

KURYEM STAJ PROJESİ

Projemiz Getir mantığında çalışmasını plaladığımız bir proje, fakat ayrı olarak kuryelerin freelance olarak iş yapmak için kendi üyeliklerini oluşturup iş alma imkanı sağlaması baz alınarak oluşturulmuştur

Kurye Üyeliği: bizim düşüncemize göre kullanıcının motoru veya kuryelik yapabileceği herhangi bir motorlu taşıta sahip olması uygulamada kurye olarak üye olma şansı tanıyacaktır,kurye üyeliğine sahip olan kullanıcıların konum bildirme yetkisine sahip olması planlanarak hesap ayarları kısmına konum bildir butonu tanımlanmıştır

Satıcı Üyeliği: bir diğer üyelik modelimiz ise satıcı üyeliği olup, satıcı olarak üye olan kişiler ise kendi mağzalarında, marketlerinde bulunan ürünlerin bilgisini ve resmini hesap ayarları kısmında sadece satıcı olarak üye olan kişilere görünebilirliği olan ürün ekle butonuna basarak ürünlerini veri tabanına ekleyip uygulamada listeleme şansına sahiptirler.

Müşteri Üyeliği: Müşteri Üyeliği ise hiçbir ayrıcalığa sahip olmayıp normal e ticaret uygulamalarındaki gibi sepete ekleme, sepetten silme gibi etkileşimlere sahiptirler.

Proje Adımları

Adım 1: projede kullanılacak kütüphaneler için gradle kısmında gerekli implementation işlemleri yapıldı

```
implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
implementation 'androidx.core:core-ktx:1.6.0'
implementation 'androidx.appcompat:appcompat:1.3.1'
implementation 'com.google.android.material:material:1.4.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
implementation 'androidx.legacy:legacy-support-v4:1.0.0'
implementation 'androidx.navigation:navigation-fragment:2.3.5'
implementation 'androidx.navigation:navigation-ui:2.3.5'
implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
implementation 'com.google.android.gms:play-services-maps:17.0.1'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
// implementation 'com.google.firebase:firebase-storage:11.8.0'
// compile 'com.google.firebase:firebase-core:11.8.0'
// implementation 'com.google.firebase:firebase-auth:11.8.0'
// compile 'com.google.firebase:firebase-database:11.8.0'
// compile 'com.google.firebase:firebase-messaging:11.8.0'

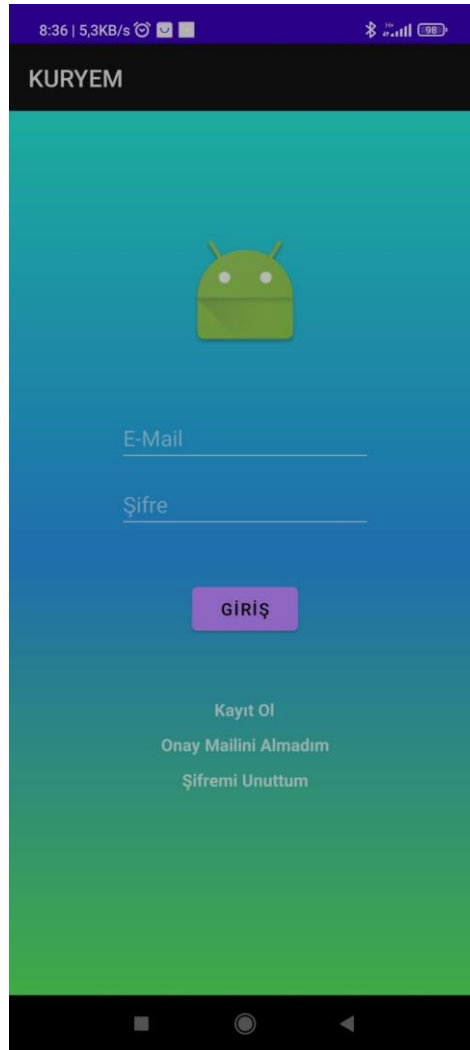
implementation 'com.google.firebase:firebase-analytics-ktx'
implementation 'com.google.firebase:firebase-database:20.0.1'
implementation 'com.google.firebase:firebase-storage:20.0.0'
implementation 'com.firebaseui:firebase-ui-storage:7.2.0'
implementation 'com.google.firebase:firebase-auth:21.0.1'
// Import the BoM for the Firebase platform
implementation platform('com.google.firebase:firebase-bom:15.4.0')

// Declare the dependency for the Realtime Database library
// When using the BoM, you don't specify versions in Firebase library
dependencies
implementation 'com.google.firebase:firebase-database-ktx'
implementation 'com.squareup.picasso:picasso:2.5.2'
implementation 'de.hdodenhof:circleimageview:3.1.0'
implementation 'com.google.firebase:firebase-firestore-ktx'
```

Adım 2:Kullanıcı izinleri için manifest kısmında gerekli izinler istendi

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />  
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Adım 3:Login Activity kısmının xml kısmı yapıldı giriş ekranında olması gereken butonlar,textler ve edit textler gibi tasarım elemanları tasarım sayfasına dünenli şekilde yerleştirilerek gerekli id tanılamaları ve boyutlandırmalar yapıldı



(Login Sayfası Görünüm)

Adım 3:Tasarım sayfasında bulunan elemanlara gerekli tıklama fonksiyonları eklendi ve bu tıklama sonucunda yapılacak işlemler bu kod bloklarının içerisinde tanımlandı

```
kayitOl.setOnClickListener {
    var intent = Intent(this, RegisterActivity::class.java)
    startActivity(intent)
}

newSifre.setOnClickListener {
    var dialogSifreyenile = SifremiUnuttumDialogFragment()
    dialogSifreyenile.show(supportFragmentManager, "sifretrydialog")
}

onayMailGonder.setOnClickListener {
    var dialogGoster = OnayMailTryFragment()
    dialogGoster.show(supportFragmentManager, "gosterdialog")
}
```

