T.C. ONDOKUZ MAYIS ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

2024 – 2025 GÜZ DÖNEMİ

VERİ TABANI YÖNETİM SİSTEMLERİ DERSİ FİNAL ÖDEV RAPORU PROJE YÖNETİM OTOMASYONU

21060666 Nuri Can ACAR

21060612 Mehmet Akif CEBECİ

22060653 Eyüphan BİNİCİ

22060674 Murat GÜMÜŞ

SAMSUN-2024

Contents

Giriş

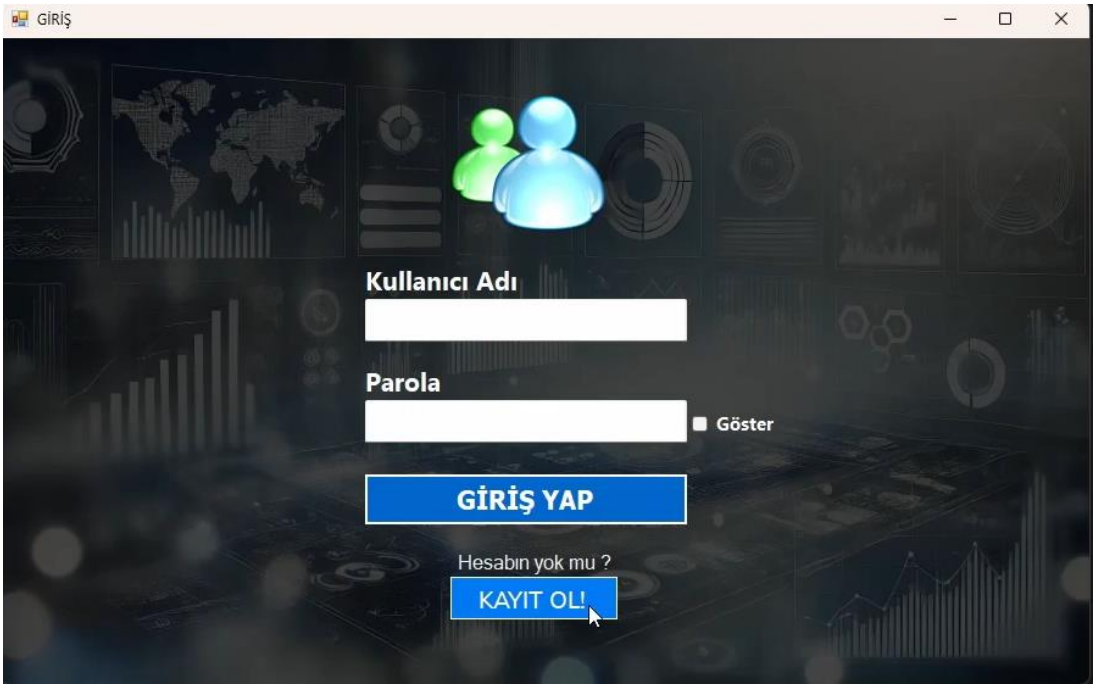
Proje Amacı:

Bu çalışma, kullanıcıların proje, görev ve çalışan yönetimini etkin bir şekilde gerçekleştirebilmesi amacıyla geliştirilmiş bir proje yönetim sistemini tanıtmaktadır. Sistem, kullanıcıların projelerle ilgili görevler ve çalışanları ekleme, silme ve güncelleme işlemlerini yapabilmesine olanak tanımaktadır. Proje kapsamında oluşturulan veri tabanı yapısı, UML diyagramları ve ilişki şemaları ile detaylandırılmış, temel işlevler ise proje tanımlama, çalışan ve görev yönetimi, görev durumlarının takibi ve kontrolü gibi fonksiyonlar üzerinden açıklanmıştır. Ayrıca sistemin kullanıcı arayüzüne ait ekran görüntüleri paylaşılmış, SQL sorguları ile çalışan detayları, kayıt ve giriş işlemleri, görev durumları ve durum değişimi gibi işlemleri gerçekleştiren kodlara yer verilmiştir. Projenin tüm kod yapıları açıklamalarıyla birlikte detaylandırılmış olup, ekler bölümünde kaynakça, GitHub bağlantısı ve YouTube anlatım bağlantıları sunulmuştur. Bu sistem, proje yönetim süreçlerini optimize etmek ve kullanıcıların iş akışını verimli hale getirmek amacıyla tasarlanmıştır.

Kodlar ve Arayüz

GİRİŞ KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class LoginPage : Form
    {
        private const string ConnectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';";
        MySqlConnection connection = new MySqlConnection(ConnectionString);
        MySqlCommand command;
        MySqlDataReader dataReader;
        string loginQuery = "SELECT * FROM Users " +
            "WHERE UserName = @UserName AND UserPassword = @UserPassword";
        public LoginPage()
        {
            InitializeComponent();
        }

        private void register_Click(object sender, EventArgs e)
        {
            RegisterPage registerPage = new RegisterPage();
            this.Hide();
            registerPage.ShowDialog();
        }

        private void hideShow_CheckedChanged(object sender, EventArgs e)
        {
            loginUserPassword.UseSystemPasswordChar = hideShow.Checked;
            hideShow.Text = hideShow.Checked ? "Gizle" : "Göster";
        }

        private void login_Click(object sender, EventArgs e)
        {
            try
            {
                string userName = loginUserName.Text;
                string password = loginUserPassword.Text;

                if (string.IsNullOrEmpty(userName) || string.IsNullOrEmpty(password))
                {
                    MessageBox.Show("Kullan□c□ ad□ veya şifre boş olamaz.");
                    return;
                }

                using (connection)
                {
                    connection.Open();

                    command = new MySqlCommand(loginQuery, connection);
                    command.Parameters.AddWithValue("@UserName", userName);
                    command.Parameters.AddWithValue("@UserPassword", password);

                    using (dataReader = command.ExecuteReader())
                    {
                        if (dataReader.Read())
                        {
                            int userId = Convert.ToInt32(dataReader["UserId"]);
                            // Başarılı giriş durumunda MainPage formunu göster
                            MainPage mainPage = new MainPage(userId);
                            this.Hide();
                            mainPage.ShowDialog();
                        }
                        else
                        {
                            MessageBox.Show("Kullan□c□ Bulunamad□");
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Bir hata oluştu: " + ex.Message);
            }
        }
    }
}
```



1. Kayıt Olma İşlemi (register_Click Fonksiyonu)

Kullanıcıların sisteme yeni bir hesap kaydı oluşturabilmeleri için "Kayıt Ol" butonu tasarlanmıştır. Bu butona tıklanması durumunda, RegisterPage adlı yeni bir form oluşturularak kullanıcıya kayıt işlemlerini gerçekleştirebileceği bir arayüz sunulur. Bu süreçte mevcut LoginPage formu gizlenir, böylece kullanıcı yalnızca kayıt formuyla etkileşimde bulunabilir. Bu tasarım, kullanıcı deneyimini kolaylaştırmak ve işlem sırasında dikkati odaklamak amacıyla uygulanmıştır.

---

2. Şifre Gösterme/Gizleme Seçeneği hideShow_CheckedChanged Fonksiyonu)

Kullanıcıların giriş sırasında yazdıkları şifrenin gizliliğini kontrol edebilmesi için bir "Şifre Göster/Gizle" seçeneği eklenmiştir. Checkbox işaretlendiğinde, şifre alanında yazılan karakterler düz metin olarak görünür hale gelir; işaret kaldırıldığında ise şifre karakterleri gizlenir. Bu özellik, şifre doğruluğunu manuel olarak kontrol etmek isteyen kullanıcılar için pratik bir çözüm sunar. Ayrıca, checkbox'ın üzerindeki metin, kullanıcının işlem durumuna göre "Göster" veya "Gizle" olarak dinamik şekilde güncellenir.

---

3. Giriş İşlemi (login_Click Fonksiyonu)

Kullanıcıların mevcut hesap bilgileriyle sisteme giriş yapabilmesi için "Giriş Yap" butonu işlevsel hale getirilmiştir. Bu butona tıklandığında, kullanıcının girdiği kullanıcı adı ve şifre bilgileri alınır. Eğer bu alanlardan herhangi biri boş bırakılmışsa, kullanıcıya bir uyarı mesajı gösterilir. Giriş bilgileri uygun şekilde doldurulmuşsa, veritabanına bağlanılarak Users tablosunda ilgili kullanıcıyı doğrulamak için bir sorgu çalıştırılır. Doğrulama başarılı olursa, kullanıcının kimliğini belirleyen bir ID alınarak sistemin ana sayfası olan MainPage formu açılır. Bu form, giriş yapan kullanıcının verilerine özgü bilgileri gösterecek şekilde tasarlanmıştır. Yanlış kullanıcı adı veya şifre girilmesi durumunda, kullanıcıya bilgilendirici bir hata mesajı gösterilir. Ek olarak, herhangi bir teknik hata meydana gelirse, kullanıcı dostu bir şekilde durum rapor edilir ve hata detayları gösterilir. Bu süreç, kullanıcı güvenliğini ve sistemin hata toleransını artırmayı hedefler.

KAYIT OL KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class RegisterPage : Form
    {
        private const string ConnectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';";
        MySqlConnection connection = new MySqlConnection(ConnectionString);
        MySqlCommand command;

        string registerQuery = "INSERT INTO Users (UserFirstName, UserLastName, UserName, UserPassword) " +
                               "VALUES (@UserFirstName, @UserLastName, @UserName, @UserPassword)";

        public RegisterPage()
        {
            InitializeComponent();
        }

        private void turnBack_Click(object sender, EventArgs e)
        {
            LoginPage loginPage = new LoginPage();
            this.Hide();
            loginPage.ShowDialog();
        }

        private void register_Click(object sender, EventArgs e)
        {
            try
            {
                string userFirstName = this.newUserFirstName.Text;
                string userLastName = this.newUserLastName.Text;
                string userName = this.newUserName.Text;
                string userPassword = this.newUserPassword.Text;

                if (string.IsNullOrEmpty(userFirstName) || string.IsNullOrEmpty(userLastName))
                {
                    MessageBox.Show("Kullan□c□ ad□ veya şifre boş olamaz.");
                    return;
                }

                using (connection)
                {
                    connection.Open();

                    command = new MySqlCommand(registerQuery, connection);
                    command.Parameters.AddWithValue("@UserFirstName", userFirstName);
                    command.Parameters.AddWithValue("@UserLastName", userLastName);
                    command.Parameters.AddWithValue("@UserName", userName);
                    command.Parameters.AddWithValue("@UserPassword", userPassword);

                    int rowsAffected = command.ExecuteNonQuery();

                    if (rowsAffected > 0)
                    {
                        MessageBox.Show("Başar□yla Kay□t Olundu");
                    }
                    else
                    {
                        MessageBox.Show("Kay□t s□ras□nda bir hata oluştu.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("An error occurred: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void hideShow_CheckedChanged(object sender, EventArgs e)
        {
            if (hideShow.CheckState == CheckState.Checked)
            {
                newUserPassword.UseSystemPasswordChar = false;
                hideShow.Text = "Göster";
            }
            else if (hideShow.CheckState == CheckState.Unchecked)
            {
                newUserPassword.UseSystemPasswordChar = true;
                hideShow.Text = "Gizle";
            }
        }
    }
}
```

