Envanter Takip Sistemi Mobil Uygulama

Inventory Management System Mobile Application

Kaan Toraman, Eda Obuz, Hacer Sueda Efe

Bilişim Sistemleri Mühendisliği - Teknoloji Fakültesi

Kocaeli Üniversitesi

kaan41tor@gmail.com , edaobuz4@gmail.com , suedaefe@gmail.com

Özet

Bu rapor Android Studio da geliştirilen bir envanter takip sistemi mobil uygulaması projesine ait problem tanımı, yapılan araştırmalar, akış şeması, yazılım mimarisi, veritabanı diyagramı ve projenin genel yapısıyla ilgili bilgiler içermektedir. Bu uygulamada veritabanı olarak mySQL, programlama dili olarak Java kullanılmış, Xampp, Volley, Retrofit gibi yazılım ve kütüphanelerden yararlanılmıştır.

Abstract

This report contains information about the problem definition, researches, flowchart, software architecture, database diagram and general structure of the project of an inventory management system mobile application project developed in Android Studio. In this application, Java was used as a programming language and mySQL was used as a database, also softwares and libraries such as Xampp, Volley and Retrofit were used.

1. Problem Tanımı

Bir işletmenin, yönetimini daha kolay ve düzenli bir şekilde yürütmesi gerekmektedir. Buna bağlı olarak bir yardımcıya ihtiyaç doğmaktadır. İşletmenin ihtiyaçlarını yüksek oranda karşılamak amaçlanmalıdır. Bu amaçla, her şehir için depo kaydetme, ürünlerin depo envanterlerine eklenmesi, ürün alış-verişlerinin kontrol edilmesi ve müşteri/tedarikçi takibinin yapılabileceği bir envanter takip sistemi uygulaması geliştirilmesi ihtiyacı bulunmaktadır.

Bu uygulama aracılığıyla, işletme sahipleri sahip oldukları depolar için ayrıntılı bilgilere erişebilecek, ayrıca ürün stoklarıyla ilgili anlık olarak bilgiye sahip olabileceklerdir.

Bu uygulama sayesinde, müşteri bilgileri de kaydedilebilir ve müşterilerle ilgili ayrıntılı bilgilere erişilebilir.

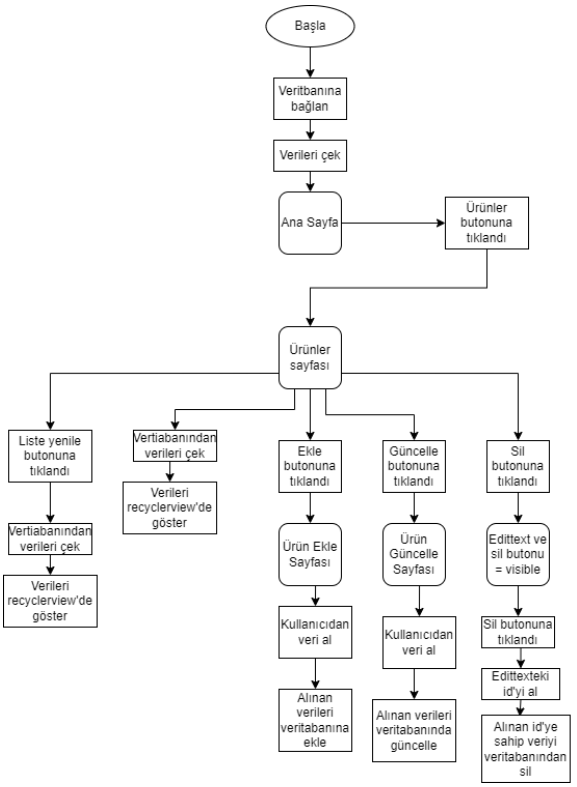
Sonuç olarak, bu envanter takip sistemi uygulaması ile işletme sahiplerinin işlerini kolaylaştırmak ve verimliliklerini artırmak amaçlanmıştır. Bu uygulama sayesinde, işletme sahipleri her zaman ürün stoklarını takip edebilecek, tedarikçileri ve müşterileri yönetebilecek ve işletmelerinin daha başarılı olmasını sağlayabileceklerdir.

2. Yapılan Araştırmalar

Halihazırda bulunan envanter takip sistemi uygulamalarının arayüzleri, içerikleri ve çalışma prensipleri incelenmiştir, bu tür bir uygulamanın MySQL kullanarak Android Studioda nasıl hayata geçirilebileceği araştırılmıştır. Android Studio'yu MySQL'e bağlama, uygulamadan MySQL veritabanına veri ekleme, daha sonra bu eklenen verileri scrollview içeren bir sayfaya çekerek görüntüleme işlemleri gibi işlemler için araştırmalar yapılmıştır. Projeyi mobil uygulama olarak geliştirmek kararlaştırıldığı için hangi IDE'nin kullanılması gerektiğine de karar verilmesi gerekmiştir. Bu sürece kullanılmak istenen yazılım dili şekil vermiştir. C# ve Java arasında bir seçim yapılmıştır ve hangi IDE'lerin uygun olduğu incelenmiştir. Araştırmalar sonucunda projenin Javayla Android Studio ortamında geliştirilmesinin daha uygun olduğuna karar verilmiştir. Ancak MySQL veritabanı kullanıldığı için Android Studio ile direkt bağlantı kurulamamıştır. Ama araştırmalar sonucunda sunucuyu localhost ile çalıştırmak mümkün olursa Retrofit ve Volley gibi kütüphanelerin kullanılabileceği öğrenilmiştir. Veritabanını localhostta çalıştırmak için Xampp ve Wamp server kullanılabileceği bilgisi edinilmiş, bu ikisi arasında Xampp'e ait kaynak daha çok olduğundan dolayı Xampp kullanılmasına karar verilmiştir. Xampp kullanabilmek amacıyla ihtiyaç doğrultusunda bir miktar php öğrenilmesi gerekmiştir.

