert.ToBoolean(chb\_delete.Checked),

Print = Convert.ToBoolean(chb\_print.Checked),

UsersManagement = Convert.ToBoolean(chb\_usersManagement.Checked),

RolesManagement = Convert.ToBoolean(chb\_rolesManagement.Checked),

SeeAllShifts = Convert.ToBoolean(chb\_seeAllShifts.Checked),

Status = Convert.ToBoolean(dgv\_roles.CurrentRow.Cells[9].Value)

};

\_roleDal.Update(role);

GetList();

Clear();

}

else

{

MessageBox.Show("Rol adı boş olamaz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Güncellenecek rolu seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (dgv\_roles.SelectedRows.Count != 0)

{

if (MessageBox.Show("Rolu silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

\_roleDal.Delete(Convert.ToInt32(dgv\_roles.CurrentRow.Cells[0].Value));

GetList();

Clear();

}

}

else

{

MessageBox.Show("Silinecek rolu seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

public void Clear()

{

txt\_name.Clear();

chb\_add.Checked = false;

chb\_update.Checked = false;

chb\_delete.Checked = false;

chb\_print.Checked = false;

chb\_usersManagement.Checked = false;

chb\_rolesManagement.Checked = false;

chb\_seeAllShifts.Checked = false;

}

}

}

# Resmi Tatiller Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Windows.Forms;

namespace PersonelVardiyaOtomasyonu

{

public partial class Holidays : Form

{

HolidayDal \_holidayDal = new HolidayDal();

public Holidays()

{

InitializeComponent();

}

private void Holidays\_Load(object sender, EventArgs e)

{

GetList();

}

public void GetList()

{

dgv\_holidays.Rows.Clear();

var holidays = \_holidayDal.GetAll();

foreach (var holiday in holidays)

{

dgv\_holidays.Rows.Add(

holiday.Id,

holiday.Name,

holiday.Date.ToShortDateString()

);

}

dgv\_holidays.ClearSelection();

}

private void btn\_add\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrWhiteSpace(txt\_name.Text) && !string.IsNullOrWhiteSpace(txt\_date.Text))

{

Holiday holiday = new Holiday

{

Name = txt\_name.Text,

Date = Convert.ToDateTime(txt\_date.Text)

};

\_holidayDal.Add(holiday);

GetList();

Clear();

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_update\_Click(object sender, EventArgs e)

{

if (dgv\_holidays.SelectedRows.Count != 0)

{

if (!string.IsNullOrWhiteSpace(txt\_name.Text) && !string.IsNullOrWhiteSpace(txt\_date.Text))

{

Holiday holiday = new Holiday

{

Id = Convert.ToInt32(dgv\_holidays.CurrentRow.Cells[0].Value),

Name = txt\_name.Text,

Date = Convert.ToDateTime(txt\_date.Text)

};

\_holidayDal.Update(holiday);

GetList();

Clear();

}

else

{

MessageBox.Show("Gerekli alanları doldurunuz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Güncellenecek kullanıcıyı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (dgv\_holidays.SelectedRows.Count != 0)

{

if (MessageBox.Show("Kullanıcıyı silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

\_holidayDal.Delete(Convert.ToInt32(dgv\_holidays.CurrentRow.Cells[0].Value));

GetList();

Clear();

}

}

else

{

MessageBox.Show("Silinecek kullanıcıyı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void dgv\_holidays\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

object nameCellValue = dgv\_holidays.Rows[e.RowIndex].Cells[1].Value;

object dateCellValue = dgv\_holidays.Rows[e.RowIndex].Cells[2].Value;

if (nameCellValue != null && dateCellValue != null)

{

txt\_name.Text = nameCellValue.ToString();

txt\_date.Text = dateCellValue.ToString();

}

else

{

// İlgili hücrelerde değer bulunmamaktaysa yapılacak işlem veya mesajı buraya ekleyebilirsiniz.