1. turnBack_Click Fonksiyonu
Bu fonksiyon, kullanıcının kayıt ekranından çıkıp giriş ekranına dönmesini sağlar. Butona tıklanıldığında, mevcut RegisterPage formu gizlenir ve LoginPage formu açılarak kullanıcı giriş ekranına yönlendirilir. Bu işlem, kullanıcı deneyimini kolaylaştırmak ve yanlışlıkla kayıt ekranına giren kullanıcıların hızlı bir şekilde geri dönmesini sağlamak amacıyla tasarlanmıştır.

---

2. register_Click Fonksiyonu
Kullanıcının sisteme yeni bir hesap kaydı yapmasını sağlayan bu fonksiyon, ilk olarak formdaki ad, soyad, kullanıcı adı ve şifre alanlarının boş olup olmadığını kontrol eder. Eğer herhangi bir alan boş bırakılmışsa, işlem durdurulur ve kullanıcıya uyarı mesajı gösterilir. Veriler eksiksiz ise, bir MySQL bağlantısı açılarak Users tablosuna bu bilgiler kaydedilir.

Kayıt işlemi başarılı olduğunda kullanıcıya "Başarıyla Kayıt Olundu" mesajı gösterilir, aksi bir durumda ise hata mesajı görüntülenir. Bu süreç, kullanıcı bilgilerini güvenli bir şekilde veritabanına eklemek ve oluşabilecek hataları kullanıcıya bildirmek üzerine yapılandırılmıştır.

---

3. hideShow_CheckedChanged Fonksiyonu
Şifre giriş alanındaki karakterlerin görünürlüğünü kontrol eden bu fonksiyon, kullanıcı deneyimini artırmak için tasarlanmıştır. Checkbox işaretlendiğinde şifre düz metin olarak görünür hale gelir, işaret kaldırıldığında ise yıldızlı karakterlerle gizlenir. Checkbox metni bu duruma göre dinamik olarak "Göster" veya "Gizle" şeklinde güncellenir. Bu özellik, şifre yazarken yanlış girişleri fark etme ve doğrulama kolaylığı sunar.

ANA EKRAN KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class MainPage : Form
    {
        private int userId;

        public MainPage(int userId)
        {
            InitializeComponent();
            this.userId = userId;
        }
        private void projects_Click(object sender, EventArgs e)
        {
            ProjectPage projectPage = new ProjectPage(userId);
            this.Hide();
            projectPage.ShowDialog();
        }

        private void tasks_Click(object sender, EventArgs e)
        {
            TaskPage teskPage = new TaskPage(userId);
            this.Hide();
            teskPage.ShowDialog();
        }

        private void employees_Click(object sender, EventArgs e)
        {
            EmployeePage employeePage = new EmployeePage(userId);
            this.Hide();
            employeePage.ShowDialog();
        }

        private void logOut_Click(object sender, EventArgs e)
        {
            LoginPage loginPage = new LoginPage();
            this.Hide();
            loginPage.ShowDialog();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

MainPage Sınıfı
Kullanıcı giriş yaptıktan sonra sistemin ana ekranını temsil eder ve kullanıcıya projeler, görevler, çalışanlar gibi çeşitli alt sayfalara erişim sağlar. Kullanıcıya ait userId bilgisi, giriş sırasında alınır ve bu formda saklanarak diğer sayfalara aktarılır.

---

projects_Click Fonksiyonu
Bu fonksiyon, kullanıcı "Projeler" butonuna tıkladığında çalışır. Kullanıcının kimliğini (userId) içeren bir ProjectPage nesnesi oluşturulur ve mevcut form gizlenerek yeni form açılır. Bu sayede, kullanıcının sadece kendi projelerine erişimi sağlanır.

---

tasks_Click Fonksiyonu
Kullanıcı "Görevler" butonuna tıkladığında, ilgili görev sayfasına yönlendirilmesini sağlar. Kullanıcının userId bilgisi, yeni form olan TaskPage'e aktarılır. Bu form açılırken mevcut MainPage formu gizlenir.

---

employees_Click Fonksiyonu
"Çalışanlar" butonuna tıklandığında çalışır ve çalışanlarla ilgili bilgilerin bulunduğu EmployeePage formunu açar. Kullanıcı kimliği, çalışan sayfasına iletilerek kullanıcıya özel erişim kontrol edilir. Ana form geçici olarak gizlenir.

---

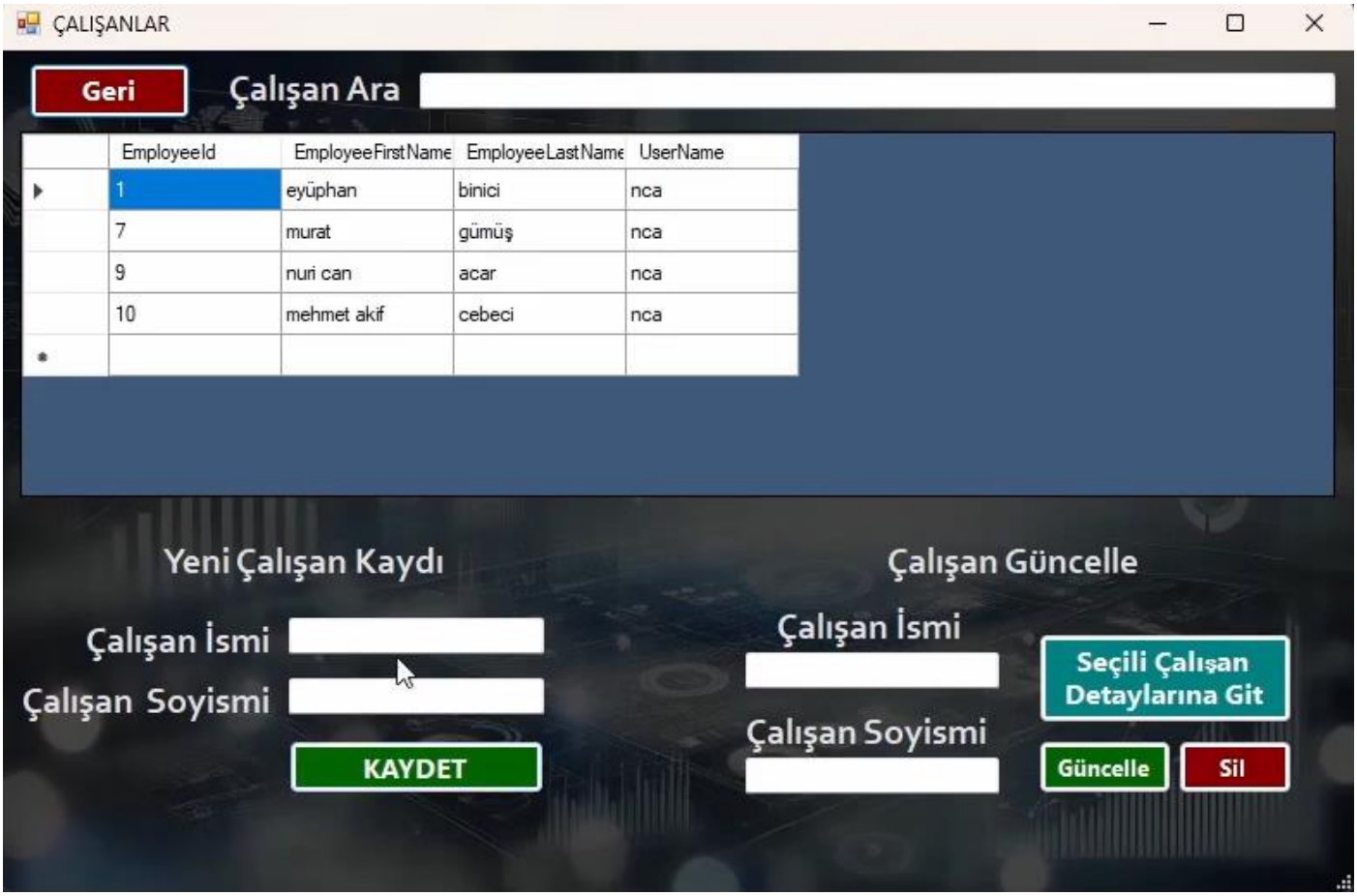
logOut_Click Fonksiyonu
Bu fonksiyon, kullanıcının sistemden çıkış yapmasını sağlar. Çıkış işlemi sırasında giriş ekranı (LoginPage) açılır ve mevcut form gizlenir. Kullanıcı, tekrar giriş yaparak sisteme erişim sağlayabilir.

