Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı II

MÜZİK DOSYAM

Taha Uz/Burak Emir Ayrancı

maskeden_2000@hotmail.com[Taha]/190202014@kocaeli.edu.tr[Emir]

Projenin Özeti:

Programlama laboratuvar II Projesi olarak bizden "Müzik Dosyam" adlı uygulama geliştirmemiz beklenmektedir.

Biz proje için Java programlama dilini ve SQL sorgu dilini kullanmayı , Netbeans geliştirme ortamında ve MySQL veri tabanı sistemi kullanmayı tercih ettik.

Proje dökümanında bizden , kullanıcıların (Premium ve normal kullanıcı olmak üzere) şarkı dinleyeceği, Premium kullanıcıların çalma listelerinin takip edilebileceği ve tüm kullanıcıların kendi çalma listelerini oluşturabileceği veri tabanı sistemi ve bu sistemi destekleyen arayüz tasarımı ile bir müzik uygulaması tasarlamamız istenmektedir .

Biz projede Java programlama dilinde bulunan "Swing" adlı arayüz tasarım kütüphanesinden , veri tabanı tasarımında ise MySQL veri tabanı yönetim sisteminden yararlandık .

İlk olarak kullanıcı uygulamamıza "Giriş" ekranında "Kayıt Ol" butonuna tıklayarak "Kayıt Form" ekranına giriş yapıp burada "Kullanıcı ID , Kullanıcı AD , Email , Sifre , AbonelikTuru , Ulke " bilgileri alarak "musicapidb" veri tabanında bulunan "kullanici" adlı tabloya bu bilgileri aktarıyoruz . Daha sonra kullanıcı kayıt ekranına geri dönerek hesabına giriş gerçekleştiriyor . Uygulamada "Muziklerim,Top10,Tum Muzikler,Tum Kullanıcılar,Album ve Sanatci , Admin Mod " gibi butonlar karşılıyor . Bu butonların gözükmesi kullanıcı tipine göre değişim göstermektedir. Daha sonra kullanıcı "Top10,Tum Muzikler" gibi butonlara basarak muzik ekleyebilir ve veri tabanındaki tüm müzikleri , en çok dinlenenleri kendi listesine ekleyebilir . İsterse "Album ve Sanatci " butonuna tıklayarak müziklerin album ve sanatci tablolarının detaylarını inceleyebilir. İsterse de "Tum Kullanıcılar" butonuna basarak başka kullanıcıların bilgilerini görebilir ve kullanıcıların bilgilerinin yazdığı bloklara tıklayarak başka kullanıcıların müzik listelerini görüntüleyebilir ve

kendi listesine müzik ekleyebilir . Eğer giriş yapan kullanıcı “Admin” yetkisine sahip ise “Admin Mod” adlı buton ile veri tabanındaki “Sarki , Album , Sanatci , Dinlenmesayisi ” tablolarına ekle , sil , güncelle işlemlerini bellirli kurallara uygun olucak şekilde ,program kurallara uygunluğunu kontrol eder , gerçekleştirebilir .

1.GİRİŞ

Proje için Java programlama dilini ve SQL sorgu dilini kullanmayı , Netbeans geliştirme ortamında ve MySQL veri tabanı sistemi kullanmayı tercih ettik.

Java programlama dili ; Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu , nesneye yönelik , zeminden bağımsız , yüksek verimli , çok işlevli , yüksek seviye , adım adım işletilen (yorumlanan-interpreted) bir dildir.

SQL , verileri yönetmek ve tasarlaman için kullanılan bir dildir. SQL , kendisi bir programlama dili olmamasına rağmen birçok kiişi tarafından programlama dili olarak bilinir . SQL herhangi bir veri tabanı ortamında kullanılan bir alt dildir .

Netbeans platformu; Oracle tarafından geliştirilen bir java geliştirme ortamıdır (IDE) ve ücretsiz olarak dağıtılmaktadır. Özellikle kullanıcı arayüzü tasarımında sağladığı kolaylıklardan dolayı tercih edilmektedir .

MySQL , altı milyondan fazla sistemde yüklü bulunan çoklu iş parçaçıklı , çok kullanıcılı , hızlı ve sağlam bir veri tabanı yönetim sistemidir. UNIX , OS/2 ve Windows platformları için ücretsiz dağıtılmakla birlikte ticari lisans kullanmak isteyenler için de ücretli bir listans seçeneği de mevcuttur.

2.TEMEL BİLGİLER

Projemizde ; “AdminArayuz , AnaProgram , MuzikDosyam , KullaniciPop , LoginScreen , RegisterScreen” olmak üzere toplam da altı adet “.java” uzantılı dosyalardan oluşmaktadır . Anlatım olarak tek tek “.java” uzantıların adı ile alt başlıklara ayırarak anlatıcağız .

2.1. MuzikDosyam

Bu Class yapısının içerisinde “LoginScreen” nesnesi oluşturulur ve “LoginScreen” Class yapısının içerisinde yer alan “al(logisc)” fonksiyonunun içerisine oluşturduğumuz Class’ı yollayarak ve bu yapının “logisc.setVisible(true)” ile görünürlüğünü sağlayarak işlemi bitiriyoruz .

2.2. LoginScreen

Burada Kullanıcı kayıt olduğu bilgilerdeki “ Kullanici ID , Sifre ” bilgilerini Username ve Password istenen kutucuklara girerek sisteme giriş yapıyor. Kullanicinin bilgilerini SQL sorgu dili ile “kullanici” tablosundaki bilgilerini buluyor ve kutucuklardaki bilgilerle doğruluğunu test ediyor ve giriş izni veriyor .

```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();
    ResultSet myRs = myStat.executeQuery("select * from kullanici");
    while (myRs.next()) {

        if (myRs.getString("kullaniciid").equals(id) && myRs.getString("sifre").equals(sifre)) {
            JOptionPane.showMessageDialog(null, "Giris Yapildi !");
            ad = myRs.getString("kullaniciad");
            onay = 1;
        }
    }
    if(onay == 0){
        JOptionPane.showMessageDialog(null,"kullanici ad/sifre kutularini kontrol et !");
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null,"Data Base Hatasi : "+e.getMessage());
}
```

Kullanıcı başarılı bir şekilde giriş yaptıktan sonra kullanıcıya göre AnaProgram nesnesi oluşturuluyor .

Kullanıcı isterse bu arayüzdeki “Kayıt Ol” butonunu kullanarak “RegisterScreen” nesnesi oluşturuluyor ve kullanıcıyı kayıt olma ekranına geçiş (regs.setVisible(true)) yapılıyor ve giriş ekranı (logisc.setVisible(false)) kapatılıyor .