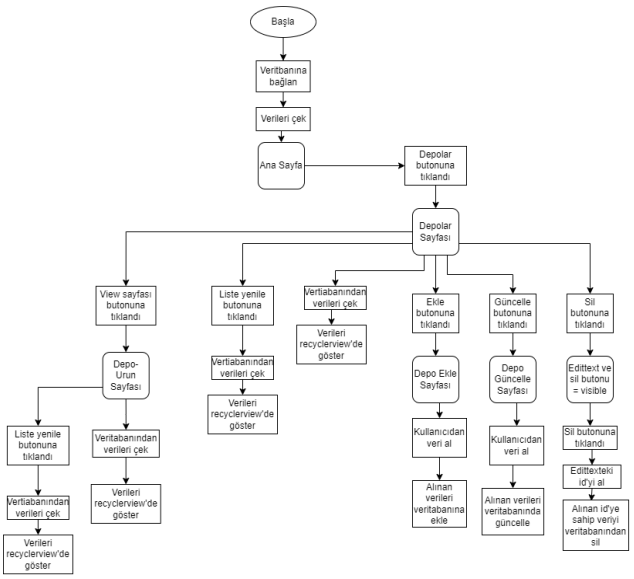
3. Akış şeması

Örnek akış şeması Şekil-1'de verilmiştir. Alış Satış ve Ürün sayfaları bu akış şemasındaki şekilde çalışmaktadır. Verilen ilk akış şemasında, uygulama başlatıldıktan sonra veritabanına bağlanılarak depo ve ürün sayısı bilgileri için veritabanından veri çekilmektedir. Ürünler butonuna basıldığında ise ürünler sayfası açılmaktadır. Ürünler sayfasında veritabanından recyclerviewe veriler çekilmektedir, bu sayfada liste yenileme, veri ekleme, güncelle ve sil butonuna tıklama eylemleri bulunmaktadır. Bu eylemlerde isimleriyle uyumlu bir şekilde kullanıcı uygun sayfalara yönlendirilmekte veya belirtilen işlemler gerçekleştirilmektedir.



Şekil 1: Akış şeması 1

Diğer bir örnek akış şeması Şekil-2'de verilmiştir. Tedarikçi, Müşteri ve Depo sayfaları bu akış şemasındaki şekildeki gibi çalışmaktadır. Bu sayfaların 1. akış şemasındaki sayfalardan tek farkı veritabanından view ile çekilen verilerin gösterildiği başka sayfalara geçiş imkanı sağlayabilmeleridir.



Şekil 2: Akış şeması 2

4. Yazılım mimarisi

4.1 Geliştirme Aşamaları

a. Frontend Geliştirme:

Sayfaların tasarımı belirlendikten sonra activityler ve bunlara ait xml dosyaları projeye eklenmiştir. Önce ana sayfanın tasarımı oluşturulmuştur. Ana sayfada ürünler, tedarikçiler, müşteriler, alışlar, satışlar ve depolar sayfalarına geçiş yapmayı sağlayacak butonlar eklenmiş, butonların şeklini ve renklerini ayarlamak için drawable resource dosyaları oluşturulmuştur. Bu dosyalar oluşturulurken root elementi shape olarak belirlenmiş, bu dosyalarda renk, kenar yuvarlaklığı gibi özellikler belirlenmiştir. Şekil-3'te verilen örnek kodda gradient kısmında startcolor, endcolor özellikleriyle color codelar kullanılarak renk geçişi ayarlanmıştır.

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#F6D4CE"></solid>
    <corners android:radius="20dp"></corners>
    <gradient
        android:angle="90"
        android:startColor="#F6D4CE"
        android:endColor="#F1E4E2"
        android:type="linear"/>