ÇALIŞANLAR SAYFASI KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class EmployeePage : Form
    {
        private const string ConnectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';";
        private int userId;
        private int employeeId;
        MySqlConnection connection = new MySqlConnection(ConnectionString);
        MySqlCommand command;
        DataTable dataTable;
        DataView dataView;
        string addEmployeeQuery = "INSERT INTO Employee (EmployeeFirstName, EmployeeLastName, UserId) " +
                          "VALUES (@EmployeeFirstName, @EmployeeLastName, @UserId)";
        string getDataQuery = "SELECT E.EmployeeId, E.EmployeeFirstName, E.EmployeeLastName, U.UserName " +
            "FROM vtys.employee E " +
            "JOIN vtys.users U ON E.UserId = U.UserId " +
            "WHERE E.UserId = @userId";
        string deleteEmployeeQuery = "DELETE FROM Employee WHERE EmployeeId = @EmployeeId";
        string updateEmployeeQuery = "UPDATE Employee " +
                    "SET EmployeeFirstName = @EmployeeFirstName, EmployeeLastName = @EmployeeLastName " +
                    "WHERE EmployeeId = @EmployeeId";
        public EmployeePage(int userId)
        {
            InitializeComponent();
            this.userId = userId;

        }
        private void newEmployee_Click(object sender, EventArgs e)
        {
            try
            {
                string employeeFirstName = addEmployeeFirstName.Text;
                string employeeLastName = addEmployeeLastName.Text;

                if (string.IsNullOrEmpty(employeeFirstName) || string.IsNullOrEmpty(employeeLastName))
                {
                    MessageBox.Show("Kullan□c□ ad□ veya şifre boş olamaz.");
                    return;
                }

                using (MySqlConnection connection = new MySqlConnection(ConnectionString))
                {
                    connection.Open();
                    try
                    {
                        command = new MySqlCommand(addEmployeeQuery, connection);
                        command.Parameters.AddWithValue("@EmployeeFirstName", employeeFirstName);
                        command.Parameters.AddWithValue("@EmployeeLastName", employeeLastName);
                        command.Parameters.AddWithValue("@UserId", userId); // Corrected parameter name

                        // Execute the query
                        command.ExecuteNonQuery();


                        MessageBox.Show("Başar□yla Kay□t Olundu");
                        // Optionally, you can clear the textboxes after successful insertion
                        addEmployeeFirstName.Text = "";
                        addEmployeeLastName.Text = "";
                        GetData();



                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(""+ex);
                    }

                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Bir hata oluştu: " + ex);
            }
        }
```

```
void GetData()
{
    try
    {
        dataTable = new DataTable();

        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
            connection.Open();

            // Parametre kullanarak sorguyu oluştur
            MySqlCommand command = new MySqlCommand(getDataQuery, connection);
            command.Parameters.AddWithValue("@userId", userId);

            using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))
            {
                adapter.Fill(dataTable);
            }
        }

        employeeList.DataSource = dataTable;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error fetching data: " + ex.Message);
    }

}

private void EmployeePage_Load(object sender, EventArgs e)
{
    GetData();
    updateEmployeeFirstName.Text = "";
    updateEmployeeLastName.Text = "";
}

private void turnBack_Click(object sender, EventArgs e)
{
    MainPage mainPage = new MainPage(userId);
    this.Hide();
    mainPage.ShowDialog();
}

private void employeeList_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    updateEmployeeFirstName.Text = employeeList.CurrentRow.Cells[1].Value.ToString();
    updateEmployeeLastName.Text = employeeList.CurrentRow.Cells[2].Value.ToString();
}

private void deleteEmployee_Click(object sender, EventArgs e)
{
    try
    {
        if (employeeList.SelectedRows.Count > 0)
        {
            // DataGridView'de seçili satır varsa
            int selectedRowIndex = employeeList.SelectedRows[0].Index;

            // Doğru sütun adını bulun
            string columnName = "EmployeeId"; // Bu adı DataGridView'de bulunan sütun adıyla değiştirin

            int employeeId = Convert.ToInt32(employeeList.Rows[selectedRowIndex].Cells[columnName].Value);

            using (MySqlConnection connection = new MySqlConnection(ConnectionString))
            {
                connection.Open();

                using (MySqlCommand command = new MySqlCommand(deleteEmployeeQuery, connection))
                {
                    command.Parameters.AddWithValue("@EmployeeId", employeeId);
                    command.ExecuteNonQuery();
                }
            }
```

```
        private void updateEmployee_Click(object sender, EventArgs e)
        {
            try
            {
                int selectedRowIndex = employeeList.SelectedRows[0].Index;

                string columnName = "EmployeeId"; // Bu adı DataGridView'de bulunan sütun adıyla değiştirin

                employeeId = Convert.ToInt32(employeeList.Rows[selectedRowIndex].Cells[columnName].Value);

                command = new MySqlCommand(updateEmployeeQuery, connection);
                command.Parameters.AddWithValue("@EmployeeFirstName", updateEmployeeFirstName.Text);
                command.Parameters.AddWithValue("@EmployeeLastName", updateEmployeeLastName.Text);
                command.Parameters.AddWithValue("@EmployeeId", employeeId);

                connection.Open();
                command.ExecuteNonQuery();
                connection.Close();
                GetData();
                MessageBox.Show("Guncellendi");
            }
            catch (Exception)
            {

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

        private void searcByEmployeeFirstName_TextChanged(object sender, EventArgs e)
        {
            try
            {
                dataView = dataTable.DefaultView;
                dataView.RowFilter = "EmployeeFirstName LIKE '" + searcByEmployeeFirstName.Text + "%'";
                employeeList.DataSource = dataView;
            }
            catch (Exception ex)
            {

                MessageBox.Show("Hata :" + ex);
            }

        }

        private void goToEmployeeDetails_Click(object sender, EventArgs e)
        {
            try
            {
                int selectedRowIndex = employeeList.SelectedRows[0].Index;

                // Doğru sütun adını bulun
                string columnName = "EmployeeId"; // Bu adı DataGridView'de bulunan sütun adıyla değiştirin

                int employeeId = Convert.ToInt32(employeeList.Rows[selectedRowIndex].Cells[columnName].Value);
                EmployeeDetail employeeDetail = new EmployeeDetail(userId, employeeId);
                this.Hide();
                employeeDetail.ShowDialog();
            }
            catch (Exception)
            {