2.3. RegisterScreen

Bu Class Kullanıcı kayıt sistemini içermektedir . Kullanıcı burada “Kullanici AD , Kullanici ID, Email , Sifre , Ulke , Abonelik Turu” bilgilerini yanlarındaki kutucukların içine girerek kayıt olma işlemini gerçekleştiriyor .

Biz burada kullanıcıdan aldığımız bilgileri SQL sorgu dilini kullanarak “musicapidb” veri tabanının içerisindeki “kullanici” tablosuna verileri yerleştiriyoruz .

```
try {

    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");

    String sorgu = "INSERT INTO kullanici values('"+id+"','" + ad + "','" + mail + "','" + sifre + "','" + abonelik + "','"+country+"')";
    Statement sta = myConn.createStatement();
    int x = sta.executeUpdate(sorgu);
    if(x == 0){
        JOptionPane.showMessageDialog(null,"Kullanici zaten var !");
    }else{
        JOptionPane.showMessageDialog(null,"Hesabiniz basariyla olusturuldu !");

        onay = 1;
        kullaniciid++;

    }
} catch (Exception e) {

    if(e.getMessage().indexOf("Duplicate") != -1){
        JOptionPane.showMessageDialog(null,"Bu kullanici ad ve sifresinde bir kullanici zaten var !");
    }else{
        JOptionPane.showMessageDialog(null,"Data Base Hatasi : "+e.getMessage());
    }

}
```

Kullanıcı kayıt bilgilerini girip “Tamam” butonuna bastığında biz , bu butonun içerisinde kullanıcının girdiği ismiyle kullanıcıya özel veri tabanında müziklerini ekleyebileceği bir tablo oluşturuyoruz .

```
if(onay == 1){
    try{

        Connection myConn2 = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
        String sorgu2 = "CREATE TABLE "+ad+"( sarkiad VARCHAR(65) NOT NULL,tarih VARCHAR(45) NOT NULL,"+
                "album VARCHAR(65) NOT NULL,tur VARCHAR(45) NOT NULL,sure FLOAT NOT NULL,PRIMARY KEY (sarkiad, tarih))";
        Statement sta2 = myConn2.createStatement();
        sta2.executeUpdate(sorgu2);
    }catch(Exception e){
        JOptionPane.showMessageDialog(null,"Data Base Hatasi : "+e.getMessage());
    }
    regs.setVisible(false);
    logisc.setVisible(true);
    onay=0;
}
```

Daha sonra “al” fonksiyonu sayesinde aldığımın “LoginScreen” nesnesini tekrardan “setVisible()” fonksiyonu sayesinde açıyor , kayıt formunu ise kapatıyoruz .

2.4.AnaProgram

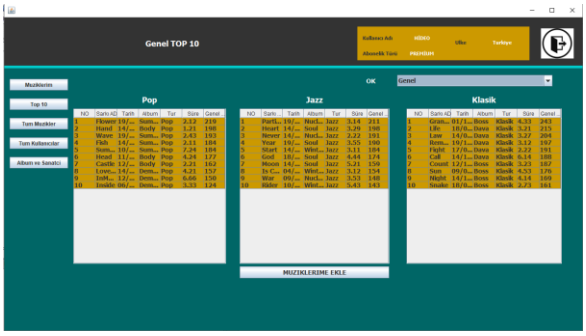
Bu nesne ilk kez çağırılırken kullanıcının ; ad , şifre ve id bilgilerini parametre olarak kendi içindeki değişkenlere atamak için alır . Daha sonra kullanıcının ismini , abonelik türünü ve ülkesini “arayuz_musteri_info()” metotu sayesinde arayüzde kullanıcıya yansıtıyoruz . Bu fonksiyonda önceden aldığımız kullanıcıID ve şifre bilgilerinden SQL sorgu ile “SELECT * from kullanici ” diyerek veri tabanında arayarak kullanıcının abonelik türünü ve ülkesini buluyor ve arayüze aktarıyoruz . Bununla birlikte kullanıcının abonelik türüne göre “Admin Mod” ‘u görünürlüğünü belirliyoruz . Daha sonra Muziklerim , Top10 , Tum Muzikler , Tum Kullanicilar , Album ve Sanatci , Admin Mod butonları karşınıza çıkıyor . Bazı butonlar sizin kullanıcı türlerinize göre değişim göstermektedir.

```
public void arayuz_musteri_info() {

    String abonelik = null;
    String ulke = null;
    try {
        Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
        Statement myStat = (Statement) myConn.createStatement();
        ResultSet myRs = myStat.executeQuery("select * from kullanici");
        while (myRs.next()) {

            if (myRs.getString("kullaniciid").equals(kullaniciID) && myRs.getString("sifre").equals(kullaniciSifre)) {
                abonelik = myRs.getString("abonelikturu");
                ulke = myRs.getString("ulke");
            }
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
    }
    Musteri_isim.setText(kullaniciAD.toUpperCase());
    Kullanici_turu.setText(abonelik.toUpperCase());
    musteriulke.setText(ulke);

    tumKullaniciOnOff(false);
    muziklerimOnOff(false);

    if (abonelik.toLowerCase().equals("admin")) {
        jButton4.setVisible(true);
    } else {
        jButton4.setVisible(false);
    }
}
```

İlk olarak Muziklerim butonundan bahsedelim . Bu buton kullanıcının adı ile oluşturulmuş (Örneğin taha,emir) tablolarınızdan müzik bilgilerinizi “jTable” ların satırlarına müzik türlerine dikkat ederek aktarır ve kullanıcıya sunulur . Böylelikle kullanıcı kendi şarkılarını görebilir ve daha sonra sıkıldığında kendi listesinden (Muzik Sil butonu ile) kaldırabilir . Bu işlemlerin hepsini “muziklerimTablo()” metotu kullanarak arayüz ile aktarımı sağlanmaktadır .

```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();
    ResultSet myRs = myStat.executeQuery("select * from " + kullaniciAD + " where tur = '" + MusicType.get(1) + "'");
    while (myRs.next()) {
        String sarkiad = myRs.getString("sarkiad");
        String tarih = myRs.getString("tarih");
        String album = myRs.getString("album");
        String tur = myRs.getString("tur");
        float sure = myRs.getFloat("sure");

        String tbData[] = {sarkiad, tarih, album, tur, sure + ""};

        tblModel.addRow(tbData);

    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
}
```

Daha sonra “Top10” butonuna basarak kullanıcı müziklerin dinlenme sayılarına göre sıralanan tabloları görmektedir . Kullanıcı bu kısımda Genel , Türkiye , Almanya , ABD olmak üzere bakabileceği “TOP 10” listesine ulaşabilmektedir. Kullanıcı bu butona ilk bastığında standart olarak “Genel” sıralama bilgiler gelmektedir .