</shape>
```

Şekil 3: Buton Background xml

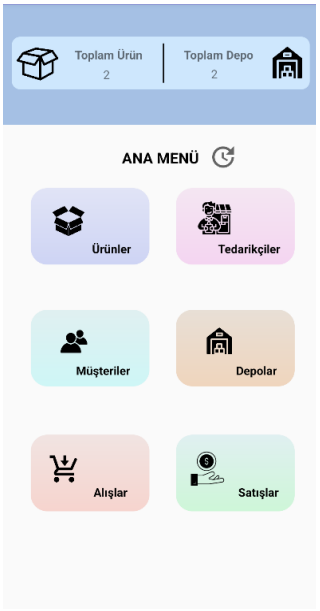
Activity'nin xml dosyasında butonun xml koduna android:background="@drawable/dosya" kodu eklenerek bu dosya butona background olarak atanmıştır.

Butonlara resim eklemek için seçilen pngler drawable dosyasına yapıştırıldıktan sonra imagebutton kullanılmıştır. Xml dosyasında imagebutton'ın xml koduna
android:backgroundTint="@android:color/transparent"

ve android:scaleType="centerInside" kodları eklenmiştir. Bu şekilde pngnin arkası şeffaf olarak ayarlanmış ve resim merkeze konumlandırılmıştır, bu yapılmazsa resmin boyutu değiştirilmek istendiğinde resim kaymaktadır. Şekil-4'te örnek bir buton görüntüsü verilmiştir.

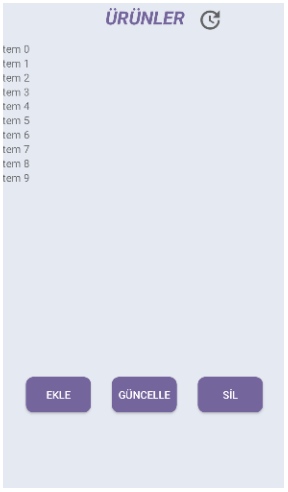
Şekil 4: Buton

Sayfanın üst kısmında ise renkli textviewlar ile bir görünüm oluşturulmuş, resimler eklenmiştir. Bu kısma textviewlar konularak veritabanından çekilen ürün sayısı ve depo sayısı bilgilerinin görüntülenmesi sağlanmıştır. Bir adet sayfa yenile butonu konulmuştur, bu buton drawable dosyasına png değil Android Studio'da hazır bulunan vector assetlerden eklenerek yapılmıştır, image button kullanılarak da eklenen vector asset background olarak atanmıştır. Bunda da backgroundTint="@android:color/transparent" olarak ayarlanmıştır, eğer bu yapılmazsa butonun arka planı gri olarak kalmaktadır. Şekil 5'te sonuç verilmiştir.



Şekil 5: Ana sayfa

Ürünler, tedarikçiler, müşteriler, alışlar, satışlar ve depolar sayfaları için sayfalara recyclerview görünümü, ekle, güncelle, sil ve yenile butonları eklenmiştir, örnek Şekil-6'da verilmiştir. Bu butonların renkleri ve şekilleri için yine ana sayfadaki butonlara yapıldığı gibi drawable resource dosyaları vb. oluşturularak butona background olarak atanmıştır. Recyclerview için app seviyesindeki gradle dosyasına 'androidx.recyclerview:recyclerview:1.2.1' implementation'ı eklenmiştir.

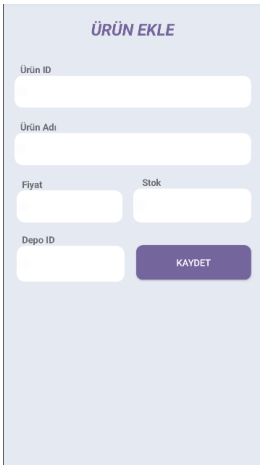


Şekil 6: Ürünler Sayfası

Sil işlemi için birer adet edittext ve buton eklenmiştir ama android:visibility="invisible" olarak atanmıştır. .java sınıfında sil butonunun onclick metodunda bu edittext ve buton visible hale getirilmiştir. Şekil-7'de gösterilmiştir.

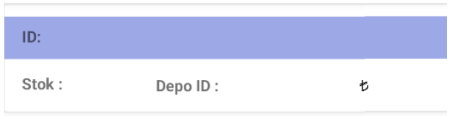
Şekil 7: Silme işlemi

Ekle ve güncelle sayfaları için edittext kullanılarak kullanıcının veri girişi yapacağı kısımlar belirlenmiştir. Bu edittextlerin backgroundları için yine root elementi shape olan bir drawable resource dosyası oluşturulmuş ve radius istenen şekle göre ayarlanmıştır. Textviewlar ile edittexlerin hangisinin hangi veriyi içermesi gerektiği belirtilmiştir. Ardından kaydetme işlemi için buton eklenmiştir.



Şekil 8: Ekleme *Şekil 9:* Güncelleme

Veritabanından veri çekildiğinde verileri sayfalara konan recyclerviewlarda görüntüleyebilmek için cardview içeren layout resource dosyaları oluşturulmuştur. Bu cardviewlar recyclerview itemlerinin her birini temsil etmektedir. Xml dosyasının en dışında constraint layout bulunmaktadır, onun içine cardview eklenmiştir. Cardview'in içine de textviewların konumlandırılmasının kolay olması açısından tekrar bir constraint layout eklenmiştir. Constraint layout için ise recyclerview implementationı eklenen aynı yere 'androidx.constraintlayout:constraintlayout:1.1.3' eklenmiştir. Örnek bir cardview görüntüsü Şekil-10'da verilmiştir.



Şekil 10: Cardview

Ayrıca tedarikçiler, depolar ve müşteriler sayfalarından buton aracılığıyla ek sayfalara geçiş yapılabilmektedir. Bu sayfalarda veritabanımızda oluşturduğumuz view tablolarından çektiğimiz veriler görüntülenebilmektedir. Yine recyclerview ve cardview kullanılmıştır. Örneğin Şekil-11'deki tedarikçiler sayfasında bulunan sağ üstteki butona basarak Şekil-12'deki sayfaya erişilebilmektedir.

Şekil 11: Tedarikçiler sayfası

Şekil 12: View'lar için ek sayfa

Bütün bu sayfalarda tasarımlar bittikten sonra Android Studio'da attributes bölümünden bileşenlerin constraint bağlantıları yapılmıştır, bu bağlantılar yapılmadığı takdirde eklenen bileşenler ekranın sol üst köşesinde toplanır ve istenilen görüntü elde edilemez.

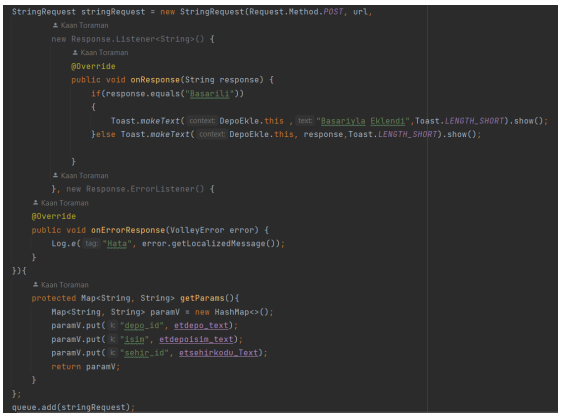
b. Backend Geliştirme:

Uygulamanın kodlanması sürecinde nesne yönelimli bir dil olan Java, işaretleme dili olan Xml, veritabanı tasarlamak için kullanılan Sql ve veritabanı bağlantıları için kullanılan Php kullanılmıştır. Kullanılan IDE Android Studio olduğu için Xml ile sayfaların tasarımı yapılmıştır. Bu tasarımların bağlantısı ve gerçekleştireceği eylemleri ise Java ile kodlanmıştır.

Android Studio, normal şartlar altında MySql veya herhangi bir sunucuyla direkt bağlantı kuramadığından dolayı bağlantı kurması için bir takım HTML kütüphaneleri kullanılması gerekmiştir. Bu projede kullanılan iki adet HTML kütüphanesi vardır. Bunlar RESTful API (Retrofit) ve Volley'dir.

İkiside benzer işler yapmalarına rağmen ikisinin birlikte kullanılmasının en büyük sebebi Retrofit'in HTML.GET işlemlerini daha iyi yapıyorken Volley'in HTML.POST işlemlerini daha iyi yapmasıdır. Bu yüzden birinin kullanılması yerine ikisinin birden kullanılması tercih edilmiştir.

Şekil-13'deki örnek kodda bir sql veritabanındaki bir tabloya volley kütüphanesi yardımıyla veri gönderme işlemi verilmiştir.