...gibi

Adım 4:Giriş Butonu tıklama fonksiyonları Firebase Authentication yapısı kullanılarak yapılan girişin sistmede kayıtlı olup olmadığı, yapılan kayıt sonrası E-maile gönderilen ailin onaylanıp onaylanmadığı, Email ve şifre uyuşup uyuşmadığı durumları kontrol eden fonksiyonlar eklendi

```
progressBarGoster()

if (eMailgiris.text.isNotEmpty() && sifreGiris.text.isNotEmpty()) {
    FirebaseAuth.getInstance().signInWithEmailAndPassword(
        eMailgiris.text.toString(),
        sifreGiris.text.toString()
    ).addOnCompleteListener(object : OnCompleteListener<AuthResult> {
        override fun onComplete(p0: Task<AuthResult>) {

            if (p0.isSuccessful) {
                progressBarGizle()
                Toast.makeText(
                    this@LoginActivity,
                    "Giriş Yapılıyor... : " +
                    FirebaseAuth.getInstance().currentUser?.email,
                    Toast.LENGTH_SHORT
                ).show()
                if (!p0.result?.user?.isEmailVerified!!) {
                    FirebaseAuth.getInstance().signOut()
                }
            }
        }
    })
}
```

```

        } else {
            progressBarGizle()
            Toast.makeText(
                this@LoginActivity,
                "Giriş Yapılamadı, Email veya Şifre Yanlış ",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
})

} else {
    Toast.makeText(this@LoginActivity, "Boş Alanları Doldurunuz",
        Toast.LENGTH_SHORT
    ).show()
    progressBarGizle()
}
}

```

(Giriş Yap Butonu Kod Yapısı)

```

private fun initMyAuthStateListener() {
    mAuthStateListener = object : FirebaseAuth.AuthStateListener {
        override fun onAuthStateChanged(p0: FirebaseAuth) {
            var kullanıcı = p0.currentUser

            if (kullanıcı != null) {
                if (kullanıcı.isEmailVerified) {
                    var intent = Intent(this@LoginActivity,
                        MainActivity::class.java)
                    startActivity(intent)
                    finish()
                } else {
                    Toast.makeText(
                        this@LoginActivity,
                        "Lütfen Giriş Yapmak İçin Size Gönderilen Maili
                        Onaylayınız",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }
        }
    }
}

```

(Kullanıcının Üyeliği Hangi Aşamada olduğuna göre sisteme giriş yapmasını sağlayacak olan kod bloğu)

Adım 5: Login Ekranı İşlemleri tamamlandıktan sonra Üyeliği olmayan kullanıcıların üye olması için lazım olan ve kayıt ol kısmına verdiğimiz sayfa değişikliği yetkisiyle yönlendirileceğimiz sayfa olan Register Activity yani kayıt ol kısmının tasarım kısmına xml kodlarına başladım.

Gerekli buton, textview ve edittext tanımlamalarından sonra tasarımsal olarak güzel görecektik imageviewler ekledim, sonrasında ise gerekli boyutlandırma, id tanımlama gibi işlemleri tamamlayarak tasarım kısmını bitirdim

(Kayıt Ol Kısmının Ekran Görüntüsü)

Adım 6: Kayıt Ol Kısımının Tasarım İşlemleri bittikten sonra ise kullanıcıdan aldığımız verileri kayıt ol butonu ile Firebase Göndermemiz gereken gerekli kod parçasını yazmaya başladım

```
kayit_ol.setOnClickListener {  
    if (eMail.text.isNotEmpty() && sifre.text.isNotEmpty() &&  
        sifreTekrar.text.isNotEmpty()) {  
        if (sifre.text.toString().equals(sifreTekrar.text.toString())) {  
            yeniUyeKayit(eMail.text.toString(), sifre.text.toString())  
        } else {  
            Toast.makeText(  
                this,  
                "Şifreler Eşleşmiyor, Lütfen Kontrol Edin",  
                Toast.LENGTH_SHORT  
            ).show()  
        }  
    } else {  
        Toast.makeText(this, "Boş Alanları Doldurunuz", Toast.LENGTH_SHORT)  
    }  
}
```

öcelikle kayıt ol kısmındaki şifrelerin ve boş alanları kontrolünü yaparak istediğimiz bilgileri am olarak almışmıyız bunun kontrolünü yaparak yeniUyeKayit fonksiyonunu buna göre çağırdık

```
private fun yeniUyeKayit(mail: String, usifre: String) {  
    progressBarGoster()  
  
    FirebaseAuth.getInstance().createUserWithEmailAndPassword(mail, usifre)  
        .addOnCompleteListener(  
            object : OnCompleteListener<AuthResult> {  
                override fun onComplete(p0: Task<AuthResult>) {  
  
                    if (p0.isSuccessful) {
```

```

        var addDatabaseUser = Kullanici()
        addDatabaseUser.isim = isim.text.toString()
        addDatabaseUser.kullanici_adi = userNameRegister.text.toString()
        addDatabaseUser.e_mail = eMail.text.toString()
        addDatabaseUser.kullanici_id =
            FirebaseAuth.getInstance().currentUser?.uid
        addDatabaseUser.profil_resmi = ""
        addDatabaseUser.telefon = telefon.text.toString()
        addDatabaseUser.adres=userAdres.text.toString()

        if (checkBoxCustomer.isChecked){
            addDatabaseUser.seviye="1"
        }else if (checkBoxSales.isChecked){
            addDatabaseUser.seviye="2"
        }else if (checkBoxCurye.isChecked){
            addDatabaseUser.seviye="3"
        }
        else{
            Toast.makeText(this@RegisterActivity,"Lütfen Satıcı
Veya Müşteri Kısmını Doldurunuz",Toast.LENGTH_SHORT).show()
        }

        FirebaseDatabase.getInstance().reference
            .child("kullanici")
            .child(FirebaseAuth.getInstance().currentUser?.uid!!)
            .setValue(addDatabaseUser).addOnCompleteListener {

task ->
                if (task.isSuccessful) {
                    Toast.makeText(
                        this@RegisterActivity,
                        "Üye Kaydedildi ",
                        Toast.LENGTH_SHORT
                    )
                        .show()
                    onayMailiGonder()
                    FirebaseAuth.getInstance().signOut()
                    loginSayfasinaGit()
                } else {
                    Toast.makeText(
                        this@RegisterActivity,
                        "Üye Kaydedilirken Bir Sorun Oluştı " +
p0.exception?.message,
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }

            eMail.setText("")
            sifre.setText("")
            sifreTekrar.setText("")
        } else {
            Toast.makeText(
                this@RegisterActivity,
                "Üye Kaydedilirken Bir Sorun Oluştı " +
p0.exception?.message,
                Toast.LENGTH_SHORT