Kodun bu kısımları nasıl yapıldığına bakıcak olursak butona bastığında “top10standart()” metodu sayesinde arayüze aktarım sağlanır . Arayüze aktarılırken SQL sorgu dilinde Sarki ve Dinlenmesayisi tablolarını kartezyen çarpımları yapılır ve Dinlenmesayisi tablosundaki şarkı adı ile Sarki tablosundaki ad verileri aynı ise bize o satırları döndür koşulunu yazıyoruz . Bunları yaparken de şarkının türlerine göre seçip bu sayede her tür ayrı tabloda sıralanmak üzere ayrım yapıp işlemi tamamlıyoruz . Bu işlemlerde SQL sorgu dilini "select * from sarki,dinlenmesayisi where sarki.ad = dinlenmesayisi.sarkiad and sarki.tur = '" + MusicType.get(1) + "' order by abd+turkiye+almanya DESC" olucak şekilde sorgumuzu yapıyoruz . Böylelikle şarkının her ülkede dinlenme sayisini veri tabanımızdan almıp arayüze aktarmış oluyoruz . Bu şekilde kullanıcı kendi seçtiği ülkeyi aynı bu biçimde istediği gibi sıralayabiliyor .

```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();

    ResultSet myRs = myStat.executeQuery("select * from sarki,dinlenmesayisi where sarki.ad = dinlenmesayisi.sarkiad and sarki.tur = '" + MusicType.get(1)
    while (myRs.next() && no<=10) {

        String ad = myRs.getString("ad");
        String tarih = myRs.getString("tarih");
        String sanatci = myRs.getString("album");
        String tur = myRs.getString("tur");
        String sure = myRs.getString("sure");
        int geneldinlenmesayi = myRs.getInt("turkiye") + myRs.getInt("abd") + myRs.getInt("almanya");
        String tbData[] = {no + "", ad, tarih, sanatci, tur, sure, geneldinlenmesayi + ""};
        tblModel.addRow(tbData);
        no++;
    }
```

Daha sonra “Tum Muzikler” butonuna basarak kullanıcı veri tabanındaki tüm müzikleri türlerine ayrılmış bir biçimde tablolarda sıralı bir biçimde görülmesi sağlanıyor . Bunu yaparken SQL sorgu dilinde sorgulanarak "select * from sarki where tur = '" + MusicType.get(1) + "' " tek tek verileri elde edilip tablolara koyuluyor . Böylelikle kullanıcı tüm veri tabanına bakarak beğendiği müzikleri listesine eklenmesi sağlanılıyor .

```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();
    ResultSet myRs = myStat.executeQuery("select * from sarki where tur = '" + MusicType.get(1) + "'");
    while (myRs.next()) {
        String sarkiad = myRs.getString("ad");
        String tarih = myRs.getString("tarih");
        String album = myRs.getString("album");
        String tur = myRs.getString("tur");
        String sure = myRs.getString("sure");

        String tbData[] = {no + "", sarkiad, tarih, album, tur, sure};

        tblModel.addRow(tbData);
        no++;
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
}
no = 1;
```

Butonlardan “Tum kullanicilar” butonunda ise veri tabanına kayıt olmuş tüm kullanıcıların bilgileri sınırlandırılmış şekilde kullanıcıya gösterilmektedir . Böylelikle kullanıcı müzik keşif etmek istediğinde başka kullanıcıların listesinden de kendi listesine müzik eklenmesi sağlanır . Burada kodun çalışma şekli ilk olarak kullanıcının “Tum kullanicilar” butonuna basması ile kullanıcı bilgileri ile karşı karşıya gelmesi SQL sorgu dilindeki “select * from kullanici order by kullaniciid ASC” sağlanır . Daha sonra kullanıcı bu satırlardan birine tıkladığında “jTable1MouseClicked()” metodu devreye girer ve satırdaki tüm bilgiler tek tek alınarak bir “arraylist” e aktarılır . Bu bilgiler sayesinde veri tabanındaki aradığı kişinin listesi kullanıcıya gösterilmek üzere “KullanıcıPop” arayüzü açılır ve buradan koşulları sağlar ise takip edebilir ve kişinin listesinden kendisine müzikler ekleyebilir.

“Album ve Sanatci” butonu sayesinde kullanıcıya veri tabanındaki albumlerin ve sanatçıların bilgilerine ulaşır . Bu oluşumları “SanatciTablo() ve AlbumTablo()” metotları sayesinde daha önce yaptığımız tablo oluşturma işlemlerindeki gibi SQL sorgu dili kullanılarak veriler alınır ve kullanıcıya arayüzde bilgilendirme yapılır .

“Admin Mod” butonu ile “AdminArayüz” ile “AnaProgram” aralarındaki bağlantı sağlanarak Adminleri de bir kullanıcı gibi görüp hem uygulamayı kullanır hem de verilere ekle , sil , güncelle işlemlerini sağlayarak veri tabanını yönetir .

Muzik ekleme işlemleri “Muziklerime Ekle ” butonu üzerinden yapılır . Bu buton sayesinde kullanıcının özel tablosuna veriler eklenir hem de Dinlenmesayisi tablosundaki şarkının kulllanıcının ülkesine bağlı bir biçimde şarkının dinlenme sayısı arttırılır . Böylece bir “Top10” listesi oluşturmamız sağlanmaktadır .

Muzik silme işlemlerini “Muzik Sil” butonu üzerinden yapılır . Böylelikle kullanıcının dinlemekten sıkıldığı müzikleri ya da yanlışlıkla eklediği müzikleri listesinden kaldırmasında etkili olur . SQL sorgu dilinde "DELETE FROM " + kullaniciAD + " WHERE (sarkiad = '" + tblColmnData.get(0) + "') and (tarih = '" + tblColmnData.get(1) + "') " biçiminde sorgulanır böylelikle kullanıcının listesinden şarkı silinmiş olur .

2.5. AdminArayuz

İlk olarak nesne çağırıldığında arayüzde “ Sarki , Album , Sanatci , Dinlenmesayisi ” tablolarını oluşturmak için 4 farklı fonksiyon çağırılır .

```
public AdminArayuz() {
    initComponents();
    SarkiTablo();
    SanatciTablo();
    AlbumTablo();
    DinlenmeSayisiTablo();

}
```

Bu fonksiyonlar sayesinde Admin ‘e veri tabanındaki tüm tablolardaki verileri “jTable” nesnesinin yardımıyla arayüzde Admin ‘e gösterilir .