```
StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        Kaan Toraman
        new Response.Listener<String>() {
            Kaan Toraman
            @Override
            public void onResponse(String response) {
                if(response.equals("Basarili"))
                {
                    Toast.makeText( context: DepoEkle.this , text: "Basariyla Eklendi",Toast.LENGTH_SHORT).show();
                }else Toast.makeText( context: DepoEkle.this, response,Toast.LENGTH_SHORT).show();

            }
        Kaan Toraman
        }, new Response.ErrorListener() {
    Kaan Toraman
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Hata", error.getLocalizedMessage());
    }
}){
    Kaan Toraman
    protected Map<String, String> getParams(){
        Map<String, String> paramV = new HashMap<>();
        paramV.put( k: "depo_id", etdepo_text);
        paramV.put( k: "isim", etdepoisim_text);
        paramV.put( k: "sehir_id", etsehirkodu_Text);
        return paramV;
    }
};
queue.add(stringRequest);
```

Şekil 13: Volley Kütüphanesinin kullanımına örnek kod

Veriler Retrofit ile veritabanından çekilmektedir. Gelen verileri değişkenlere atamak veya yazdırmak için Json formatına çevirmek gerekmektedir. Bunun için de GSON adı verilen bir kütüphane kullanılmıştır. Bu kütüphane Retrofitin getirmiş olduğu verileri Json formatına çevirerek kullanabileceğimiz bir hale getirilmesini sağlamaktadır. Json formatına çevrilen veriler ise recyclerview görünümü içerisinde yazdırılmaktadır. Bu listeleri kullanmak üzere her tablo için ayrı ayrı adapter sınıfları oluşturmak gerekmektedir. Bu adapter verileri her sınıf için de farklı olarak oluşturulmuş, verileri get ve set eden metotların bulunduğu sınıflardan alınarak, tasarımı xml ile yapılmış cardviewlar aracılığıyla ekrana yazdırma işlemi gerçekleştirilmiştir.

Şekil 14: Retrofit ile veritabanından tablo çekme kod örneği

Veritabanını localhostta Apache sunucusu üzerinde kullanıldığı için veriler Php kodları yardımıyla HTML ile çekilmektedir. Php kod mimarisiyle yazılmış kod tekrar kullanıma uygun olduğu için yazılan .php uzantılı kodların hepsi aslen birbirlerinin kopyasıdır. Sadece kullanılan tabloların ve değişkenlerin isimleri değiştirilerek tekrarlı bir kullanım gerçekleştirilmiştir.

Şekil 15: alis_bilgi adlı tabloya veri eklemeye yarayan örnek Php kodu

c. Veritabanı:

Uygulamanın ihtiyaçları doğrultusunda, MYSQL Workbench kullanarak bir veritabanı tasarlanmış ve uygulamanın ihtiyaçlarına uygun şekilde düzenlenmiştir. Veritabanı, 7 farklı tablo kullanılarak tasarlanmıştır ve bu tablolar birbiriyle ilişkilendirilerek istenilen verim elde edilmiştir. Ayrıca veri tabanında 3 index, 5 trigger ve 3 view bulunmaktadır.
Bu veritabanında yer alan ana tablolar şunlardır:
1-'sehir_bilgi' tablosu: Şehirlerin adını ve kodunu tutmaktadır.
2-'depo_bilgi' tablosu: Depo adını, ID'sini ve bağlı olduğu şehri tutmaktadır.
3-'urun_bilgi' tablosu: Barkod mantığıyla çalışan ürün ID bilgisi, ürün adı, stok adedi, fiyatı ve bulunduğu deponun kodunu tutmaktadır.
4-'satici' tablosu: Her yeni satici eklendiğinde otomatik olarak artan satıcıya ait ID, ad, telefon no, e-posta, adres ve saticinin bulunduğu şehir kodunu tutmaktadır.
5-'musteri_bilgi' tablosu: Her yeni müşteri eklendiğinde otomatik olarak artan müşteriye ait ID, ad, telefon no, e posta, adres ve müşterinin bulunduğu şehir kodunu tutmaktadır.
6-'satis_bilgi' tablosu: Her yeni satış eklendiğinde otomatik olarak artan satış ID, satış yapılan kişiye ait müşteri ID, satılan ürüne ait ürün ID, ürünün nasıl satıldığı bilgisini tutan alış şekli ve ürün adet bilgilerini tutmaktadır.
7-'alis_bilgi' tablosu: Her yeni alış gerçekleştiğinde otomatik olarak artan alış ID, alım yapılan satıcıya ait satici ID, satın alınan ürüne ait ID ve alınan ürüne ait adet bilgilerini tutmaktadır.

Geliştirdiğimiz envanter uygulamasında verimli bir envanter yönetimi sağlamak için "index" yapısı kullanılmaktadır. Bu, veritabanındaki "primary key" yapısı ile benzer bir yapıyla çalışır ve diğer önemli parametrelere erişimde kolaylık sağlamaktadır.
Uygulamada, 3 farklı index kullanılmaktadır. Bunlar, satış bilgi tablosundaki ürün ID bilgisini tutan index, depo bilgi tablosundaki şehir ID bilgisini tutan index ve ürün bilgi tablosundaki depo ID bilgisini tutan index'tir. Bu index'ler, ilgili bilgilere hızlı ve etkili bir şekilde erişmemizi sağlamaktadır ve envanter yönetimimizi kolaylaştırmaktadır.
Veri tabanında "trigger" yapısı kullanarak ürün stok bilgisinde alış ve satış tablolarındaki ürün stok indexine bağlı olarak karşılaşılabilecek düzensizliklerin önüne geçilebilmesi amaçlanmaktadır. Alış işlemlerinde yapılan güncellemeler sonucunda stok bilgisinde dengesiz bir artış meydana gelmektedir. Bu sorunu çözmek amacıyla, 'AFTER INSERT' ve 'AFTER UPDATE' kısımlarında bulunan iki farklı trigger kullanılmaktadır.
Satış işlemlerinde ise yeni satış bilgisi eklendiğinde ve satış bilgisi güncellendiğinde stok bilgisinde iki kez azalma meydana gelmektedir. Bu sorunun çözümü için de 'AFTER INSERT' ve 'AFTER UPDATE' kısımlarında iki farklı trigger kullanılmaktadır.
Ayrıca, ürün stok adedinden fazla satış yapılması gibi bir sıkıntı da mevcuttur, bunu çözmek için 'BEFORE INSERT' kısmına bir trigger eklenmiştir. Bu sayede ürün stok bilgisine ait tüm sorunlar başarılı bir şekilde çözülmüştür.
Uygulamamız, kullanıcıların bilgiye daha kolay bir erişim sağlaması amacıyla 3 farklı view kullanılmaktadır. View mantığı ile birçok tabloya erişim sağlanarak istediğimiz indexleri filtrelenerek bir tablo oluşturulmaktadır. Hangi satıcıdan hangi ürünler alındığına dair bilgilerin bulunduğu tedarikçilerin ürünleri adlı tablo bulunmaktadır. Bu tabloda satıcının adı, ürünün adı ve ürünün fiyat bilgileri tutulmaktadır.
Hangi şehirlerde daha fazla aktif olunduğu bilgisine erişim sağlamak amacıyla müşteri sayısı adlı tablo bulunmaktadır.Bu tabloda şehir id ve toplam müşteri sayısı indexleri bulunmaktadır.Buna bağlı alarak hangi şehirde kaç müşteri bulunduğuna dair bilgi rahatça edinilebilmektedir.
Depo ürünleri tablosu ise hangi ürünün hangi depoda bulunduğu bilgisini saklar ve ürün id'si, ürün adı ve depo id'si bilgilerini içerir.

4.2 Kod Yapısı

Java Kodları

- **Data Sınıfları**

Data sınıfları sayfasının direkt olarak ismini almış sınıflardır. Bu sınıflar verileri veritabanından çekerek get ve set metotları ile değişkenlere atamaktadır. Projede dokuz adet data sınıfı vardır bunlar: Alis, Depo, Musteri, Satici, Satis, Urun, DepoUrun, MusteriSehir ve TedarikciUrun.