```



```

        ).show()
    }
}
})
progressBarGizle()
}

```

burada ise yenUyeKayıt Fonsiyonumuzu yazdık ve bu fonksiyona parametre olarak kullanıcı maili ve şifresini gönderdik ki işlemleri bunlara göre yapalım, bu bilgileri ise tanımladığımız tasarımdaki girdi kısılarından aldık, sonrasında bu girdileri daha rahat ve toplu şekilde kullanabilmek adına Kullanici adında bir Class oluşturduk

```

package com.example.firebasekotlin

class Kullanici {
    var isim: String? = null
    var kullanıcı_adi: String? = null
    var e_mail: String? = null
    var telefon: String? = null
    var adres : String? = null
    var profil_resmi: String? = null
    var ürün_resmi : String? = null
    var seviye: String? = null
    var kullanıcı_id: String? = null

    constructor(
        isim: String,
        kullanıcı_adi: String,
        e_mail: String,
        telefon: String,
        adres : String,
        profil_resmi: String,
        ürün_resmi:String,
        seviye: String,
        user_id: String
    ) {
        this.isim = isim
        this.kullanıcı_adi = kullanıcı_adi
        this.e_mail = e_mail
        this.telefon = telefon
        this.adres=adres
        this.profil_resmi = profil_resmi
        this.ürün_resmi=ürün_resmi
        this.seviye = seviye
        this.kullanıcı_id = user_id
    }

    constructor()
}

```

(Kullanici() Classı)

Oluşturduğumuz bu classtan addDataBaseUser Adında yeni bir nesne oluşturarak o andaki kullanıcının verilerini bu classa atadık, sonrasında ise checkbox kontrolü yaparak kullanıcının hangi tür bir kullanıcı olacağının kontrolünü her kullanıcı türüne farklı seviye vererek bu seviye işleminide addDataBaseUser nesnemizin seviye kısmına atadık

Adım 6:addDataBaseUser nesnemizi Fire Base Database işlemlerini kullanarak kullanıcının ID sine göre kullanıcı veritabının bir çocuğu olarak kullanıcı database'ine kayıt işlemini bitirdik, kayıt işlemi olmaması durumunda ise olabilecek hataları Toast Mesajlar ile kullanıcıya bildirme işlemlerini yaptık.

```
private fun onayMailiGonder() {  
    var kullanıcı = FirebaseAuth.getInstance().currentUser  
  
    if (kullanıcı != null) {  
        kullanıcı.sendEmailVerification()  
        .addOnCompleteListener(object : OnCompleteListener<Void> {  
            override fun onComplete(p0: Task<Void>) {  
                if (p0.isSuccessful) {  
                    Toast.makeText(  
                        this@RegisterActivity,  
                        "Kayıt işlemini tamamlamak için size gönderilen maili  
onaylayın",  
                        Toast.LENGTH_SHORT  
                    ).show()  
                } else {  
                    Toast.makeText(  
                        this@RegisterActivity,  
                        "Onaylama Maili Gönderilemedi",  
                        Toast.LENGTH_SHORT  
                    ).show()  
                }  
            }  
        })  
    }  
}
```

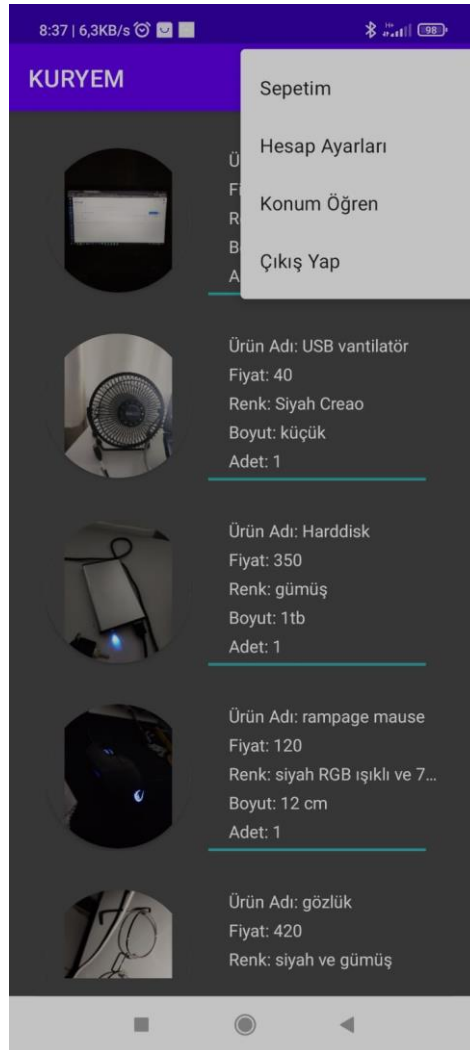
(Onay Maili Gönderme Kod Bloğu)

Kayıt işlemi Başarı ile tamamlanan kullanıcıya güvenlik için onay maili gönderme fonksiyonunu yazdık ve kayıt olma kısmında çağırdık, kayıt

olma işlemleri bittikten sonra ise işlem başarılı olduğu takdirde kullanıcıyı login sayfasına yönlendirdik ve kayıt olma işleminden sonra gerekli giriş işlemlerini yapmasını sağladık.