MessageBox.Show("Seçili hücrelerde değer bulunmamaktadır.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

public void Clear()

{

txt\_name.Clear();

txt\_date.Value = DateTime.Now;

}

}

}

# İzin Ekranı

using PersonelVardiyaOtomasyonu.Class.DAL;

using PersonelVardiyaOtomasyonu.Class.Helper;

using PersonelVardiyaOtomasyonu.Entities;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace PersonelVardiyaOtomasyonu

{

public partial class Leaves : Form

{

LeavesDal \_leaveDal = new LeavesDal();

public Leaves()

{

InitializeComponent();

List<User> userList = GetAll();

cb\_users.DisplayMember = "NameSurname";

cb\_users.ValueMember = "Id"; // veya istediğiniz başka bir özellik

cb\_users.DataSource = userList;

}

public List<User> GetAll()

{

ConnectionStrings.ConnectionControl();

SqlCommand sqlCommand = new SqlCommand("Select \* from Users", ConnectionStrings.\_sqlConnection);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

List<User> users = new List<User>();

while (sqlDataReader.Read())

{

User user = new User

{

Id = Convert.ToInt32(sqlDataReader["Id"]),

RoleId = Convert.ToInt32(sqlDataReader["RoleId"]),

IdentificationNumber = sqlDataReader["IdentificationNumber"].ToString(),

Name = sqlDataReader["Name"].ToString(),

Surname = sqlDataReader["Surname"].ToString(),

Password = sqlDataReader["Password"].ToString(),

Address = sqlDataReader["Address"].ToString(),

EMail = sqlDataReader["E-Mail"].ToString(),

PhoneNumber = sqlDataReader["PhoneNumber"].ToString(),

leave\_day\_1 = sqlDataReader["Leave\_day\_1"].ToString(),

leave\_day\_2 = sqlDataReader["Leave\_day\_2"].ToString(),

Status = Convert.ToBoolean(sqlDataReader["Status"]),

NameSurname = sqlDataReader["Name"].ToString() + " " + sqlDataReader["Surname"].ToString()

};

users.Add(user);

}

sqlDataReader.Close();

ConnectionStrings.\_sqlConnection.Close();

return users;

}

private void Leaves\_Load(object sender, EventArgs e)

{

GetList();

}

public void GetList()

{

dgv\_leaves.Rows.Clear();

var leaves = \_leaveDal.GetAll();

foreach (var leave in leaves)

{

dgv\_leaves.Rows.Add(

leave.id,

leave.name,

leave.surname,

leave.sicil\_no,

leave.start\_date.ToShortDateString(),

leave.end\_date.ToShortDateString()

);

}

dgv\_leaves.ClearSelection();

}

private void btn\_add\_Click(object sender, EventArgs e)

{

// ComboBox'ta seçili öğe var mı kontrolü

if (cb\_users.SelectedItem != null && cb\_users.SelectedItem is User selectedUser)

{

Leave leave = new Leave

{

name = selectedUser.Name,

surname = selectedUser.Surname,

sicil\_no = selectedUser.Id, // Varsa IdentificationNumber kullanabilirsiniz.

start\_date = Convert.ToDateTime(txt\_start\_date.Text),

end\_date = Convert.ToDateTime(txt\_end\_date.Text)

};

\_leaveDal.Add(leave);

GetList();

Clear();

}

else

{

MessageBox.Show("Lütfen bir kullanıcı seçin!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (dgv\_leaves.SelectedRows.Count != 0)

{

if (MessageBox.Show("Kullanıcıyı silmek istediğinizden emin misiniz?", "Bilgi", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

\_leaveDal.Delete(Convert.ToInt32(dgv\_leaves.CurrentRow.Cells[0].Value));

GetList();

Clear();

}

}

else

{

MessageBox.Show("Silinecek kullanıcıyı seçiniz!", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void dgv\_leaves\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

object startDateCellValue = dgv\_leaves.Rows[e.RowIndex].Cells[4].Value;

object endDateCellValue = dgv\_leaves.Rows[e.RowIndex].Cells[5].Value;

if (startDateCellValue != null && endDateCellValue != null)

{

txt\_start\_date.Text = startDateCellValue.ToString();

txt\_end\_date.Text = endDateCellValue.ToString();

}

else

{

// İlgili hücrelerde değer olmadığında yapılacak işlem veya mesajı buraya ekleyebilirsiniz.

MessageBox.Show("Seçili hücrelerde değer bulunmamaktadır.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

public void Clear()

{

txt\_start\_date.Value = DateTime.Now;

txt\_end\_date.Value = DateTime.Now;

}

private void tableLayoutPanel4\_Paint(object sender, PaintEventArgs e)

{

}

private void tableLayoutPanel5\_Paint(object sender, PaintEventArgs e)

{

}

}

}