Tabloların arayüzde oluşturulmasına örnek vericek olursak , sanatci tablosunu ele alalım . İlk olarak “jTable” modeli oluşturulur.

```
DefaultTableModel tblModel = (DefaultTableModel) jTable4.getModel();
```

Oluşturulan bu model yardımıyla oluşturucağımız tablonun satır ve sütunlarına veriler aktarılır .

```
ArrayList<String> ColmnName = new ArrayList<>();

ColmnName.add("ID");
ColmnName.add("AD");
ColmnName.add("ULKE");

for (int i = 0; i < ColmnName.size(); i++) {
    tblModel.addColumn(ColmnName.get(i));
}
```

Yukarıdaki resimde görüldüğü üzere Sanatci tablosunun sütuna “ID,AD,ULKE” verileri girilir. Daha sonra satır bilgisi girmek için veri tabanından Sanatci tablosundan SQL sorgu dili yardımıyla ilgili veriler alınır .

```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();
    ResultSet myRs = myStat.executeQuery("select * from sanatci order by id");
    while (myRs.next()) {
        int idsanatci = myRs.getInt("id");
        String sanatciad = myRs.getString("ad");
        String ulke = myRs.getString("ulke");

        String tbData[] = {idsanatci + "", sanatciad, ulke};

        tblModel.addRow(tbData);

    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
}
```

Daha sonra bu verileri bir “String” türündeki bir diziye depolanır . Buradan da “addRow()” fonksiyonu yardımıyla tablonun satırlarına veri aktarılır .

```
int idsanatci = myRs.getInt("id");
String sanatciad = myRs.getString("ad");
String ulke = myRs.getString("ulke");

String tbData[] = {idsanatci + "", sanatciad, ulke};

tblModel.addRow(tbData);
```

Bu işlem veri tabanındaki ilgili tablodaki verilerin tamamı aktarılana kadar devam eder ve sonunda tablomuzu oluşturmuş oluruz .

Tabloya veri eklerken ilk olarak Admin ‘in kutucuklara girdiği bilgiler “null” olup olmadığı kelimenin en az iki harften oluşur koşulu sayesinde engellenir . Daha sonra girilen veriler ile “EKLE” butonuna basıldığında SQL dilinde “INSERT INTO tablo adı values (bilgi1,bilgi2,..)” şeklinde kaç adet veri varsa onlar SQL kurallarına uygun biçimde girilir. Eğer bu veri tabloda zaten var ise sorgu “0” sonucunu döndür eğer yok ise başarı ile veri ekleme gerçekleşir .

```
private void SanatciEkleMouseClicked(java.awt.event.MouseEvent evt) {
    int idsanatci = Integer.parseInt(Sanatcitxt1.getText());
    String sanatciad = Sanatcitxt2.getText();
    String ulke = Sanatcitxt3.getText();

    if (idsanatci > 0 && sanatciad.length() >= 2 && ulke.length() >= 2) {
        try {
            Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");

            String sorgu = "INSERT INTO sanatci values('" + idsanatci + "','" + sanatciad + "','" + ulke + "')";
            Statement sta = myConn.createStatement();

            int x = sta.executeUpdate(sorgu);

            if (x == 0) {
                JOptionPane.showMessageDialog(null, "Sanatci zaten var !");
            } else {
                JOptionPane.showMessageDialog(null, "Sanatci basariyla olusturuldu !");
                jTable4.setModel(new DefaultTableModel());
                SanatciTablo();

            }
        } catch (Exception e) {

            if (e.getMessage().indexOf("Duplicate") != -1) {
                JOptionPane.showMessageDialog(null, "Bu  Sanatci zaten var !");
            } else {
                JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
            }

        }
    } else {
        JOptionPane.showMessageDialog(null, "idsanatci/sanatciad/ulke kutularini kontrol et !");
    }
}
```

Tablodan veri silme işlemini gerçekleştirirken eklemedeki gibi genel kontroller yapılır ve null hatasının önüne geçilmeye çalışılır . Daha sonra girilen veriler ile “SİL” butonuna basıldığında SQL dilinde “DELETE FROM tablo adı where (primary key,..)” şeklinde birincil anahtar kelimenin içerdiği satır tamamen silinir . Eğer birincil anahtar kelime tabloda bulunmaz ise sorgu “0” döndürür eğer var ise silme işlemi başarılı biçimde gerçekleşir .

```
private void SanatciSilMouseClicked(java.awt.event.MouseEvent evt) {
    int idsanatci = Integer.parseInt(Sanatcitxt1.getText());
    String sanatciad = Sanatcitxt2.getText();
    String ulke = Sanatcitxt3.getText();

    if (idsanatci > 0 && sanatciad.length() >= 2 && ulke.length() >= 2) {
        try {
            Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");

            String sorgu = "DELETE FROM sanatci WHERE (id = " + idsanatci + ") and (ad = '" + sanatciad + "')";

            Statement sta = myConn.createStatement();

            int x = sta.executeUpdate(sorgu);

            if (x == 0) {
                JOptionPane.showMessageDialog(null, "Sanatci zaten yok !");
            } else {
                JOptionPane.showMessageDialog(null, "Sanatci basariyla silindi !");
                jTable4.setModel(new DefaultTableModel());
                SanatciTablo();

            }
        } catch (Exception e) {

            if (e.getMessage().indexOf("Duplicate") != -1) {
                JOptionPane.showMessageDialog(null, "Bu  Sanatci zaten var !");
            } else {
                JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
            }

        }
    } else {
        JOptionPane.showMessageDialog(null, "idsanatci/sanatciad/ulke kutularini kontrol et !");
    }
}
```

Tabloda veri güncellenirken güncelenicek satıra tıklanır ve veriler otomatik olarak kutucukların içine girilir . Daha sonra güncellenicek verinin yenisi girilir . Girilen verilerin kontrolü koşullar(if-else) yardımıyla gerçekleştikten sonra