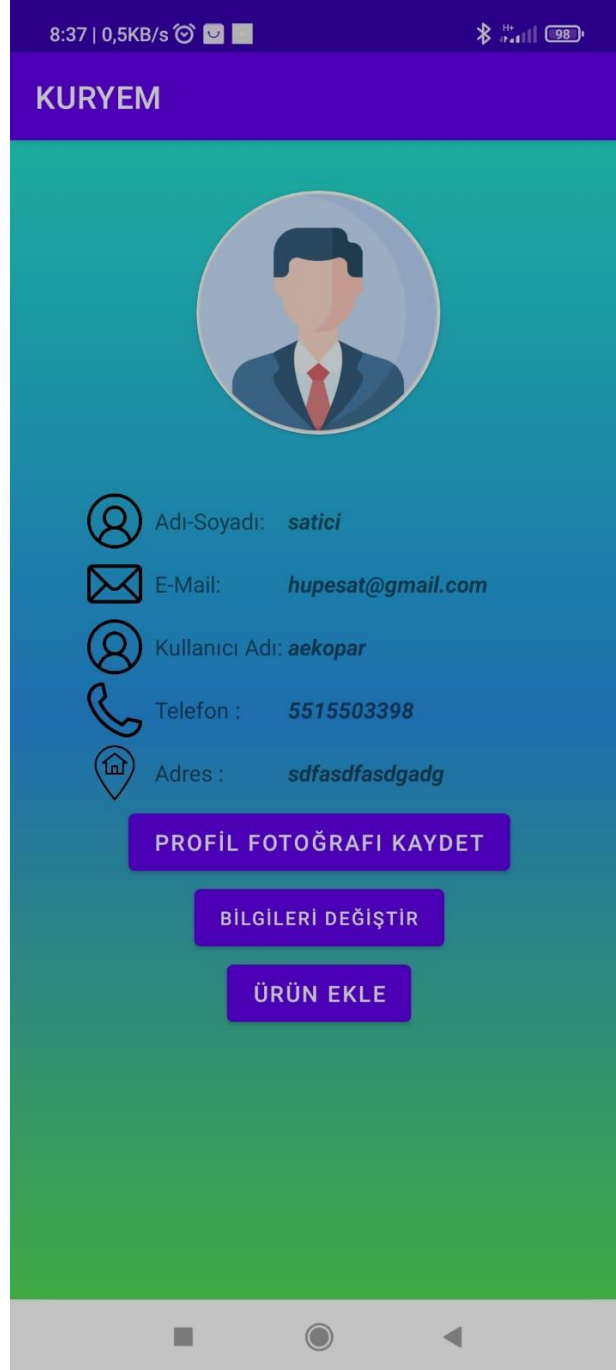
```
<item android:id="@+id/myBasket" android:title="Sepetim"></item>
<item android:id="@+id/hesapAyar" android:title="Hesap Ayarları"></item>
<item android:id="@+id/getLocation" android:title="Konum Öğren"></item>
<item android:id="@+id/menuCikisYap" android:title="Çıkış Yap"></item>
```

Şimdilik Main Activity kısmına herhangi bir ürün veri tabanımız olmadığı için sadece menü işlemlerini eklemek için gerekli menü tasarımını yaptık



(Menü Kısımının Görünümü)

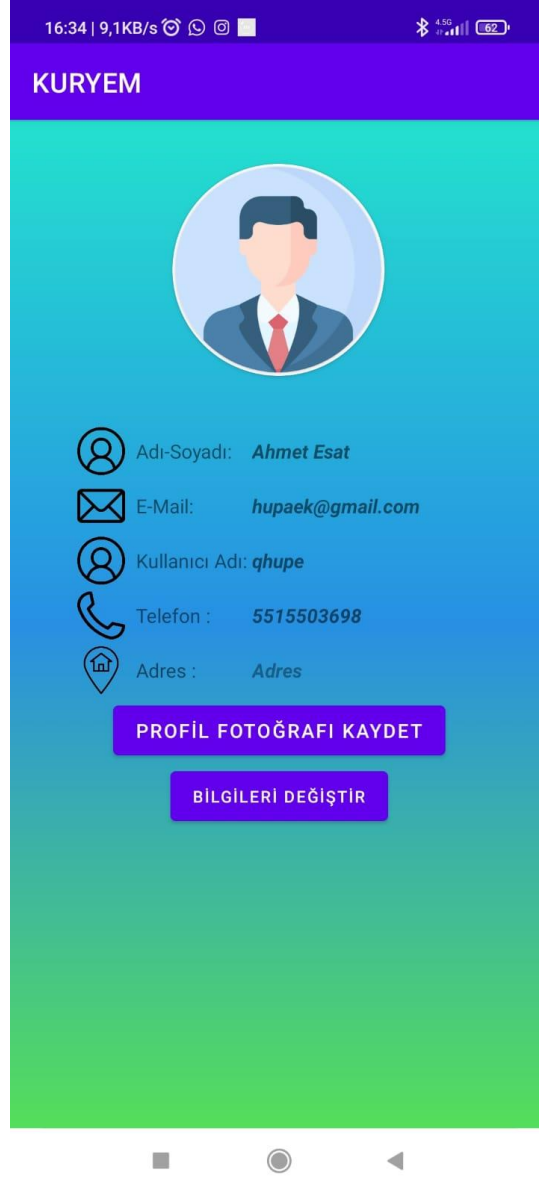
Adım 7: Menü kısmında kullanıcı ayarları kısmına tıklandığında geçilecek olan kullanıcı_ayarlari Activiyi kısmına geçiş işlemlerini tanımladım. Ve geçilen activity'nin tasarım kısmını yapmaya başladım



(Satıcı Olan Kullanıcının Kullanıcı Ayarları Ekranı)



(Kurye Üyeliği Olan Kullanıcının Kullanıcı Ayarları Ekranı)



(Normal Bir Müşteri Üyeliği Olan Kullanıcının Kullanıcı Ayarları Ekranı)

Adım 7:Kullanıcı Ayarları kısmında ise öncelikle kullanıcının bazı bilgilerini değiştirebilmesine izin verdik, ilk girildiği anda bu bilgiler textView'ler ile gösterilirken kullanıcı isteği doğrultusunda bilgileri değiştir butonuna tıklayarak bu bilgileri değiştirebileceği EditTextler ile karşılaşacaktır.