                MessageBox.Show("Calisan secilemedi");
            }

        }
    }
}
```



EmployeePage Sınıfı
Bu sınıf, kullanıcıya çalışanların yönetimi için bir arayüz sağlar. Kullanıcı, çalışan ekleme, silme, güncelleme ve listeleme gibi temel işlemleri bu sayfada gerçekleştirebilir. Sınıf aynı zamanda çalışan detaylarına erişim ve çalışanları filtreleme özelliklerini de içerir.

---

newEmployee_Click Fonksiyonu
Kullanıcı, yeni bir çalışan eklemek için bu butona tıkladığında çalışır. İlk olarak çalışan adı ve soyadı girişlerinin boş olup olmadığı kontrol edilir. Bilgiler eksiksiz ise, MySQL veritabanına bir INSERT sorgusu gönderilerek yeni çalışan eklenir. İşlem başarılı olursa, kullanıcıya bir onay mesajı gösterilir ve çalışan listesi yenilenir.

GetData Fonksiyonu
Bu fonksiyon, mevcut kullanıcıya ait tüm çalışanları veritabanından çekmek ve ekrana listelemek için çalışır. Veriler, bir DataTable içine doldurulur ve ardından employeeList adlı DataGridView bileşenine bağlanır. Hata durumunda kullanıcıya bilgilendirici bir mesaj gösterilir.

EmployeePage_Load Fonksiyonu
Sayfa yüklendiğinde çağrılır. Bu fonksiyon, çalışanların listesini yenilemek için GetData fonksiyonunu çağırır ve güncelleme metin kutularını temizler. Bu, sayfanın her açıldığında düzenli ve güncel bir görünüm sağlamasını amaçlar.

---

turnBack_Click Fonksiyonu
Bu buton, kullanıcının ana sayfaya dönmesini sağlar. Tıklandığında, mevcut form gizlenir ve MainPage formu açılır. Bu işlem, kullanıcıya kolay bir gezinme imkanı sunar.

---

employeeList_CellEnter Fonksiyonu
Bu olay, kullanıcı employeeList içinde bir satır seçtiğinde çalışır. Seçilen çalışanın adı ve soyadı bilgileri, güncelleme metin kutularına doldurulur. Böylece kullanıcı, ilgili bilgileri kolayca düzenleyebilir.

---

deleteEmployee_Click Fonksiyonu
Kullanıcı bir çalışanı silmek istediğinde çalışır. Seçilen çalışanın ID'si alınarak bir DELETE sorgusu gönderilir ve kayıt veritabanından silinir. Eğer çalışan bir göreve bağlıysa, kullanıcıya uygun bir hata mesajı gösterilir. Başarılı bir silme işleminde, liste yenilenir.

---

updateEmployee_Click Fonksiyonu
Bu fonksiyon, seçilen çalışanın bilgilerini günceller. Güncelleme sırasında kullanıcı, seçilen satırdaki çalışan bilgilerini düzenleyebilir. Düzenlenen bilgiler bir UPDATE sorgusuyla veritabanına kaydedilir. Başarılı işlem sonrası liste yenilenir ve kullanıcıya onay mesajı gösterilir.

---

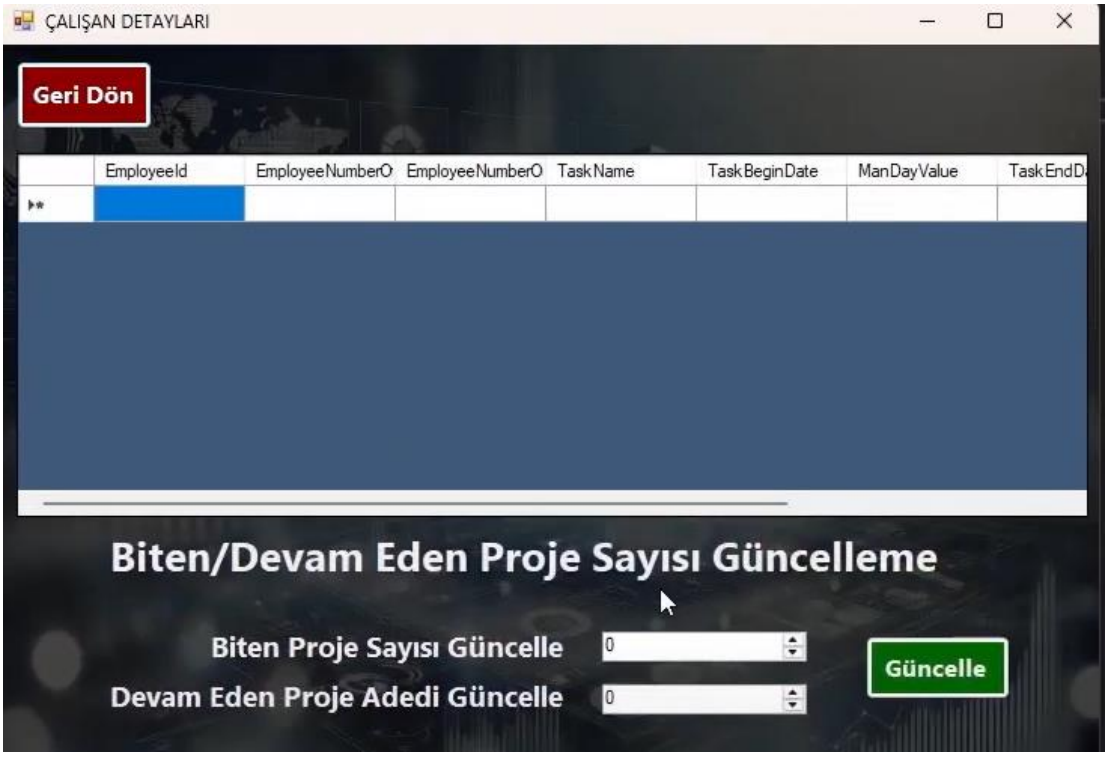
searcByEmplyeeFirstName_TextChanged Fonksiyonu
Kullanıcı, çalışanları ada göre filtrelemek istediğinde çalışır. Metin kutusuna girilen ada göre, DataView aracılığıyla liste filtrelenir. Bu özellik, büyük çalışan listelerinde arama kolaylığı sağlar.

---

goToEmployeeDetails_Click Fonksiyonu
Kullanıcı, seçili çalışanın detay sayfasına gitmek istediğinde çalışır. Seçilen çalışanın ID bilgisi alınarak, EmployeeDetail adlı yeni bir form açılır ve mevcut form gizlenir. Bu özellik, çalışan detaylarının ayrı bir sayfada incelenmesine olanak tanır. Eğer bir çalışan seçilmezse, kullanıcıya uygun bir uyarı mesajı gösterilir.

ÇALIŞAN DETAY SAYFASI KODLARI ve ARAYÜZÜ

```
namespace ProjcetLock.Forms
{
    public partial class EmployeeDetail : Form
    {
        private const string connectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';";
        private int userId;
        private int employeeId;
        private MySqlConnection connection = new MySqlConnection(connectionString);
        private MySqlCommand command;
        private MySqlDataAdapter adapter;
        private DataTable dataTable;
        private string getDataQuery = "SELECT e.EmployeeId, e.EmployeeNumberOfTasksCompleted, e.EmployeeNumberOngoingTasks, t.TaskName, t.TaskBeginDate, t.ManDayValue, t.TaskEndDate,
                    "FROM vtys.Task t " +
                    "LEFT JOIN vtys.Employee e ON t.EmployeeId = e.EmployeeId " +
                    "LEFT JOIN vtys.Project p ON p.TaskId = t.TaskId " +
                    "WHERE t.EmployeeId = @EmployeeId";
        string updateTaskQuery = "UPDsATE Employee " +
                            "SET EmployeeNumberOfTasksCompleted = @EmployeeNumberOfTasksCompleted, EmployeeNumberOngoingTasks = @EmployeeNumberOngoingTasks " +
                            "WHERE EmployeeId = @EmployeeId";
        public EmployeeDetail(int userId, int employeeId)
        {
            InitializeComponent();
            this.userId = userId;
            this.employeeId = employeeId;

        }

        private void turnBack_Click(object sender, EventArgs e)
        {
            EmployeePage employeePage = new EmployeePage(userId);
            this.Hide();
            employeePage.ShowDialog();
        }

        private void EmployeeDetail_Load(object sender, EventArgs e)
        {
            GetData();
        }
        void GetData()
        {
            try
            {
                dataTable = new DataTable();
                using (MySqlConnection connection = new MySqlConnection(connectionString))
                {
                    connection.Open();

                    // Parametre kullanarak sorguyu oluştur

                    command = new MySqlCommand(getDataQuery, connection);
                    command.Parameters.AddWithValue("@EmployeeId", employeeId);

                    adapter = new MySqlDataAdapter(command);
                    adapter.Fill(dataTable);
                }

                employeeTaskList.DataSource = dataTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error fetching data: " + ex);
            }
        }

        private void updateTasks_Click(object sender, EventArgs e)
        {
            using (MySqlConnection connection = new MySqlConnection(connectionString))
            {
                try
                {
                    connection.Open();
                    command = new MySqlCommand(updateTaskQuery, connection);
                    command.Parameters.AddWithValue("@EmployeeNumberOfTasksCompleted", (int)updateEmployeeNumberOfTasksCompleted.Value);
                    command.Parameters.AddWithValue("@EmployeeNumberOngoingTasks", (int)updateEmployeeNumberOngoingTasks.Value);
                    command.Parameters.AddWithValue("@EmployeeId", employeeId);

                    command.ExecuteNonQuery();

                    GetData();
                    MessageBox.Show("Güncellendi");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Lütfen bir satır seçin." + ex, "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
            }