```
public class Depo {
    3 usages
    @SerializedName("depo_id")
    private int depoid;
    3 usages
    @SerializedName("isim")
    private String isim;
    3 usages
    @SerializedName("sehir_id")
    private int sehirid;
    Kaan Toraman
    public Depo(int depoid,String isim, int sehirid)
    {
        this.depoid=depoid;
        this.isim=isim;
        this.sehirid=sehirid;
    }
    1 usage   Kaan Toraman
    public int getDepoid() { return depoid; }
    Kaan Toraman
    public void setDepoid(int depoid){this.depoid=depoid;}
    Kaan Toraman
    public String getIsim() { return isim; }
    Kaan Toraman
    public void setIsim(String isim){this.isim=isim;}
    1 usage   Kaan Toraman
    public int getSehirid() { return sehirid; }
    Kaan Toraman
    public void setSehirid(int sehirid){this.sehirid=sehirid;}
}
```

Şekil 16: Depo data sınıfı

- **Adapterler**

Adapterler, recycleview'in içinde nasıl görüntüleneceği ve hangi verilerin görüntüleneceğinin belirlenmesini sağlayan, data sınıflarıyla recycleviewler arasında oluşan bir köprüdür. Projede recycleviewler için özel cardviewlar tasarlanmıştır ve adapterlar içinde bunlar kullanılmıştır. Projede dokuz adet adepter vardır, bunlar : AlisAdapter, DepoAdapter, MusteriAdapter, SaticiAdapter, SatisAdapter, UrunAdapter, DepoUrunAdapter, MusteriSehirAdapter ve TedarikciUrunAdapter.

- **API Interface**

API Interfaceleri, Retrofit yardımıyla yapılacak işleme ait olan php dosyasını alarak data sınıfına ait bir çağrı listesine döndürmektedir. Data sınıfları ve Adapterler gibi dokuz tane API Interface vardır: AlisAPI, DepoAPI, MusteriAPI, SaticiAPI, SatisAPI, UrunAPI, DepoUrunAPI, MusterilerSehirAPI ve TedarikciUrunAPI .

- **Ekleme Sayfaları**

Bu sayfalar veritabanında bulunan tablolara ekleme yapmak için yazılmıştır. Kullanıcılardan edittextler içine veriler girildikten sonra kaydet butonuna basıldığında Volley kütüphanesi yardımıyla bu veriler veritabanına gönderilmektedir. Gönderilen veriler .php uzantılı dosyalar yardımıyla veritabanına eklenmektedir. Projede altı adet ekleme sayfası vardır:AlisEkle, DepoEkle,MusteriEkle, SaticiEkle, SatisEkle ve UrunEkle.

- **Güncelleme Sayfaları**

Güncelleme sayfaları ekleme sayfalarıyla yapısal olarak birebir aynıdır. Sadece çalışma mantıkları birbirinden farklıdır. Ekleme daha önce var olmayan bir veri oluştururken, güncelleme önceden veritabanına eklenmiş bilgileri düzenlemektedir. Güncelleme sayfalarında, ekleme sayfalarına ek olarak kullanıcıdan id de girmesini istemektedir. Bunun nedeni girilen id'ye göre güncelleme işleminin yapılmasıdır. Kullanıcı bilgileri girdikten sonra "Güncelle" butonuna basıldığında veritabanındaki bilgiler güncellenmektedir. Projemizde altı adet güncelleme sayfası vardır: AlisGuncelle, DepoGuncelle, MusteriGuncelle, SaticiGuncelle, SatisGuncelle ve UrunGuncelle.

- **Bilgi sayfaları**

Bu sayfaların isimleri aslen diğer sayfalar gibi bilgi sayfası değildir. Ama bu sayfalar tüm diğer sayfaları bir araya topladığından dolayı bilgi sayfası olarak isimlendirilmiştir.. Bu sayfalara girildiğinde güncelleme ve ekleme sayfalarına yönlendiren butonlar bulunmaktadır. Ama bu sayfaların asıl amacı veri tabanından Retrofit ile çekilen tabloları recycleview içine yazdırıyor olmasıdır. Şekil-14'de bulunan retrofit ile veritabanından kod çekme kısmı bu sayfaların hepsinin içinde bulunmaktadır.

Bilgi sayfalarında tabloları göstermeye ek olarak silme işlemleri de bulunmaktadır. Her bilgi sayfası içerisinde Volley kütüphanesi yardımıyla silme işlemi gerçekleştirilmektedir. Kullanıcı silmek istediği bilginin id'sini girdikten sonra girilen id bir değişkene atanmaktadır ve ardından Şekil-17'de gösterildiği gibi Volley kütüphanesi yardımıyla HTML.POST kullanılarak id veri tabanına gönderilerek silme işlemi gerçekleştirilmektedir.