“GUNCELLE” butonuna basılır. SQL dilinde “UPDATE tablo adi SET sütun1 = bilgi1 , sütun2 = bilgi2 ,... Where (primary key,..) ” olucak şekilde giriş yapılır . Burada girilen bilgilerin türüne göre yazım şeklinin değişmesi gerektiğini sakın unutmayalım . Eğer bu veri tablosunda verinin aynısı var ise sorgu “0” sonucu döndürerek “Zaten veri var ” şeklinde uyarı döndürür diğer sorgulardaki gibi (ekle,sil) Admin’e uyarı mesajı gönderilir . Eğer aynı veri yok ise işlem başarılı mesajı gönderilerek işlem sonlandırılır .

```
private void SanatciGuncelleMouseClicked(java.awt.event.MouseEvent evt) {
    int idsanatci = Integer.parseInt(Sanatcitxt1.getText());
    String sanatciad = Sanatcitxt2.getText();
    String ulke = Sanatcitxt3.getText();

    if (idsanatci > 0 && sanatciad.length() >= 2 && ulke.length() >= 2) {
        try {
            Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
            //UPDATE `musicapidb`.`sanatci` SET `idsanatci` = '13', `sanatciad` = 'Alec3', `ulke` = 'Holland3' WHERE (`sanatciad` =

            String sorgu = "UPDATE sanatci SET id = " + idsanatci + ",ad = '" + sanatciad + "',ulke = '" + ulke
                    + "' WHERE (id = " + TemptxtColmnData.get(0) + ") and (ad = '" + TemptxtColmnData.get(1) + "')";

            Statement sta = myConn.createStatement();

            int x = sta.executeUpdate(sorgu);

            if (x == 0) {
                JOptionPane.showMessageDialog(null, "Sanatci zaten var !");
            } else {
                JOptionPane.showMessageDialog(null, "Sanatci basariyla guncellendi !");
                jTable4.setModel(new DefaultTableModel());
                SanatciTablo();

            }
        } catch (Exception e) {

            if (e.getMessage().indexOf("Duplicate") != -1) {
                JOptionPane.showMessageDialog(null, "Bu  Sanatci zaten var !");
            } else {
                JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
            }

        }
    } else {
        JOptionPane.showMessageDialog(null, "idsanatci/sanatciad/ulke kutularini kontrol et !");
    }
}
```

Bu şekilde tüm tablolarda (sanatci , sarki , album , dinlemesayisi) aynı durum farklılık göstererek benzer biçimde devam eder .

Bu ekleme silme güncelleme işlemleri sırasında bazı özel koşul durumları ile isterlere uygun tablolar arası kontroller yapılır .

İlk olarak “sarkialbumCheck()” fonksiyonu sayesinde veri tabanındaki “sarki” tablosuna veri eklemeden önce girilen şarkının veri tabanında girilen albümün var olup olmadığı kontrol edilir ve sadece albüm veri tabanında var ise şarkı “sarki” tablosuna eklenir .

İkinci kontol ise “sarkiEkleCheck()” fonksiyonu sayesinde veri tabalındaki “sarki” tablosuna veri eklemeden önce girilen şarkının albümü ile şarkının türlerinin aynı olup olmadığını kontrol eder . Türleri aynı olursa şarkı veri tabanındaki tabloya eklenir aksi taktirde şarkı eklenmez ve kullanıcıya konu ile ilgili uyarı mesaji gösterilir.

Üçüncü kontrol ise “albumsanatci()” fonksiyonu ile album tablosuna girilen sanatçının veri tabanında olup olmadığını kontrol eder . Sadece sanatçı veri tabanında var ise sanatçı ekleme albüm ekleme başarıyla gerçekleşir .

Dördüncü kontrol ise “dinlenmeSarkicheck()” fonksiyonu sayesinde “dinlenmesayisi” tablosuna girilen şarkının veri tabanında olup olmadığı kontrol edilir ve veri tabanında var ise tabloya eklenir .

Beşinci kontrol ise “sarkiDinlenmeTabloCheck()” fonksiyonu sayesinde Admin tablodan çıkarken eklediği şarkıların dinlenme sayısını eklemeyi unutursa diye eklenmiş olup olmadığını kontrol eder ve Admin’ e hata döndürerek yeni şarkının dinlenme sayilarını girmesi gerektiğini hatırlatır.

Altıncı kontrol ise “sarkiAlbumTabloCheck()” album ve sarki tablolarındaki album çeşitliliğine bakar . Böylelikle sarki tablosu ve album tablosundaki album şarkıları kontrolleri yapılır.

Yedinci kontrol ise “sanatciAlbumTabloCheck()” ile girilen sanatçıların çeşit sayısı kontrollü yapılır ve böylelikle albüme girilen sanatçının sanatçı tablosuna da girilmesinin unutulmaması gerektiğini hatırlatır .

Bu işlemlerin yanı sıra “kullanicilarSilSarki()” fonksiyonu ile veri tabanından silinen bir şarkı eğer kullanıcılar eklediyse onların listelerinden silinmesini sağlar ve böylelikle şarkı tamamen uygulamadan silinir .

2.6. KullaniciPop

Kullanici bu nesneye AnaProgram ‘deki “Tum Kullanıcılar ” butonuna bastığında gelen “kullanici” tablosundaki müzik listesini aradığı kişinin ve kendisinin bazı önemli bilgilerini aktararak bu nesneye geliyor . Bu bilgiler sırasıyla “kullaniciAD , mykullaniciType , myUserName , kullaniciType , ulke ” değişkenlerinde ilgili bilgileri saklıyor .

```
public KullaniciPop(String kullaniciad, String mykullanicitype, String myusername, String kullanicitype , String ulke)
    initComponents();
    this.kullaniciAD = kullaniciad;
    this.mykullaniciType = mykullanicitype;
    this.myUserName = myusername;
    this.kullaniciType = kullanicitype;
    this.Ulke = ulke;
```

Kullanıcının müzik listesini aradığı kişi ilk olarak önceden takip edilip edilmediğini kontrol eden “checkTakipListesi()” adında bir fonksiyon ile kontrol ediliyor .

```
public void checkTakipListesi() {
    TakipYakala = 0;
    try {
        Connection myConn4 = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
        Statement myStat4 = (Statement) myConn4.createStatement();
        ResultSet myRs4 = myStat4.executeQuery("select * from takiplistesi where kim = '" + myUserName + "'");
        while (myRs4.next()) {
            if (myRs4.getString("kimi").equals(kullaniciAD)) {
                TakipYakala = 1;
            }

        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
    }
}
```

```
if (TakipYakala == 1) {
    jLabel6.setVisible(false);
    TakipEt.setVisible(false);
    musicekle.setVisible(true);
    profilGoster();
}
```

Takip ettiği bir kişi ise müzik listesi direkt tablolar halinde gözüküyor . Eğer takip etmediği kişi ise “Takip Et” butonu gösterilerek kullanıcıya bilgi veriliyor .