        }
    }
}
```



EmployeeDetail Sınıfı
Bu sınıf, bir çalışanın görev detaylarını yönetmek için tasarlanmıştır. Kullanıcı, bir çalışanın tamamlanan veya devam eden görevlerini görüntüleyebilir ve bu bilgileri güncelleyebilir. Sınıf, çalışan ve görev bilgilerinin veritabanından çekilmesi ve kullanıcı arayüzünde sunulmasını sağlar.

---

turnBack_Click Fonksiyonu
Bu fonksiyon, kullanıcının çalışan detay sayfasından ana çalışan yönetim sayfasına (EmployeePage) dönmesini sağlar. Tıklandığında, mevcut form gizlenir ve EmployeePage formu açılır. Bu işlem, kullanıcıya hızlı ve kolay bir gezinme deneyimi sunar.

EmployeeDetail_Load Fonksiyonu
Sayfa yüklendiğinde çalışır ve GetData fonksiyonunu çağırarak çalışana ait görev verilerini veritabanından çeker. Bu işlem, sayfa yüklendiği anda tüm gerekli bilgilerin görüntülenmesini sağlar.

GetData Fonksiyonu
Bu fonksiyon, çalışanın görev detaylarını veritabanından almak için tasarlanmıştır. MySqlCommand kullanılarak sorgu çalıştırılır ve sonuçlar bir DataTable içine doldurulur. Veriler, employeeTaskList adlı kullanıcı arayüzü bileşeninde (genellikle bir DataGridView) listelenir. Eğer bir hata oluşursa, kullanıcı bilgilendirilir.

---

updateTasks_Click Fonksiyonu
Bu fonksiyon, çalışanın tamamlanan ve devam eden görev sayılarını güncellemek için kullanılır. Kullanıcı, form üzerinde ilgili değerleri girdikten sonra bu güncellemeyi gerçekleştirebilir. MySqlCommand ile güncelleme sorgusu çalıştırılır ve işlem tamamlandıktan sonra görev verileri yenilenir. Başarılı işlem durumunda kullanıcıya bilgilendirme mesajı gösterilir. Eğer bir hata meydana gelirse, kullanıcıya uygun bir uyarı mesajı iletilir.

GÖREVLER SAYFASI KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class TaskPage : Form
    {
        private const string ConnectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';";
        private int userId;
        MySqlConnection connection = new MySqlConnection(ConnectionString);
        MySqlCommand command;
        MySqlDataReader dataReader;
        DataTable dataTable;
        private int selectedEmployeeId;
        string getDataQuery = "SELECT T.TaskId, T.TaskName, T.TaskBeginDate, T.ManDayValue, T.TaskEndDate, T.TaskState, E.EmployeeFirstName, U.UserName " +
                    "FROM vtys.Task T " +
                    "JOIN vtys.Users U ON T.UserId = U.UserId " +
                    "JOIN vtys.Employee E ON T.EmployeeId = E.EmployeeId " +
                    "WHERE T.UserId = @userId";
        string fillComboBoxQuery = "SELECT * FROM employee WHERE UserId = @userId";
        string addTaskQuery = "INSERT INTO Task (TaskName, TaskBeginDate, ManDayValue, TaskEndDate, TaskState, EmployeeId, UserId) " +
                              "VALUES (@TaskName, @TaskBeginDate, @ManDayValue, @TaskEndDate, @TaskState, @EmployeeId, @UserId)";
        string isValidEmployeeIdQuery = "SELECT COUNT(*) FROM Employee WHERE EmployeeId = @EmployeeId AND UserId = @UserId";
        string deleteTaskQuery = "DELETE FROM task WHERE TaskId = @TaskId";
        string updateTaskQuery = "UPDATE Task " +
                                 "SET TaskEndDate = @TaskEndDate, ManDayValue = @ManDayValue, TaskState = @TaskState " +
                                 "WHERE TaskId = @TaskId";

        public TaskPage(int userId)
        {
            InitializeComponent();
            this.userId = userId;
            fillComboBox();
            GetData();
        }

        void GetData()
        {
            try
            {
                dataTable = new DataTable();
                using (MySqlConnection connection = new MySqlConnection(ConnectionString))
                {
                    connection.Open();
                    command = new MySqlCommand(getDataQuery, connection);
                    command.Parameters.AddWithValue("@userId", userId);

                    MySqlDataAdapter adapter = new MySqlDataAdapter(command);
                    adapter.Fill(dataTable);
                }

                taskList.DataSource = dataTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veri çekme hatası: " + ex.Message);
            }
            finally
            {
                connection.Close();
            }
        }

        public void fillComboBox()
        {
            try
            {
                connection.Open();
                // UserId ile eşleşen çalışanları getir
                command = new MySqlCommand(fillComboBoxQuery, connection);
                command.Parameters.AddWithValue("@userId", userId);

                using (dataReader = command.ExecuteReader())
                {
                    while (dataReader.Read())
                    {
                        int employeeId = dataReader.GetInt32(0);
                        string employeeName = dataReader.GetString(1);
                        addEmployeeToTask.Items.Add(new KeyValuePair<int, string>(employeeId, employeeName));
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("ComboBox doldurma hatası: " + ex.Message);
            }
            finally
            {
                connection.Close();
            }
        }
```

```
private void addTask_Click(object sender, EventArgs e)
{
    try
    {
        string taskName = addTaskName.Text;
        DateTime taskBeginDate = addTaskBeginDate.Value;
        int manDayValue = (int)addManDayValue.Value;
        DateTime taskEndDate = addTaskEndDate.Value;
        string taskState = addTaskState.Text;
        int employeeId = selectedEmployeeId;

        if (string.IsNullOrEmpty(taskName) || string.IsNullOrEmpty(taskState))
        {
            MessageBox.Show("Liste boş olamaz.");
            return;
        }

        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
            connection.Open();

            if (!IsValidEmployeeId(employeeId, connection))
            {
                MessageBox.Show("Geçerli bir çalışan seçilmelidir.");
                return;
            }

            command = new MySqlCommand(addTaskQuery, connection);
            command.Parameters.AddWithValue("@TaskName", taskName);
            command.Parameters.AddWithValue("@TaskBeginDate", taskBeginDate);
            command.Parameters.AddWithValue("@ManDayValue", manDayValue);
            command.Parameters.AddWithValue("@TaskEndDate", taskEndDate);
            command.Parameters.AddWithValue("@TaskState", taskState);
            command.Parameters.AddWithValue("@EmployeeId", employeeId);
            command.Parameters.AddWithValue("@UserId", userId);

            int rowsAffected = command.ExecuteNonQuery();

            if (rowsAffected > 0)
            {
                MessageBox.Show("Başarıyla Kayıt Olundu");
                addTaskName.Text = "";
                GetData();
            }
            else
            {
                MessageBox.Show("Kayıt sırasında bir hata oluştu.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Bir hata oluştu: " + ex.Message);
    }
}

private bool IsValidEmployeeId(int employeeId, MySqlConnection connection)
{
    try
    {
        using (command = new MySqlCommand(isValidEmployeeIdQuery, connection))
        {
            command.Parameters.AddWithValue("@EmployeeId", employeeId);
            command.Parameters.AddWithValue("@UserId", userId);

            int count = Convert.ToInt32(command.ExecuteScalar());
            return count > 0;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Geçerli bir EmployeeId kontrolü sırasında hata oluştu: " + ex.Message);
        return false;
    }
}
private void TaskPage_Load(object sender, EventArgs e)
{
    GetData();

}
private void turnBack_Click(object sender, EventArgs e)
{
    MainPage mainPage = new MainPage(userId);
    this.Hide();
    mainPage.ShowDialog();

}
```

```
private void deleteTask_Click(object sender, EventArgs e)
{
    try
    {
        if (taskList.SelectedRows.Count > 0)
        {
            int selectedRowIndex = taskList.SelectedRows[0].Index;

            string columnName = "TaskId";

            int taskId = Convert.ToInt32(taskList.Rows[selectedRowIndex].Cells[columnName].Value);

            using (MySqlConnection connection = new MySqlConnection(ConnectionString))
            {
                connection.Open();

                command = new MySqlCommand(deleteTaskQuery, connection);
                command.Parameters.AddWithValue("@TaskId", taskId);

                command.ExecuteNonQuery();
            }

            GetData();
            MessageBox.Show("Kayıt Silindi");
        }
        else
        {
            MessageBox.Show("Lütfen bir satır seçin.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Bu Gorev bir Projeye bagimli once o Projeyi siliniz.");

    }

}

private void searcByTaskName_TextChanged(object sender, EventArgs e)
{
    try
    {
        DataView dv = dataTable.DefaultView;
        dv.RowFilter = "TaskName LIKE '" + searcByTaskName.Text + "%'";
        taskList.DataSource = dv;
    }
    catch (Exception ex)
    {

        MessageBox.Show("Hata :" + ex);
    }
}

private void updateTask_Click(object sender, EventArgs e)
{
    try
    {
        connection.Open();



        int selectedRowIndex = taskList.SelectedRows[0].Index;
        string columnName = "TaskId";
        int taskId = Convert.ToInt32(taskList.Rows[selectedRowIndex].Cells[columnName].Value);



        using (MySqlCommand command = new MySqlCommand(updateTaskQuery, connection))
        {
            command.Parameters.AddWithValue("@TaskEndDate", updateTaskEndDate.Value.ToString("yyyy-MM-dd"));
            command.Parameters.AddWithValue("@ManDayValue", (int)updateManDayValue.Value);
            command.Parameters.AddWithValue("@TaskState", updateTaskState.Text);
            command.Parameters.AddWithValue("@TaskId", taskId);

            command.ExecuteNonQuery();
        }

        GetData();
        MessageBox.Show("Güncellendi");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Lütfen bir satır seçin." + ex, "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    finally
    {
        connection.Close();
    }
}
```

TaskPage Sınıfı

Bu sınıf, kullanıcıların görevleri yönetmesi için tasarlanmıştır. Kullanıcı, görev ekleme, silme, güncelleme, listeleme ve çalışanlara görev atama gibi işlemleri gerçekleştirebilir. Tüm işlemler, görevlerin düzenli bir şekilde takip edilmesini ve yönetilmesini sağlar.

Görev Listesi ve Çalışan Eşleştirme

GetData fonksiyonu, mevcut kullanıcının görevlerini veritabanından çeker ve bir tabloya doldurur. Bu veriler, kullanıcı arayüzündeki DataGridView bileşeninde listelenir. fillComboBox fonksiyonu ise çalışanları bir listeye ekleyerek görev atanacak çalışanların seçiminde kolaylık sağlar.