8:37 | 12,1KB/s | 98%

KURYEM

Adı-Soyadı: satıcı

E-Mail: hupesat@gmail.com

Kullanıcı Adı: aekopar

Telefon : 5515503398

Adres :

Şimdiki Şifre

Yeni Şifre

Yeni Şifre Tekrar

KAYDET

(Bilgileri değiştir butonuna basıldıktan sonraki ekran görüntüsü)

Bir diğer işlem ise kullanıcı burada profil resmi iconuna tıklayarak kendi profil resmini isteği doğrultusunda kameradan veya galeriden seçebilme imkanı sunuldu



```
userPicture.setOnClickListener {  
    if (permitStatus) {  
        var dialog = userPicPermissionFragment()  
        dialog.show(supportFragmentManager, "fotosec")  
    } else {  
        askPermission()  
    }  
}
```

(Profil Resmi İkonuna Tıkladığında açılacak dialog ekran ve çağıran kod bloğu)

Bu işlemten sonra ise Kullanıcın yükleyeceği fotoğraf boyutları veritabanı için fazla alan kaplamaması için seçilen veya çekilen fotoğrafı sıkıştırma işlemine tabi tuttuk

Adım 8:

```
inner class BackgroundPictureCompress : AsyncTask<Uri, Double, ByteArray?> {  
    var myBitmap: Bitmap? = null
```



```

constructor() {
}

constructor(bm: Bitmap) {

    if (bm != null) {
        myBitmap = bm
    }

}

override fun onPreExecute() {
    super.onPreExecute()
}

//Resmi Alıp Sıkıştırıp ByteArray e Çevirme Metodu
override fun doInBackground(vararg params: Uri?): ByteArray? {

    //Galeriden resim seçilme durumu
    if (myBitmap == null) {

        myBitmap = MediaStore.Images.Media.getBitmap(
            this@kullanici_ayarlari.contentResolver,
            params[0]
        )
        Log.e("TEST", "Resim Gerçek Boyut: " +
(myBitmap!!.byteCount).toDouble() / MBresult)

    }

    var picBytes: ByteArray? = null

    for (i in 1..3) {
        picBytes = convertBitmaptoByte(myBitmap, 100 / i)
        publishProgress(picBytes?.size?.toDouble())
    }

    return picBytes

}

override fun onProgressUpdate(vararg values: Double?) {
    super.onProgressUpdate(*values)
    Log.e("Sıkıştırma", "Sıkıştırılan Boyut:" + values[0]!! / MBresult + "
Mb")
    Toast.makeText(this@kullanici_ayarlari,"Fotoğraf
Yükleniyor",Toast.LENGTH_SHORT).show()
}

override fun onPostExecute(result: ByteArray?) {
    super.onPostExecute(result)
    uploadPicturetoFirebase(result)
}}

```

(Seçilen Fotoğrafi Sıkıştırma İşlemi Yapan Kod Bloğu)

Adım 9:

Sıkıştırma işlemi tamamlandıktan sonra ise seçilen fotoğrafı belirli kimlik bilgileri ile FireBase Veri Tabanımıza Kaydetme İşlemi Kaldı bu işlem için hazırladığımız fonksiyon ise şöyle

```
private fun uploadPicturetoFirebase(result: ByteArray?) {  
    var storageReferance =  
        FirebaseStorage.getInstance().reference.child("images/users/" +  
        FirebaseAuth.getInstance().currentUser?.uid + "/profil_resmi")  
    var resimEkle = storageReferance.putBytes(result!!)  
    resimEkle.continueWithTask(object : Continuation<UploadTask.TaskSnapshot,  
        Task<Uri>> {  
        override fun then(p0: Task<UploadTask.TaskSnapshot>): Task<Uri>? {  
            if (!p0.isSuccessful()) {  
                throw p0.getException()!!  
            }  
            return storageReferance.downloadUrl  
        }  
    }).addOnCompleteListener(object : OnCompleteListener<Uri> {  
        override fun onComplete(p0: Task<Uri>) {  
            if (p0.isSuccessful) {  
                Log.e("Firebasestorage", p0.result.toString())  
                val downloadUri = p0.result  
                FirebaseDatabase.getInstance().reference  
                    .child("kullanici")  
                    .child(FirebaseAuth.getInstance().currentUser!!.uid)  
                    .child("profil_resmi")  
                    .setValue(downloadUri!!.toString())  
  
                Toast.makeText(this@kullanici_ayarlari, "Fotoğraf Yüklendi: " +  
                    downloadUri.toString(), Toast.LENGTH_SHORT).show()  
  
            } else {  
                // Handle failures  
                // ...  
            }  
        }  
    }).addOnFailureListener(object : OnFailureListener{  
        override fun onFailure(p0: Exception) {  
            Toast.makeText(this@kullanici_ayarlari, "Resim yüklenirken hata  
oluştı", Toast.LENGTH_SHORT).show()  
        }  
    })  
}
```

(Seçilen ve Sıkıştırılan Resmi Veri Tabanına Kaydetme Fonksiyonu)

bu fonksiyonda ise resmi kaydederken kullanıcının uid'sine göre bir klasör açıp kaydettik ki kullanıcının kendi ayarlar sayfasında kendi profil fotoğrafını göresini sağlayabilelim, tüm bu işlemlerden bu

sayfada yapacağımız bir diğer şey ise kullanıcı bilgilerini çeki bu sayfada gerekli yerlere yansıtma işlemi