```
Button alisidsil=findViewById(R.id.btn_alisid_sil);
EditText et_alislar_idsil = findViewById(R.id.et_alislar_idsil);
alisidsil.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String id=et_alislar_idsil.getText().toString();
        RequestQueue queue = Volley.newRequestQueue(getApplicationContext());
        String url ="http://"+ip+"/phpKodlari/alis_sil.php";
        StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        if(response.equals("Basarili")) {
                            Toast.makeText(Alislar.this ,"Basariyla Silindi",Toast.LENGTH_SHORT
                        }else Toast.makeText(Alislar.this, response,Toast.LENGTH_SHORT).show();
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e("Hata", error.getLocalizedMessage());
            }){
            protected Map<String, String> getParams(){
                Map<String, String> paramV = new HashMap<>();
                paramV.put("alis_id", id);
                return paramV;
            }
        };
        queue.add(stringRequest);
        viewJsonDataAlislar();
```

Şekil 17: Silme işlemi için kullanılan kod

Projede bilgi sayfası olarak nitelendirilen temel altı adet sayfa vardır : Alislar, Depolar, Musteriler, Saticilar, Satislar ve Urunler. Bu altı sayfa dışında ise üç adet veri tabanına ait viewları göstermek için kullanılan bilgi sayfası bulunmaktadır. Bu üç bilgi sayfasında sadece tabloya yazdırma işlemi yapılmakta olup diğer işlemler veritabanında otomatik olarak ayarlanmaktadır. Bu üç

viewlere ait bilgi sayfaları ise “MusterilerSehir”, “TedarikcilerinUrunleri” ve “DepolarinUrunleri”dir.

- **Ana Sayfa**

Bu sayfa, uygulama açıldığında ilk açılan sayfadır. Bu sayfa bir merkez noktasıdır. Diğer bütün sayfaların geçiş buradan gerçekleşmektedir. Bu yüzden sayfalar arası geçişi sağlayacak olan butonların içine onClick metotları yazılarak bu mototlarla diğer sayfalara geçiş sağlanmıştır. Aynı zamanda sayfanın üst kısmında bulunan toplam depo ve toplam ürün bilgilerini veri tabanından çekmek için data sınıflarına ait listelerin uzunlukları alınmıştır ve textviewlarla bağlanmıştır.

- **ipAdresi Sınıfı**

Veritabanı Xampp üzerinden çalıştırılsa dahi ipv4 adresinin kullanılması gerekmektedir. Bu sınıf kullanıcının ipv4 adresini tutmaktadır.

PhP Kodları

Php kodları java kodlarıyla localhostta başlatılan veri tabanı arasında köprü görevi gören kodlardır. Projede beş çeşit Php kodu kullanılmıştır: Yazdırma, okuma, silme, güncelleme ve bağlantı.

- **Yazdırma İşlemi**

Yazdırma işleminin Php kodları, veri tabanına ekleme yapılacak noktalarında kullanılmıştır. Örnek olarak, envanter sistemine yeni ürün eklenirken kullanılan “UrunEkle” sayfasındaki Şekil-18 deki gibi, urun_ekle.php dosyası verilebilir.

```
RequestQueue queue = Volley.newRequestQueue(getApplicationContext())
String url ="http://"+ip+"/phpKodlari/urun_ekle.php";
```

Şekil 18: Php kod dosyalarının Volley ile kullanımı

```
<?php
require "baglanti.php";
if(!empty($_POST['depo_id'])&&$_POST['isim']&&$_POST['sehir_id']){
$depo_id=$_POST['depo_id'];
$isim=$_POST['isim'];
$sehir_id=$_POST['sehir_id'];
if($bag){
$sql="INSERT INTO depo_bilgi(depo_id,isim,sehir_id)VALUES('".$depo_id."','".$isim."','".$sehir_id."')";
if(mysqli_query($bag,$sql)){
    echo "Basarili";
}
else echo "Veri Eklenemedi";
}
else echo "Veritabanina Baglanilamadi!!!";
}
?>
```

Şekil 19: urun_ekle.php kod dosyası

- **Okuma İşlemi**

Okuma işleminin Php kodları, veritabanındaki tabloları recycleviewlara yazdırılması gereken yerlerde kullanılmıştır. Okuma işlemine ait Php kodlarının hepsi API Interfaceler içinde kullanılmıştır. Örnek olarak Depolar ekrana yazdırılmak istendiğinde Şekil-20’de gösterildiği gibi bir API Interface üzerinden Php kod dosyasına erişim sağlanılması verilmiştir.