Görev Ekleme

addTask_Click fonksiyonu, kullanıcıdan alınan bilgilerle yeni bir görev ekler. Görev adı, başlangıç ve bitiş tarihleri, görev durumu ve atanan çalışan gibi bilgilerin eksiksiz girilmesi sağlanır. Girilen veriler, veritabanına kaydedilir ve görev listesi yenilenir.

Görev Silme ve Güncelleme

deleteTask_Click fonksiyonu, seçilen bir görevi silerken görevlerin bağlı olduğu projeler dikkate alınır. updateTask_Click fonksiyonu ise bir görevin bitiş tarihi, iş gün sayısı ve durumu gibi bilgileri günceller. Güncelleme sonrası liste yenilenir ve kullanıcıya bilgilendirme mesajı gösterilir.

Görev Arama ve Seçim İşlemleri

searcByTaskName_TextChanged fonksiyonu, görev adlarına göre filtreleme yaparak büyük listelerde arama kolaylığı sağlar. taskList_CellEnter fonksiyonu, seçilen görev satırındaki bilgileri düzenleme kutularına doldurarak kullanıcıya kolaylık sunar.

PROJELER SAYFASI KODLARI ve ARAYÜZÜ

```
namespace ProjePilot.Forms
{
    public partial class ProjectPage : Form
    {
        private int selectedTaskId;
        private int userId;

        private const string ConnectionString = "Server=localhost;Database=vtys;Uid=beyuphan;Pwd='yphnbnc53';Convert Zero Datetime = True;";

        private MySqlConnection connection;
        private MySqlCommand command;
        private DataTable dataTable;
        private MySqlDataReader dataReader;
        private DataView dataView;

        private string getDataQuery = "SELECT P.ProjectId, P.ProjectName, P.ProjectBeginDate, P.ProjectEndDate, U.UserName, T.TaskName, E.EmployeeFirstName " +
            "FROM vtys.Project P " +
            "LEFT JOIN vtys.Users U ON P.UserId = U.UserId " +
            "LEFT JOIN vtys.Task T ON P.TaskId = T.TaskId " +
            "LEFT JOIN vtys.Employee E ON P.UserId = E.UserId AND T.EmployeeId = E.EmployeeId  " +
            "WHERE P.UserId = @userId";
        private string fillComboBoxQuery = "SELECT * FROM Task WHERE UserId = @userId";
        private string addNewProjectQuery = "INSERT INTO Project (ProjectName, ProjectBeginDate, ProjectEndDate, TaskId, UserId) " +
            "VALUES (@ProjectName, @ProjectBeginDate, @ProjectEndDate, @TaskId, @UserId)";
        private string isValidTaskIdQuery = "SELECT COUNT(*) FROM task WHERE TaskId = @TaskId";
        private string getValidTaskIdFromDatabaseQuery = "SELECT TaskId FROM task LIMIT 1";
        private string deleteQuery = "DELETE FROM project WHERE ProjectId = @ProjectId";
        private string updateProjectQuery = "UPDATE Project " +
                                            "SET ProjectEndDate = @ProjectEndDate, TaskId = @TaskId " +
                                            "WHERE ProjectId = @ProjectId";

        public ProjectPage(int userId)
        {
            InitializeComponent();
            this.userId = userId;
            connection = new MySqlConnection(ConnectionString);
            FillComboBox();
        }

        private void ProjectPage_Load(object sender, EventArgs e)
        {
            GetData();
        }

        private void turnBack_Click(object sender, EventArgs e)
        {
            MainPage mainPage = new MainPage(userId);
            this.Hide();
            mainPage.ShowDialog();
        }

        private void addProject_Click(object sender, EventArgs e)
        {
            AddNewProject();
        }

        private void GetData()
        {
            try
            {
                dataTable = new DataTable();
                using (MySqlConnection connection = new MySqlConnection(ConnectionString))
                {
                    connection.Open();


                    command = new MySqlCommand(getDataQuery, connection);
                    command.Parameters.AddWithValue("@userId", userId);

                    MySqlDataAdapter adapter = new MySqlDataAdapter(command);
                    adapter.Fill(dataTable);
                }

                projectList.DataSource = dataTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error fetching data: " + ex.Message);
            }
            finally
            {
                connection.Close();
            }
        }
```

```
private void FillComboBox()
{
    try
    {
        command = new MySqlCommand(fillComboBoxQuery, connection);

        command.Parameters.AddWithValue("@userId", userId);

        connection.Open();
        dataReader = command.ExecuteReader();
        while (dataReader.Read())
        {
            int taskId = dataReader.GetInt32(0);
            string taskName = dataReader.GetString(1);
            addTask.Items.Add(new KeyValuePair<int, string>(taskId, taskName));
            changeTask.Items.Add(new KeyValuePair<int, string>(taskId, taskName));

        }
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error filling combo box: " + ex.Message);
    }
}

private void AddNewProject()
{
    try
    {
        string projectName = addProjectName.Text;
        DateTime projectBeginDate = addProjectBeginDate.Value;
        DateTime projectEndDate = addProjectEndDate.Value;
        int taskId = selectedTaskId;

        if (string.IsNullOrEmpty(projectName))
        {
            MessageBox.Show("Project name cannot be empty.");
            return;
        }
        if (addTask.SelectedItem != null)
        {
            KeyValuePair<int, string> selectedTask = (KeyValuePair<int, string>)addTask.SelectedItem;
            selectedTaskId = selectedTask.Key;
        }
        if (addTask.SelectedItem == null)
        {
            MessageBox.Show("Geçerli bir Görev seçilmelidir.");
            return;
        }

        using (MySqlConnection connection = new MySqlConnection(ConnectionString))
        {
            connection.Open();

            if (!IsValidTaskId(taskId, connection))
            {
                int validTasktId = GetValidTaskIdFromDatabase(connection);
                if (validTasktId > 0)
                {
                    taskId = validTasktId;
                }
                else
                {
                    MessageBox.Show("No valid Project found.");
                    return;
                }
            }



            command = new MySqlCommand(addNewProjectQuery, connection);
            command.Parameters.AddWithValue("@ProjectName", projectName);
            command.Parameters.AddWithValue("@ProjectBeginDate", projectBeginDate);
            command.Parameters.AddWithValue("@ProjectEndDate", projectEndDate);
            command.Parameters.AddWithValue("@TaskId", taskId);
            command.Parameters.AddWithValue("@UserId", userId);

            int rowsAffected = command.ExecuteNonQuery();

            if (rowsAffected > 0)
            {
                MessageBox.Show("Project added successfully.");
                addProjectName.Text = "";
                GetData();
            }
```

```
            else
            {
                MessageBox.Show("Error adding project.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("An error occurred: " + ex.Message);
    }
}

private bool IsValidTaskId(int taskId, MySqlConnection connection)
{
    try
    {
        using (command = new MySqlCommand(isValidTaskIdQuery, connection))
        {
            command.Parameters.AddWithValue("@TaskId", taskId);
            int count = Convert.ToInt32(command.ExecuteScalar());

            return count > 0;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error checking valid Task: " + ex.Message);
        return false;
    }
}

private int GetValidTaskIdFromDatabase(MySqlConnection connection)
{
    try
    {
        using (command = new MySqlCommand(getValidTaskIdFromDatabaseQuery, connection))
        {
            object result = command.ExecuteScalar();

            if (result != null && result != DBNull.Value)
            {
                return Convert.ToInt32(result);
            }
            else
            {
                MessageBox.Show("No valid Task found.");
                return -1;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error getting valid TaskId: " + ex.Message);
        return -1;
    }
}

private void addTask_SelectedIndexChanged(object sender, EventArgs e)
{
    if (addTask.SelectedItem != null)
    {
        if (addTask.SelectedItem is KeyValuePair<int, string> selectedTask)
        {
            selectedTaskId = selectedTask.Key;
            taskId.Text = selectedTaskId.ToString();
        }
    }
}
private void changeTask_SelectedIndexChanged(object sender, EventArgs e)
{
    if (changeTask.SelectedItem != null)
    {
        if (changeTask.SelectedItem is KeyValuePair<int, string> selectedTask)
        {
            taskId.Text = selectedTask.Key.ToString();
        }
    }
}
```

```
private void deleteProject_Click(object sender, EventArgs e)
{
    try
    {
        if (projectList.SelectedRows.Count > 0)
        {

            int selectedRowIndex = projectList.SelectedRows[0].Index;

            string columnName = "ProjectId";

            int projectId = Convert.ToInt32(projectList.Rows[selectedRowIndex].Cells[columnName].Value);

            using (connection)
            {
                connection.Open();

                command = new MySqlCommand(deleteQuery, connection);
                command.Parameters.AddWithValue("@ProjectId", projectId);

                command.ExecuteNonQuery();
            }

            GetData();
            MessageBox.Show("Kayıt Silindi");
        }
        else
        {
            MessageBox.Show("Lütfen bir satır seçin.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("basarisiz oldu.\b" + ex);

    }
}

private void updateProject_Click(object sender, EventArgs e)
{
    int taskId = selectedTaskId;



    if (changeTask.SelectedItem != null)
    {
        KeyValuePair<int, string> selectedTask = (KeyValuePair<int, string>)changeTask.SelectedItem;
        selectedTaskId = selectedTask.Key;
    }
    using (MySqlConnection connection = new MySqlConnection(ConnectionString))
    {
        try
        {
            connection.Open();


            if (!IsValidTaskId(taskId, connection))
            {

                int validTaskId = GetValidTaskIdFromDatabase(connection);

                if (validTaskId > 0)
                {
                    taskId = validTaskId;
                }
                else
                {
                    MessageBox.Show("No valid Project found.");
                    return;
                }
            }

            int selectedRowIndex = projectList.SelectedRows[0].Index;
            string columnName = "ProjectId";
            int projectId = Convert.ToInt32(projectList.Rows[selectedRowIndex].Cells[columnName].Value);



            using (command = new MySqlCommand(updateProjectQuery, connection))
            {
                command.Parameters.AddWithValue("@ProjectEndDate", updateProjectEndDate.Value.ToString("yyyy-MM-dd"));
                command.Parameters.AddWithValue("@ProjectId", projectId);
                command.Parameters.AddWithValue("@TaskId", taskId);

                command.ExecuteNonQuery();
            }
```