Adım 10:

```
private fun kullanıcıBilgileriGetir() {

    progressBarGoster(yukleme)
    progressBarGoster(Ppyukleme)

    var referans = FirebaseDatabase.getInstance().reference

    nowTimeEmail.setText(kullanici?.email.toString())

    //metot1
    var sorgu = referans.child("kullanici")
        .orderByKey()
        .equalTo(kullanici?.uid)

    sorgu.addListenerForSingleValueEvent(object : ValueEventListener {
        override fun onDataChange(p0: DataSnapshot) {

            for (singleSnapshot in p0.children) {

                var readuser = singleSnapshot.getValue(Kullanici::class.java)
                Log.e(
                    "Firebase",
                    "Adı:" + readuser?.isim + " Kullanıcı Adı:" +
readuser?.kullanici_adi
                    + " Telefon:" + readuser?.telefon + " Profil Resim:" +
readuser?.profil_resmi + " Seviye:" + readuser?.seviye
                )

                if (readuser?.seviye == "2") {
                    addProduct.visibility = View.VISIBLE
                } else if (readuser?.seviye == "3") {
                    addLocation.visibility = View.VISIBLE
                }
                else {
                    addProduct.visibility = View.GONE
                }

                nowTimeUserName.setText(readuser?.kullanici_adi)
                userNameText.setText(readuser?.kullanici_adi)
                nowTimeName.setText(readuser?.isim)
                userTelText.setText(readuser?.telefon)
                inputPhone.setText(readuser?.telefon)
                userAdres.setText(readuser?.adres)
            }
        }
    })
}
```

```

        progressBarGoster(Ppyukleme)

        if (readuser?.profil_resmi!!.isEmpty()) {

        } else {
Picasso.with(this@kullanici_ayarlari).load(readuser.profil_resmi).into(userPicture
)
        progressBarGizle(Ppyukleme)

        }
        progressBarGizle(yukleme)

    }

}

override fun onCancelled(p0: DatabaseError) {
    TODO("Not yet implemented")
}

})
}

```

(Kullanıcı Ayarları Sayfasında sisteme giriş yapmış olan mevcut kullanıcının bilgilerini almak için gerekli fonksiyon)

Adım 11:

Kullanıcı bilgilerini alıp arayüz kısmımızda kullanıcıya bu bilgileri gösterdikten sonra ise kullanıcıya bu sayfada kendi şifresini güncelleyebilmesi için bir fonksiyon yazdık

```

private fun sifreBilgisiGuncelle() {

    if (inputNewPassword.text.toString().isNotEmpty() &&
inputnewPasswordtry.text.toString()
        .isNotEmpty()
    ) {
        if (kullanici != null && inputnewPasswordtry.text.toString()
            .equals(inputNewPassword.text.toString())
        ) {
            kullanici?.updatePassword(inputNewPassword.text.toString())
            Toast.makeText(this, "Şifre Güncellendi, Tekrar Giriş Yapın",
Toast.LENGTH_SHORT)
                .show()
            FirebaseAuth.getInstance().signOut()
            finish()
        } else {
            Toast.makeText(this, "Yeni Şifreler Uyuşmuyor",
Toast.LENGTH_SHORT).show()
        }

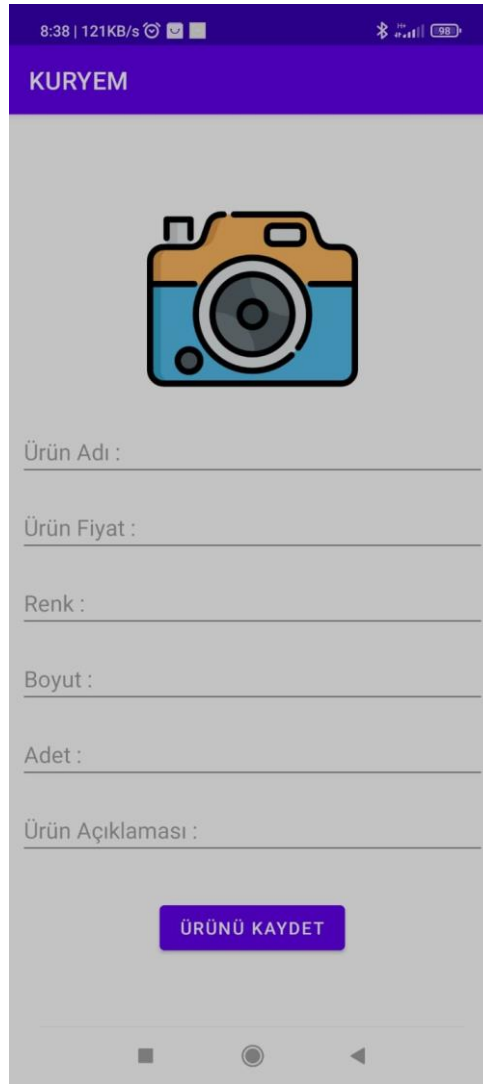
    } else {
        Toast.makeText(this, "Yeni Parola Alanları Boş Bırakılamaz",
Toast.LENGTH_SHORT).show()
    }
}

```

```
}  
}
```

Kullanıcı Bu fonksiyon sayesinde eski şifresini ve yeni şifre kısılarını doğru yazarak şifresini güncelleyebilme imkanına sahip olacaktır.

Adım 12:Kullanıcı Ayarları Sayfasındaki işlemlerimiz bittikten sonra öncelikle Müşteri Seviyesindeki kullanıcılar için lazım olan ürün ekleme sayfasının tasarımına başlayalım



(Ürün Ekleme Butonuna Basıldıktan Sonra Açılacak Olan Ürün Ekleme Sayfası)

Ürün ekleme sayfamızda ise eklenecek olan ürünün istenilen özellikleri girilmesi için EditTextler Kullandık bunun haricinde ise tıklanabilir bir ImageView oluşturarak satıcı olan kullanıcıya ürünün fotoğrafını galeriden seçmek mi istediğini veya kameradan çekmek mi istediğini bir DialogFragment ile sorduk bu profil resmi için kullandığımız dialogun aynısıydı



```
imageProduct.setOnClickListener {  
    if (permitStatus) {  
        var dialog = userPicPermissionFragment()  
        dialog.show(supportFragmentManager, "fotosec")  
    } else {  
        askPermission()  
    }  
}
```