```
public interface DepoApi {
    Kaan Toraman
    @GET("depolar.php")
    Call<ArrayList<Depo>> callArraylist();
}
```

Şekil 20: DepoAPI dosyasında php kod dosyasının Retrofit ile kullanılması

```
<?php
header('Content-Type:application/json');
require "baglanti.php";
$data=array();
if($bag)
{
    $sql="SELECT *FROM depo_bilgi";
    $result=mysqli_query($bag,$sql);
    if(mysqli_num_rows($result)!=0)
    {
        $i=0;
        while($row=mysqli_fetch_assoc($result))
        {
            $data[$i]=$row;
            $i++;
        }
        echo json_encode($data, JSON_PRETTY_PRINT);
    }
}else echo "Veritabanina Baglanmadi";

?>
```

Şekil 21: depolar.php kod dosyası

- **Silme İşlemi**

Silme işleminin Php kodları, bilgi sayfaları olarak adlandırılan sayfalarda kullanılmıştır. Adından da anlaşılacağı üzere eklenen verileri veritabanı üzerinden silmek üzere kullanılmaktadırlar.

```
<?php
require "baglanti.php";
if(!empty($_POST['musteri_id']))
{
    $id=$_POST['musteri_id'];
    if($bag)
    {
        $sql="DELETE FROM musteri_bilgi WHERE musteri_id = " . $id;
        if(mysqli_query($bag,$sql))
        {
            echo "Silindi";
        }
    }else echo"Silinemedi";
}else echo"Veritabanina Baglanmadi";
?>
```

Şekil 22: musteri_sil.php kod dosyası

- **Güncelleme İşlemi**

Güncelleme işleminin Php kodları, veritabanına eklenen verileri güncellemek için kullanılmıştır. Kullanım olarak yazdırma ile aynıdır. Kod olarak ise sql kısımları farklıdır, birisin de INSERT INTO, diğerinde ise UPDATE kullanılmaktadır.

Şekil 23: alis_update.php kod dosyası

- **Bağlantı İşlemi**

Bağlantı işlemine ait Php kodu projedeki en önemli php kodudur. Çünkü diğer dört işlemde de bu dosyaya bağlanılmaktadır. Diğer dört işlemin kodlarının başlangıcında yazan "require 'baglanti.php'" kısmı, veritabanına bağlantıyı sağlamaktadır. Bu bağlantı sağlandıktan sonra işlemler gerçekleştirilebilmektedir.

```php
<?php
$db="envanter";
$user="root";
$pass="";
$sv="localhost";
$bag =mysqli_connect($sv,$user,$pass,$db);