ProjectPage Sınıfı
Bu sınıf, kullanıcıların projeleri yönetmesine olanak tanır. Projelerin listelenmesi, eklenmesi, güncellenmesi ve silinmesi gibi işlevler içerir. Ayrıca, projelere görevler atayarak daha düzenli bir proje yönetimi sağlar.

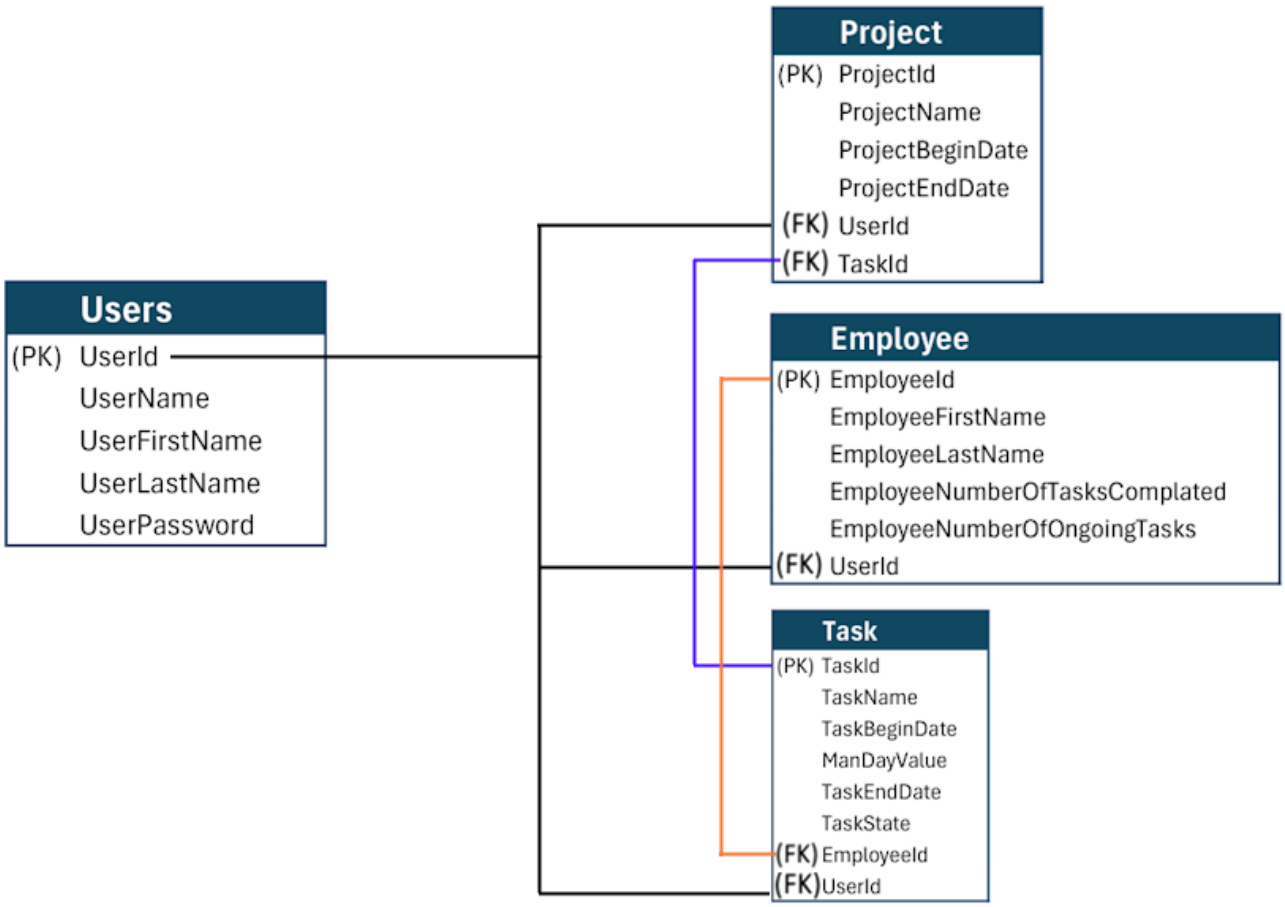
Projelerin Listelenmesi ve Filtrelenmesi
GetData fonksiyonu, kullanıcıya ait projeleri veritabanından çeker ve bunları bir tabloya doldurur. Bu bilgiler projectList bileşeninde görselleştirilir. Kullanıcı, searchByProjectName_TextChanged fonksiyonu ile proje adlarına göre listeyi filtreleyebilir. Bu sayede, büyük veri setlerinde arama kolaylığı sağlanır.

Proje Ekleme
AddNewProject fonksiyonu, kullanıcının yeni bir proje eklemesine imkan tanır. Kullanıcı proje adı, başlangıç ve bitiş tarihleri gibi bilgileri doldurarak bir görev seçer. Görev doğruluğu, IsValidTaskId fonksiyonu ile kontrol edilir. Veriler eksiksiz olduğunda, proje veritabanına kaydedilir ve liste yenilenir.

Proje Güncelleme ve Silme
Projelerde değişiklik yapmak için updateProject_Click fonksiyonu kullanılır. Kullanıcı seçili bir projede görev değişikliği yapabilir veya proje bitiş tarihini güncelleyebilir. Güncellemeler, seçili satırın bilgileriyle gerçekleştirilir. deleteProject_Click fonksiyonu, kullanıcıların projeleri silmesini sağlar. Ancak bir proje başka bir veriye bağlıysa, kullanıcı bilgilendirilir ve silme işlemi iptal edilir.

Görev ve Proje Bağlantısı
FillComboBox fonksiyonu, projelere atanabilecek görevlerin bir listesini oluşturur. Kullanıcı, bu görevlerden birini seçerek projeye bağlayabilir. Bu, proje ve görev arasında güçlü bir bağlantı kurar ve yönetim sürecini kolaylaştırır.

UML ve ER Diyagramları



1. Users Tablosu

Sistemdeki kullanıcıların bilgilerini saklar. Bu tabloda, her bir kullanıcıyı benzersiz bir şekilde tanımlayan UserId birincil anahtar (PK) olarak kullanılmıştır.

Alanlar:

- UserId: Her kullanıcıya ait benzersiz kimlik.
- UserName: Kullanıcının sisteme giriş için kullandığı kullanıcı adı.
- UserFirstName: Kullanıcının adı.
- UserLastName: Kullanıcının soyadı.

- UserPassword: Kullanıcının şifresi.

İlişkiler:

- Diğer tablolarla ilişkilendirilerek, her kullanıcının sahip olduğu projeler, görevler ve çalışanlar yönetilmektedir.

---

2. Project Tablosu

Projelerin temel bilgilerini tutar ve her proje, kullanıcılar ve görevlerle ilişkilendirilir.

Alanlar:

- ProjectId: Projeyi benzersiz şekilde tanımlayan birincil anahtar.
- ProjectName: Projenin adı.
- ProjectBeginDate: Projenin başlangıç tarihi.
- ProjectEndDate: Projenin bitiş tarihi.
- UserId (FK): Projeyi oluşturan kullanıcıya işaret eden yabancı anahtar.
- TaskId (FK): Projeye atanmış görevi işaret eden yabancı anahtar.

İlişkiler:

- UserId ile Users tablosuna bağlanarak projeyi oluşturan kullanıcı belirlenir.
- TaskId ile Task tablosuna bağlanarak projeye atanmış görevler belirlenir.

---

3. Employee Tablosu

Çalışanların bilgilerini saklar. Her çalışan, görevler ve kullanıcılarla ilişkilendirilmiştir.

Alanlar:

- EmployeeId: Çalışanı benzersiz şekilde tanımlayan birincil anahtar.
- EmployeeFirstName: Çalışanın adı.
- EmployeeLastName: Çalışanın soyadı.
- EmployeeNumberOfTasksCompleted: Çalışanın tamamladığı görev sayısı.
- EmployeeNumberOfOngoingTasks: Çalışanın devam eden görev sayısı.
- UserId (FK): Çalışanın bağlı olduğu kullanıcıyı işaret eden yabancı anahtar.