(Resime tıklanıldığında ne yapılacağını oluşturan kod bloğu)

```

private fun askPermission() {

    var permissions = arrayOf(
        android.Manifest.permission.READ_EXTERNAL_STORAGE,
        android.Manifest.permission.WRITE_EXTERNAL_STORAGE,
        android.Manifest.permission.CAMERA
    )

    if (ContextCompat.checkSelfPermission(
        this,
        permissions[0]
    ) == PackageManager.PERMISSION_GRANTED &&
        ContextCompat.checkSelfPermission(
            this,
            permissions[1]
        ) == PackageManager.PERMISSION_GRANTED &&
        ContextCompat.checkSelfPermission(
            this,
            permissions[2]
        ) == PackageManager.PERMISSION_GRANTED
    ) {
        permitStatus = true
    } else {
        ActivityCompat.requestPermissions(this, permissions, 150)
    }
}

```

(Kullanıcının galerisine ve kamerasına erişim izinlerini sorma kod bloğu)

```

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == 150) {

        if (grantResults[0] == PackageManager.PERMISSION_GRANTED &&
            grantResults[1] == PackageManager.PERMISSION_GRANTED &&
            grantResults[2] == PackageManager.PERMISSION_GRANTED
        ) {
            var dialog = userPicPermissionFragment()
            dialog.show(supportFragmentManager, "fotosec")
        } else {
            Toast.makeText(
                this,
                "Gerekli İzinler Verilmeden İşlem Gerçekleştirilemez",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}

```

(İzinlerin verilip verilmediğini sorgulayan kod bloğu)

Bu yapılar dışında girilen bilgilerin ve resmin FireBase Veri Tabanına aktarılması için yazdığımız kodlar ise şöyle...

```
saveProduct.setOnClickListener {  
    if  
(productName.text.toString()!=""&&productColor.text.toString()!=""&&productPieces.  
text.toString()!=""&&productPrice.text.toString()!=""&&productSize.text.toString()  
!=""&&productInformation.text.toString()!=""&&comeFromGalerryURI!=null) {  
        if (comeFromGalerryURI != null) {  
            pictureCompressed(comeFromGalerryURI!!)  
        } else if (comeFromCameraBitmap != null) {  
            pictureCompressed(comeFromCameraBitmap!!)  
        }  
        yeniUrunKayit()  
        productName.setText("")  
        productColor.setText("")  
        productPieces.setText("")  
        productPrice.setText("")  
        productSize.setText("")  
        productInformation.setText("")  
        imageProduct.invalidate();  
        imageProduct.setImageBitmap(null);  
    }else{  
        Toast.makeText(this,"Lütfen Tüm Alanları  
Doldurun",Toast.LENGTH_SHORT).show()  
    }  
}
```

(Kaydet Butonu Kod Yapısı)

```
private fun uploadPicturetoFirebase(result: ByteArray?) {  
    var storageReferance =  
FirebaseStorage.getInstance().reference.child("/productimages/"+kullanici?.uid+"/p  
roducts/"+productId)  
    var resimEkle = storageReferance.putBytes(result!!)  
    resimEkle.continueWithTask(object : Continuation<UploadTask.TaskSnapshot,  
Task<Uri>> {  
        override fun then(p0: Task<UploadTask.TaskSnapshot>): Task<Uri>? {  
            if (!p0.isSuccessful()) {  
                throw p0.getException()!!  
            }  
            return storageReferance.downloadUrl  
                println("ÜRÜN LİNKİ -----  
>"+storageReferance.downloadUrl)  
        }  
    })
```



```

}).addOnCompleteListener(object : OnCompleteListener<Uri> {
    override fun onComplete(p0: Task<Uri>) {
        if (p0.isSuccessful) {
            Log.e("Firebasestorage", p0.result.toString())
            val downloadUri = p0.result
            FirebaseDatabase.getInstance().reference
                .child("product")
                .child(productId.toString())
                .child("ürün_resmi")
                .setValue(downloadUri!!.toString())

            Toast.makeText(
                this@add_Product,
                "Fotoğraf Yüklendi: " + downloadUri.toString(),
                Toast.LENGTH_SHORT
            ).show()

        } else {
            // Handle failures
            // ...
        }
    }
}).addOnFailureListener(object : OnFailureListener {
    override fun onFailure(p0: Exception) {
        Toast.makeText(
            this@add_Product,
            "Resim yüklenirken hata oluştu",
            Toast.LENGTH_SHORT
        ).show()
    }
})
}
}

```

(Çekilen veya alınan resmi FireBase veri tabanına sıkıştırıp yükleyen kod bloğu)

Ürüne ait olan resimleri veri tabanına yüklerken profil fotoğrafını yüklediğimiz işlemin aynısını uyguladık.

1-)Resmi Aldık

2-)Sıkıştırdık

3-)Veri Tabanına Kayıt İşlemi Yaptık

```

private fun yeniUrunKayit() {

    var addDatabaseUser = Ürünler()
    addDatabaseUser.product_name = productName.text.toString()
    addDatabaseUser.product_price = productPrice.text.toString()
    addDatabaseUser.product_color = productColor.text.toString()
    addDatabaseUser.kullanici_id = FirebaseAuth.getInstance().currentUser?.uid
    addDatabaseUser.ürün_resmi = ""
    addDatabaseUser.product_size = productSize.text.toString()
    addDatabaseUser.product_id=productId
    addDatabaseUser.ürün_information=productInformation.text.toString()
    addDatabaseUser.product_pieces = productPieces.text.toString()

    FirebaseDatabase.getInstance().reference
        .child("product")
        .child(productId.toString())
        .setValue(addDatabaseUser).addOnCompleteListener { task ->
            if (task.isSuccessful) {
                Toast.makeText(
                    this,
                    "Üye Kaydedildi ",
                    Toast.LENGTH_SHORT
                )
                    .show()
            } else {
                Toast.makeText(
                    this,
                    "Üye Kaydedilirken Bir Sorun Oluştı ",
                    Toast.LENGTH_SHORT
                ).show()
            }
        }
    // Create a new user with a first and last name
}

```

(Yeni Ürünün bilgilerinin Veri Tabanına Kaydedilmesi İşlemi)

Adım 13:Ürün kayıt işlemlerimiz bittikten sonra ise ürün istesi sayfamızda yer alan recycler viewe tıklandıktan sonra gidilecek olan ürün özellikleri sayfamızın tasarımına başladık. Bu ekranda ise kullanıcıdan aldığımız ürünün bilgileri doğrultusunda veri tabanına kaydettiğimiz ürünün özelliklerini ve resmini çağırmak kaldı, ürünün resmini kaydettiğimiz konumun url kısmını ürünün veritabanına kaydettiğimiz için resminide url olarak çağırmış olduk.