Kullanıcı bu butona bastığında kullanıcının takip ettiği kişinin abonelik türüne bakılıyor . Takip ettiği kişi “premium” bir üye ise kullanıcının takip işlemi başarıyla gerçekleşiyor ve kullanıcının adı ve takip ettiği kişinin adı “musicapidb” veri tabanındaki “takiplistesi” tablosuna “kim(Takip eden kişi),kimi(takip ettiği kişi)” olucak şekilde isimleri tabloya veri olarak kaydediliyor . Böylelikle kullanıcının verileri programın o anki duruma bağlı değişmeyip program kapatılsa da veriler saklanmış oluyor .

```
public void ekleTakipListesi() {
    try {
        Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234

        String sorgu = "INSERT INTO takiplistesi values('" + myUserName + "','" + kullaniciAD + "')";
        Statement sta = myConn.createStatement();

        int x = sta.executeUpdate(sorgu);

        if (x == 0) {
            JOptionPane.showMessageDialog(null, "Bu " + kullaniciAD + " zaten takip ediliyor !");
        }

    } catch (Exception e) {

        if (e.getMessage().indexOf("Duplicate") != -1) {
            JOptionPane.showMessageDialog(null, kullaniciAD + " zaten takip ediliyor !");
        } else {
            JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
        }

    }
}
```

Kullanıcı takip etmekten vazgeçer ise ”Takibi Bırak” butonuna basarak takipten çıkabiliyor ve “takiplistesi” tablosundan takip edildiğinde oluşan veri kaldırılıyor.

```
public void silTakipListesi() {
    try {
        Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");

        String sorgu = "DELETE FROM takiplistesi WHERE (kim = '" + myUserName + "') and (kimi = '" + kullaniciAD + "')";
        Statement sta = myConn.createStatement();

        int x = sta.executeUpdate(sorgu);

        if (x == 0) {
            JOptionPane.showMessageDialog(null, "Islem Basariyla Gerceklestirilemedi !");
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());

    }

}
```

Eğer kullanıcının takip etmeye çalıştığı kişi “premium” kullanıcı değil ise kullanıcıya uyarı çıkarılıyor ve kullanıcı bu kişinin listesini göremiyor .

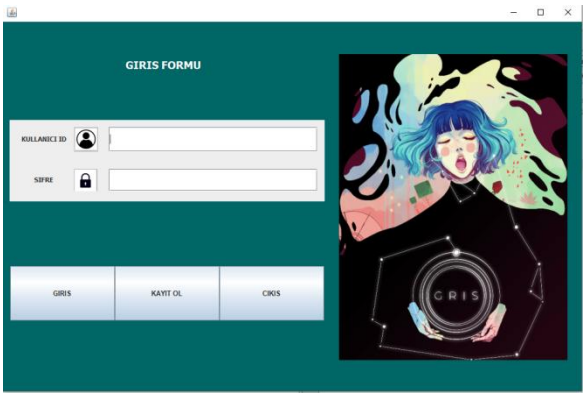
Takip işlemleri olumlu sonuçlandıktan sonra kullanıcının takip ettiği kişinin müzik listesini veri tabanında kullanıcının adıyla oluşturulmuş tablodan müziklerin hepsi alınıyor ve tablolara müzik türlerine uygun ayır ayrı yerleştiriliyor.

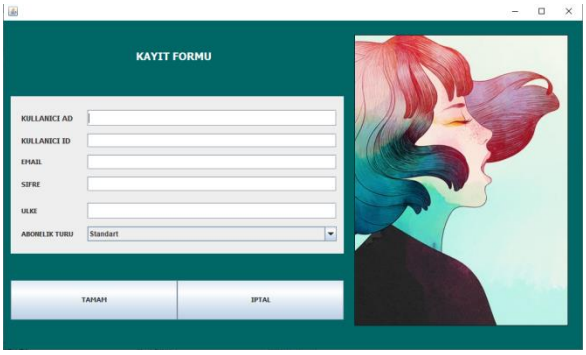
```
try {
    Connection myConn = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/musicapidb", "root", "1234");
    Statement myStat = (Statement) myConn.createStatement();
    ResultSet myRs = myStat.executeQuery("select * from " + kullaniciAD + " where tur = '" + MusicType.get(i) + "'");
    while (myRs.next()) {
        String sarkiad = myRs.getString("sarkiad");
        String tarih = myRs.getString("tarih");
        String album = myRs.getString("album");
        String tur = myRs.getString("tur");
        float sure = myRs.getFloat("sure");
        String tbData[] = {sarkiad, tarih, album, tur, sure + ""};

        tblModel.addRow(tbData);

    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Data Base Hatasi : " + e.getMessage());
}
```

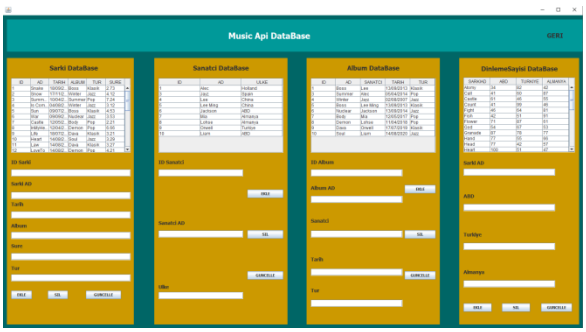
Kullanıcıdan alınan veriler “jTable” türünde bir nesnenin içinde kayıt altına alınıyor . Bu nesne sayesinde ise bilgileri kullanıcıya düzenli bir şekilde sunuyoruz.

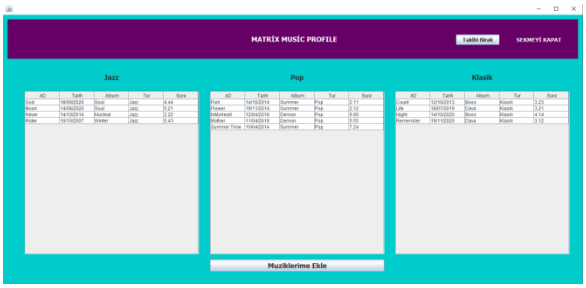
Kullanıcı gördüğü müzikleri kendi listesine eklemek isterse AnaProgram nesnesinin içerisindeki “Muziklerime Ekle” butonundaki aynı fonksiyonları kullanarak burada da listesine ekleme yapabiliyor ve yine aynı mantıkla dinlenme sayısında artış gerçekleştiriliyor .