İlişkiler:

- UserId ile Users tablosuna bağlanarak çalışanların hangi kullanıcıya ait olduğu belirlenir.

---

4. Task Tablosu

Görevlerin bilgilerini saklar ve görevlerin projeler, çalışanlar ve kullanıcılarla ilişkisini sağlar.

Alanlar:

- TaskId: Görevi benzersiz şekilde tanımlayan birincil anahtar.
- TaskName: Görevin adı.
- TaskBeginDate: Görevin başlangıç tarihi.
- ManDayValue: Görev için ayrılan adam/gün sayısı.
- TaskEndDate: Görevin bitiş tarihi.
- TaskState: Görevin durumu (örn. devam ediyor, tamamlandı).
- EmployeeId (FK): Görevi gerçekleştiren çalışanı işaret eden yabancı anahtar.
- UserId (FK): Görevin bağlı olduğu kullanıcıyı işaret eden yabancı anahtar.

İlişkiler:

- EmployeeId ile Employee tablosuna bağlanarak görevi gerçekleştiren çalışan belirlenir.
- UserId ile Users tablosuna bağlanarak görevin hangi kullanıcıya ait olduğu belirlenir.

İlişki Özeti

1. **Users ile Project:** Her kullanıcı birden fazla proje oluşturabilir (1:N).
2. **Users ile Employee:** Her kullanıcı birden fazla çalışan ekleyebilir (1:N).
3. **Users ile Task:** Her kullanıcı birden fazla görev ekleyebilir (1:N).
4. **Project ile Task:** Her proje bir göreve bağlıdır, ancak her görev birden fazla projeye bağlanabilir (N:N).
5. **Task ile Employee:** Her görev bir çalışana atanır (1:N).

Veritabanı Tasarımı ve SQL Sorguları

1. Employee Tablosu (Çalışanlar)

```sql
-- tablo yapısı dökülüyor vtys.employee
CREATE TABLE IF NOT EXISTS `employee` (
  `EmployeeId` int(11) NOT NULL AUTO_INCREMENT,
  `EmployeeFirstName` text NOT NULL,
  `EmployeeLastName` text NOT NULL,
  `EmployeeNumberOfTasksCompleted` int(11) NOT NULL DEFAULT 0,
  `EmployeeNumberOngoingTasks` int(11) NOT NULL DEFAULT 0,
  `UserId` int(11) NOT NULL,
  PRIMARY KEY (`EmployeeId`) USING BTREE,
  KEY `FK_employee_users` (`UserId`),
  CONSTRAINT `FK_employee_users` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_turkish_ci;
```

Bu tablo, sisteme dahil olan çalışanların bilgilerini tutar ve her çalışanın bağlı olduğu kullanıcı ile ilişkilendirilmesini sağlar. Ayrıca, çalışanların tamamladığı ve devam eden görev sayıları takip edilir.

- **EmployeeId**: Çalışanları benzersiz şekilde tanımlayan birincil anahtar.
- **EmployeeFirstName ve EmployeeLastName**: Çalışanın adı ve soyadı.
- **EmployeeNumberOfTasksCompleted**: Çalışanın tamamladığı görevlerin sayısı. Varsayılan değer 0 olarak atanmıştır.
- **EmployeeNumberOngoingTasks**: Çalışanın devam eden görevlerinin sayısı. Varsayılan değer 0 olarak atanmıştır.
- **UserId**: Çalışanın hangi kullanıcıya bağlı olduğunu belirten yabancı anahtar.

İlişkiler:
Bu tablo, users tablosundaki UserId sütunuyla ilişkilendirilmiştir. Bu ilişki, bir çalışanın hangi kullanıcıya ait olduğunu belirler. Silme ve güncelleme işlemlerinde ilişkisel veri tutarlılığı korunur.

2. Project Tablosu (Projeler)

```sql
-- tablo yapısı dökülüyor vtys.project
CREATE TABLE IF NOT EXISTS `project` (
  `ProjectId` int(11) NOT NULL AUTO_INCREMENT,
  `ProjectName` text NOT NULL,
  `ProjectBeginDate` date NOT NULL,
  `ProjectEndDate` date NOT NULL,
  `UserId` int(11) NOT NULL,
  `TaskId` int(11) NOT NULL,
  PRIMARY KEY (`ProjectId`) USING BTREE,
  KEY `FK_project_user` (`UserId`) USING BTREE,
  KEY `FK_project_task` (`TaskId`),
  CONSTRAINT `FK_project_task` FOREIGN KEY (`TaskId`) REFERENCES `task` (`TaskId`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_project_user` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=46 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_turkish_ci;
```

Projelerin temel bilgilerini saklar ve bu projelerin görevler ve kullanıcılarla olan ilişkilerini düzenler.

Tablo Yapısı:

- **ProjectId**: Projeleri benzersiz şekilde tanımlayan birincil anahtar.
- **ProjectName**: Projenin adı.
- **ProjectBeginDate ve ProjectEndDate**: Projenin başlangıç ve bitiş tarihleri.

- **UserId**: Projeyi oluşturan kullanıcıyı belirten yabancı anahtar.
- **TaskId**: Projeye bağlı görevi işaret eden yabancı anahtar.

İlişkiler:
Bu tablo, users ve task tablolarıyla ilişkilendirilmiştir. Her proje, bir kullanıcıya ve bir göreve bağlanır. Bu yapı, proje-görev-kullanıcı ilişkisini düzenler ve projelerin daha iyi yönetilmesini sağlar.

3. Task Tablosu (Görevler)

```
-- tablo yapısı dökülüyor vtys.task
CREATE TABLE IF NOT EXISTS `task` (
  `TaskId` int(11) NOT NULL AUTO_INCREMENT,
  `TaskName` text NOT NULL,
  `TaskBeginDate` date NOT NULL,
  `ManDayValue` int(11) NOT NULL,
  `TaskEndDate` date NOT NULL,
  `TaskState` text NOT NULL,
  `EmployeeId` int(11) NOT NULL,
  `UserId` int(11) NOT NULL,
  PRIMARY KEY (`TaskId`) USING BTREE,
  KEY `FK_task_employee` (`EmployeeId`) USING BTREE,
  KEY `FK_task_project` (`UserId`) USING BTREE,
  CONSTRAINT `FK_task_employee` FOREIGN KEY (`EmployeeId`) REFERENCES `employee` (`EmployeeId`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_task_users` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=38 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_turkish_ci;
```

Görevlerin detaylı bilgilerini saklar ve bu görevlerin projeler, çalışanlar ve kullanıcılarla olan bağlantılarını düzenler.

Tablo Yapısı:

- **TaskId**: Görevleri benzersiz şekilde tanımlayan birincil anahtar.
- **TaskName**: Görevin adı.
- **TaskBeginDate ve TaskEndDate**: Görevin başlangıç ve bitiş tarihleri.
- **ManDayValue**: Görev için gereken adam/gün değeri.
- **TaskState**: Görevin durumu (ör. "Devam Ediyor", "Tamamlandı").
- **EmployeeId**: Görevi gerçekleştiren çalışanı belirten yabancı anahtar.
- **UserId**: Görevin bağlı olduğu kullanıcıyı belirten yabancı anahtar.

İlişkiler:
Bu tablo, employee ve users tablolarıyla ilişkilendirilmiştir. Görevlerin hangi çalışana ve hangi kullanıcıya ait olduğu net bir şekilde tanımlanır. Ayrıca, project tablosuyla da bağlantı kurarak görevlerin projelerle ilişkilendirilmesini sağlar.

4. Users Tablosu (Kullanıcılar)

```
-- tablo yapısı dökülüyor vtys.users
CREATE TABLE IF NOT EXISTS `users` (
  `UserId` int(10) NOT NULL AUTO_INCREMENT,
  `UserFirstName` text CHARACTER SET utf8 COLLATE utf8_turkish_ci NOT NULL,
  `UserLastName` text CHARACTER SET utf8 COLLATE utf8_turkish_ci NOT NULL,
  `UserName` text CHARACTER SET utf8 COLLATE utf8_turkish_ci NOT NULL,
  `UserPassword` text CHARACTER SET utf8 COLLATE utf8_turkish_ci NOT NULL,
  PRIMARY KEY (`UserId`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_turkish_ci;
```

Sistemdeki kullanıcıların temel bilgilerini saklar. Kullanıcılar, projeler, görevler ve çalışanlarla ilişkilidir.

Tablo Yapısı:

- **UserId**: Kullanıcıları benzersiz şekilde tanımlayan birincil anahtar.
- **UserFirstName ve UserLastName**: Kullanıcının adı ve soyadı.
- **UserName**: Kullanıcının giriş için kullandığı kullanıcı adı.
- **UserPassword**: Kullanıcının şifresi.

İlişkiler:
Bu tablo, diğer tüm tablolarla ilişkilendirilmiştir. UserId, her kullanıcının projeler, görevler ve çalışanlarla olan bağlantısını sağlar. Böylece her kullanıcının kendi bilgilerine ve ilişkili verilere erişimi sağlanır.

Linkler

Proje kodları GitHub Repositry:
[beyuphan/Projecte: VTYS Final ödevi masaüstü uygulaaması]

Proje videosu Youtube Linki:

[Veri Tabanı Yönetim Sistemleri Final Ödevi]