?>
```

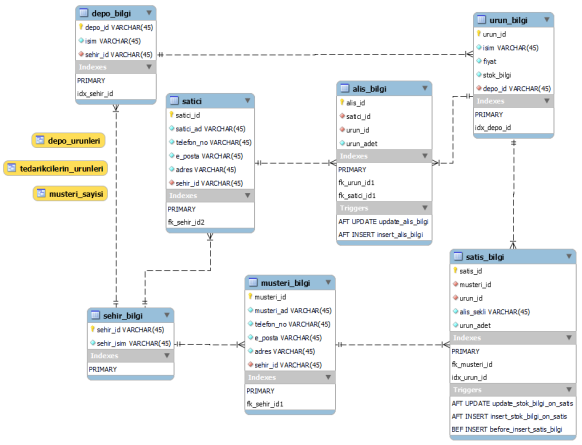
Şekil 24: baglanti.php kod dosyası

Bağlantı için kullanılan php dosyasına diğer php kodlarının aksine, uygulamanın içinden ulaşılamaz. Bu php dosyası diğer phpler içinde kullanılmaktadır.

5. Veritabanı diyagramı

Veritabanı ER diyagramı Şekil-25'de verilmiştir. Veri tabanında urun_bilgi, satis_bilgi, alis_bilgi, musteri_bilgi, depo_bilgi, sehir_bilgi ve satici adında toplam 7 adet tablo bulunmaktadır. Bu tablolar birbirine foregin key'ler aracılığı ile bağlıdır. urun_bilgi ve depo_bilgi tablolarındaki id'ler kullanıcı tarafından girilmektedir, barkod niteliğinde olsun istenmiştir. Diğer tablolardaki id'ler kendiliğinden otomatik olarak artmaktadır.

Diyagramın sol tarafında görülen sarı alanlar ise oluşturulan view tablolarını temsil etmektedir.



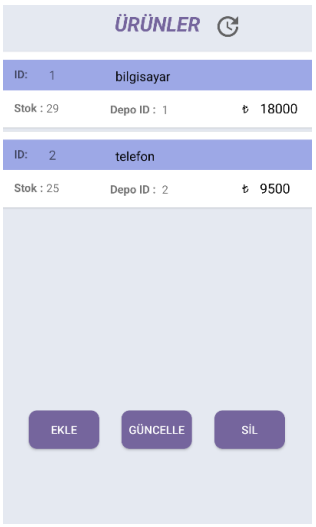
Şekil 25: ER Diyagramı

6. Genel Yapı

6.1 Projenin Arayüzü

Uygulama toplam 22 sayfadan oluşmaktadır, ilk olarak kullanıcıyı ana sayfa karşılamaktadır, ana sayfada en üstteki bölümde veri tabanından bilgi çekilerek işletme sahibinin kaç adet ürünü ve deposu olduğu gösterilmektedir. Onun altında ana menü bulunmakta olup buradaki butonlarla kullanıcı ürünler, depolar, müşteriler gibi diğer sayfalara yönlendirilmektedir.

Bu sayfaların her birinde recyclerview ve butonlar bulunmaktadır, recyclerviewda veritabanından çekilen bilgiler cardviewler aracılığıyla kullanıcıya gösterilmektedir. Şekil-26'da bu durum gösterilmiştir.



Şekil 26: Ürünler Sayfası

Recyclerviewin altındaki ekle güncelle ve sil butonlarıyla kullanıcı ayrı ayrı sayfalara yönlendirilerek edittextler aracılığıyla kullanıcı tarafından veritabanına veri girişi yapılmasını sağlanmaktadır. Veritabanına Insert ve Update işlemleri o sayfalar aracılığıyla gerçekleştirilmektedir.

Veri tabanından delete işlemi için ise sil butonuna basıldığında birer adet edittext ve buton görünür hale gelerek kullanıcıdan silmek istediği verinin id si alınmakta ve silme işlemi bu şekilde gerçekleştirilmektedir.

Tedarikçiler, depolar ve müşteriler sayfalarından da view tablolarındaki verileri görüntülemek için ayrı sayfalara erişilebilmektedir.

6.2 Projenin Kod Kısmı

Uygulamanın kod kısmı 44 sayfa java kodu, 28 sayfa php kodundan oluşmaktadır. Java kodları 6 farklı türde yazılmıştır. Bunlar sırasıyla API interfaceleri, data sınıfları, adapterlar, ekleme sınıfları , güncelleme sınıfları, bilgi sayfalarının sınıfları, ana menünün kod sayfası ve ipAdresi sınıfıdır. Php kodları 5 farklı türde yazılmıştır. Bunlar sırasıyla yazdırma işlemleri, okuma işlemleri, güncelleme işlemleri, silme işlemleri ve bağlantı işlemidir.

Projede Mysql sunucusuna bağlantı kurmak için Html metotları olan POST ve GET metotlarının kullanılması gerekmiştir. Bunun için RESTful API (Retrofit) ve Volley kütüphaneleri kullanılmıştır. Retrofit okuma işlemi için, Volley yazdırma, silme ve güncelleme işlemleri için kullanılmıştır. Retrofit kullanırken okunan verileri Json olarak alınıp kullanılması gerektiği için Gson kütüphanesi kullanılmıştır.

6.3 Projenin Veritabanı

Uygulama ihtiyaçları doğrultusunda gerekli kriterlere bağlı olarak veritabanı MYSQL Workbench üzerinden tasarlanmıştır. Veritabanı 7 ana tablodan oluşmaktadır. Uygulamanın ihtiyacına yönelik olarak veritabanına index ve trigger yapıları eklenmiştir. Veritabanında 5 trigger ve 3 index kullanılmaktadır. Böylece veritabanının uygulama için verimi arttırılmıştır. Ayrıca uygulamayı kullanmakta olan kullanıcıya artı olarak kolaylık sağlamak amaçlanmıştır. Bu amaçlar doğrultusunda veritabanına 3 farklı view eklenmiştir. Bu view mantığı ile farklı tablolardan istenen indexler birleştirilerek 3 farklı tablo oluşturulmuştur. Bu tablolar sayesinde kullanıcılara kullanım kolaylığı sağlanarak birçok bilgiye erişim kolaylaştırılmıştır.

7- Sonuç

Kararlaştırılan ihtiyaçlar doğrultusunda tanımlanmış olan problemin çözümüne yönelik Android Studio üzerinde bir uygulama geliştirilmiştir. Bu uygulama, işletme sahiplerine pek çok yönden yardımcı olabilecek nitelikte olup işletme envanterinde bulunan ürünleri, depoları, satış ve alış işlemlerini, satıcıları ve müşterileri düzenlemek ve listelemek için kullanılmaktadır.

Uygulama, kullanıcı dostu bir arayüz ile tasarlanmış olup kullanımı oldukça basittir, böylece işletme sahipleri ve çalışanları hızlıca uygulamaya adapte olabilmektedir.

Uygulama, işletmelerin envanterinde bulunan ürünlerin takibini kolaylaştırmaktadır. Ürünlerin stok durumunu, fiyatlarını, bulunduğu depoları kaydetme imkanı sunmaktadır. Böylece işletme sahipleri, envanterin güncel durumunu anlık olarak görebilmekte ve eksiklikleri tespit edebilmektedir.

Aynı zamanda uygulama, satış ve alış işlemlerini yönetmeyi kolaylaştırmaktadır. Müşteri bilgileri kaydedilebilmekte, alış ve satışların takibi yapılabilmektedir. Bu sayede işletme sahipleri, gelir ve giderlerini takip edebilmektedir.

Uygulama ayrıca depo yönetimini de kolaylaştırmaktadır. Ürünlerin hangi depoda bulunduğu takip edilebilmekte ve depo seviyeleri kontrol edilebilmektedir. Böylece işletme sahipleri, envanterin verimli bir şekilde dağıtılmasını sağlayarak stok maliyetlerini düşürebilir ve müşteri taleplerini daha hızlı karşılayabilmektedir.

Uygulama aynı zamanda satıcı ve müşteri yönetimini de desteklemektedir. Satıcı ve müşteri bilgileri kaydedebilmekte, alış satış verilerin bu bilgilere ilişkilendirebilimektedir. Bu sayede işletme sahipleri, satıcı performansını takip edebilir, müşteri tercihlerini anlayabilir ve pazarlama stratejilerini daha etkili bir şekilde yönlendirebilmektedir.

Referanslar

- https://www.codeseasy.com/google-volley-android/ [27.04.2023]
- https://google.github.io/volley/ [26.04.2023]
- https://square.github.io/retrofit/ [04.05.2023]
- https://github.com/square/retrofit/tree/master/retrofit-converters/gson [04.05.2023]
- https://www.javatpoint.com/creating-mysql-database-with-xampp [26.04.2023]
- https://developer.android.com/topic/architecture [24.04.2023]
- https://www.techypid.com/display-mysql-data-in-android-studio-recyclerview/ [04.05.2023]