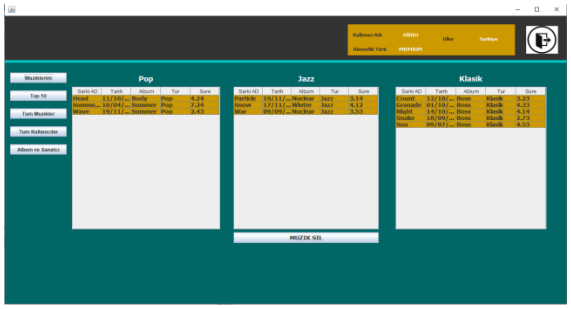
3.DENEYSEL SONUÇLAR

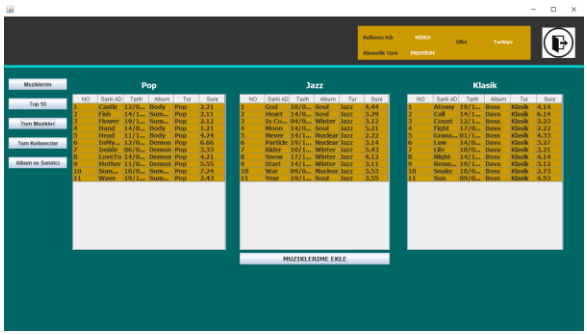


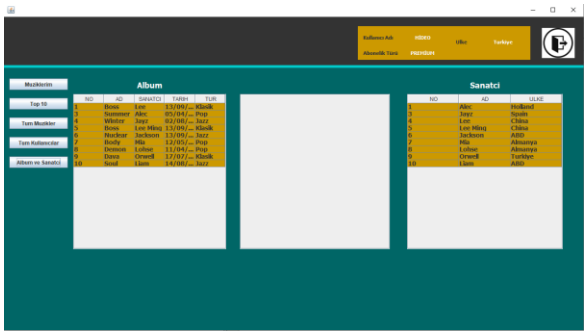


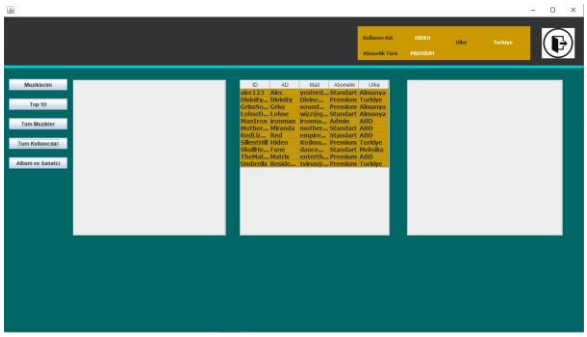








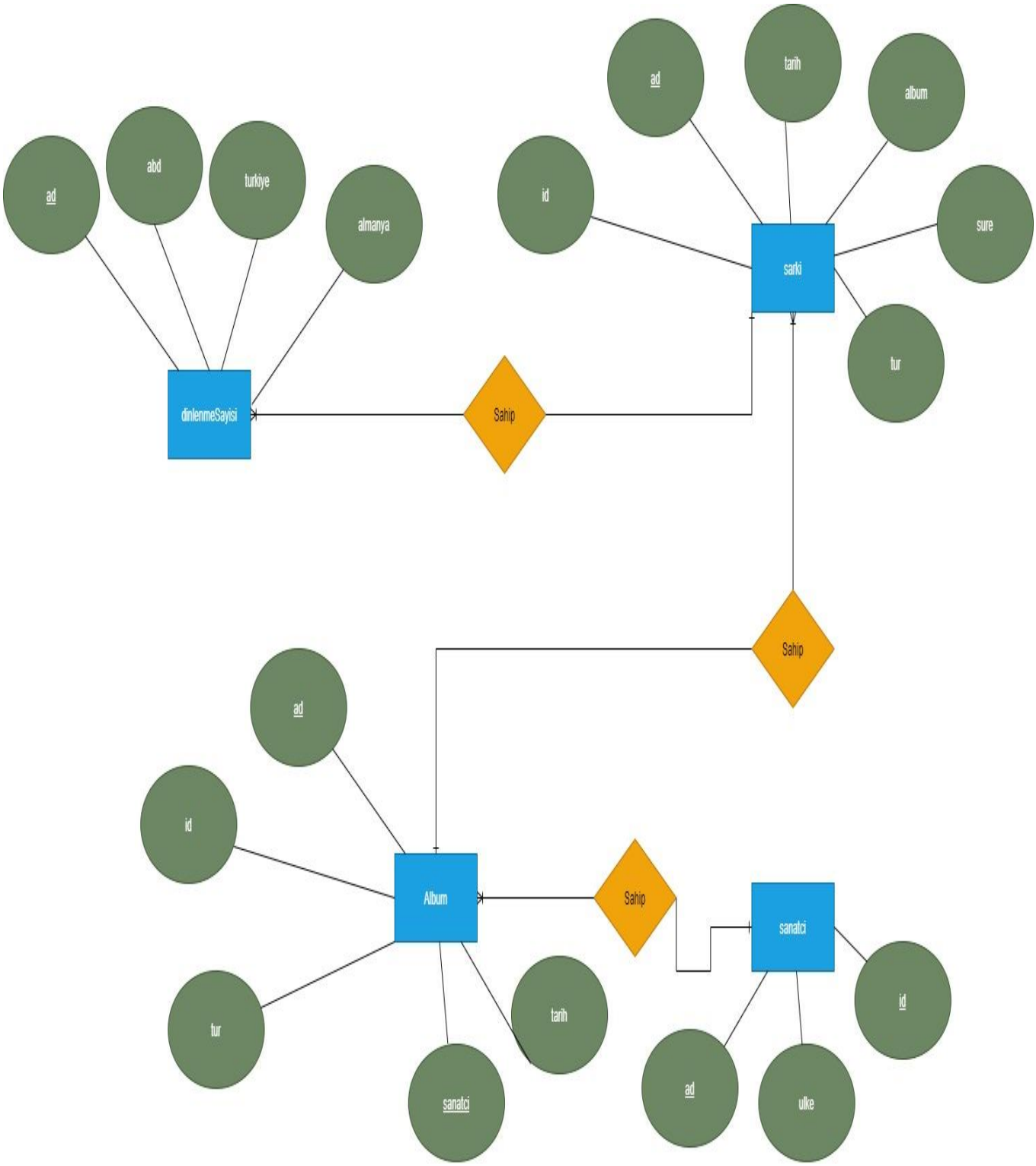


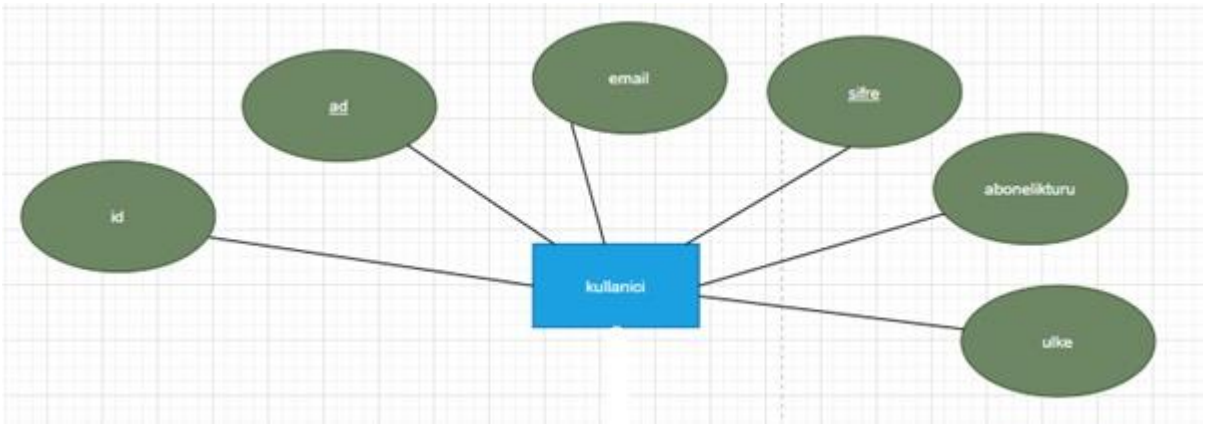


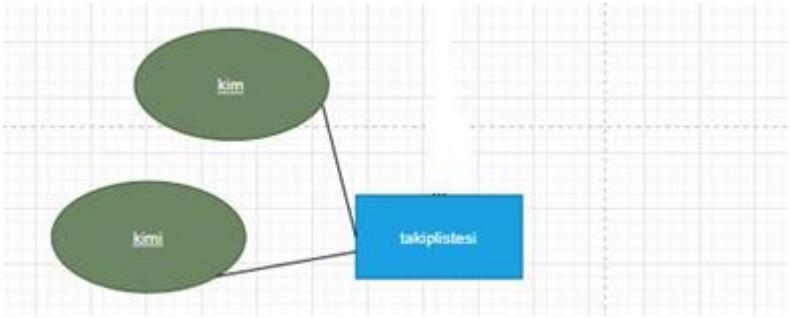
4.SONUÇ

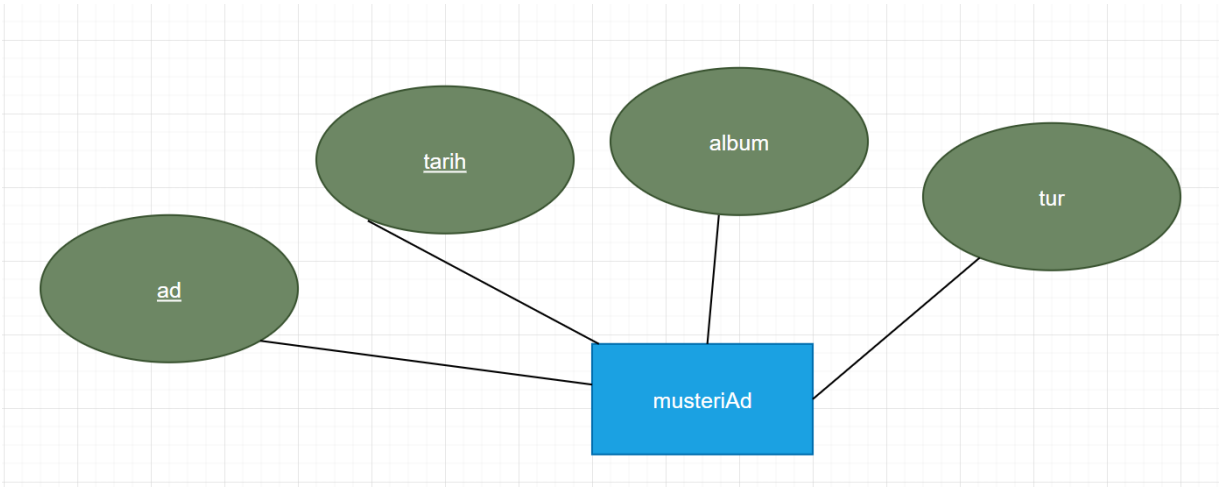
Bu proje sayesinde veri tabanı yönetim sisteminin bir uygulamaya nasıl entegre ediliceğini , SQL sorgu dilini Java gibi programlama dili ile nasıl birlikte hareket ettirildiğini ve bir veri tabanının Normalizasyonlara uyarak nasıl oluşturulması gerektiğini öğrendik .

5.ER DIAGRAM


ad
abd
turkiye
almanya
dinlenmeSayisi
Sahip
ad
tarih
album
id
sure
sarki
tur
Sahip
ad
id
Album
Sahip
sanatci
tur
tarih
sanatci
id
ad
ulke


ad
email
sifre
id
abonelikturu
kullanici
ulke


kim
kimi
takiplistesi


tarih
album
tur
ad
musteriAd

6.REFERANSLAR

6.1.Netbeans ile MySQL Bağlantısı

https://www.youtube.com/watch?v=e3gnhsGqNmI&t=283s

6.2.Veri Tabanında JTable Veri Aktarımı

https://www.youtube.com/watch?v=frafcK6fhdQ

6.3.SQL Genel Komutlar ve Nasıl Kullanılır Eğitim Seti

https://www.youtube.com/watch?v=4BN0sISkgcY&list=PL9OZDx-_1-s39SjOvT-fr-lABr7XfSjdN&index=2

6.4.JTable Hücre Düzenleme

https://www.youtube.com/watch?v=2wpcYQrPEFw

6.5.JTable Tutorial

https://www.youtube.com/watch?v=Tg62AxNRir4&list=PLv1XiDDk9Y4dL6tCfVND9OUCWDoUZ7fz1&index=3

6.6.SQL Normalizasyon Genel Kaynak

https://www.youtube.com/watch?v=goH3b3szCBQ

https://www.youtube.com/watch?v=Y0Y7J_vNahQ

https://www.youtube.com/watch?v=UrYLYV7WSHM&t=41s

6.7.SQL Cascade Delete

https://www.youtube.com/watch?v=BRX6Jhn799g