(Ürün Özellikleri Sayfası)

```
if(intent.hasExtra("product_id")){  
    checkProductId = intent.getStringExtra("product_id").toString()  
}  
  
getProductInformation()
```

Ürün Özelliklerini Çağırmadan önce Ürün Listesi Sayfasından gönderdiğimiz product_id adlı veriyi ürün özellikleri sayfasında getStringExtra metodu ile kullanmaya başlıyoruz ki veri tabanında kullanıcının tıkladığı ürüne ulaşalım, sonrasında ise bu id ile tıklanılan ürünün özelliklerini veri tabanından çekip gerekli yerlere yazdırıyoruz.

```
private fun getProductInformation() {  
  
    var referans = FirebaseDatabase.getInstance().reference  
  
    //metot1  
    var sorgu = referans.child("product")
```

```

        .orderByKey()
        .equalTo(checkProductId)

sorgu.addListenerForSingleValueEvent(object : ValueEventListener {
    override fun onDataChange(p0: DataSnapshot) {

        for (singleSnapshot in p0.children) {

            var readuser = singleSnapshot.getValue(Ürünler::class.java)

            productNameDetail.setText("Ürün Adı : "+readuser?.product_name)
            productPriceDetail.setText("Ürün Fiyatı :
"+readuser?.product_price)
            productPiecesDetail.setText("Adet : "+readuser?.product_pieces)
            ProductColorsDetail.setText("Ürün Özelli :
"+readuser?.product_color)
            productSizeDetail.setText("Ürün Boyut/Numara :
"+readuser?.product_size)
            Productexplanation.setText("Ürün Açıklaması :
"+readuser?.ürün_information)
            productlink=readuser?.product_id.toString()
            imageLink= readuser?.ürün_resmi!!

            if (readuser?.ürün_resmi!!.isEmpty()) {

            } else {

Picasso.with(this@productDetails).load(readuser.ürün_resmi).into(productDetailsPic
)

            }

        }

        override fun onCancelled(p0: DatabaseError) {
            TODO("Not yet implemented")
        }
    })

    var
productId=ref.child("kullanici").child(FirebaseAuth.getInstance().currentUser?.uid
!!).child("basket").push().key

```

(Ürün Özellikleri sayfası kod bloğu)

Veri Tabanından verileri çekerken işimizi kolaylaştırmak adına ve daha stabil bilgiler elde etmek için Ürünler sınıfımızdan yeni bir nesne oluşturup bilgileri bu nesneye aktarıp o nesne üzerinden işlem

yapıyoruz. Ve ürün özellikleri sayfasında sepete ekleme butonu ile kullanıcıya ürünü sepete ekleme imkanını sunmuş oluyoruz.

```
addBasket.setOnClickListener {
    FirebaseDatabase.getInstance().reference
        .child("kullanici")
        .child(FirebaseAuth.getInstance().currentUser?.uid!!)
        .child("basket")
        .child(productlink.toString())
        .setValue(imageLink)

    Toast.makeText(this, "Ürün Sepete Eklendi", Toast.LENGTH_SHORT).show()
}
```

(Sepete Ekle butonu Kod bloğu)

Sepete ekleme butonunda ise kullanıcının kendi veri tabanında basket isimli bir çocuk oluşturarak sepete eklediği ürünlerin id lerini buraya ekledik, bu sepet kısmında sepetteki ürünleri göstermek için bize kolaylık sağlamış olacak. Bu işlemleri görmek için ise sepetim kısmını sağ üstteki menü kısmına ekleyerek o kısma tıklandığında açılacak olan sepetim sayfasını tasarlamaya geçiyoruz.

Adım 14: Sepetim sayfasını tasarlarken Ürünler sayfasına çok benzer hatta nerdeyse aynı diyeceğimiz bir yapı kullandık, bu kısımda yine birden fazla ve artıp azalabilecek sayıda ürün olabileceği için bir RecyclerView yapısı kullanmayı tercih ettim, bunu Adapterle sayfamızın activity kısmına bağlayarak kullanıcının sepetindeki ürünleri listeledim.

```

package com.example.firebasekotlin

import android.annotation.SuppressLint
import android.app.Activity
import android.content.DialogInterface
import android.content.Intent
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.LinearLayout
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.squareup.picasso.Picasso
import kotlinx.android.synthetic.main.activity_kullanici_ayarlari.*
import kotlinx.android.synthetic.main.image_with_text.view.*

class BasketViewAdapter(mActivity: AppCompatActivity, allProduct:
ArrayList<Ürünler>) : RecyclerView.Adapter<BasketViewAdapter.BasketHolder>() {

    var idRef=""
    var products=allProduct
    var kullanici=FirebaseAuth.getInstance().currentUser

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
BasketHolder {
        var inflater=LayoutInflater.from(parent?.context)
        var productBox = inflater.inflate(R.layout.image_with_text, parent, false)

        return BasketHolder(productBox)
    }

    override fun getItemCount(): Int {
        return products.size
    }

    override fun onBindViewHolder(holder: BasketHolder, position: Int) {

        var NowCreateBasket = products.get(position)
        holder?.setData(NowCreateBasket, position)
    }

    inner class BasketHolder(itemView: View?) :
RecyclerView.ViewHolder(itemView!!) {

```

```

var productActivity=itemView as LinearLayout
var productImageAdp = productActivity.product_image
var productColorAdp=productActivity.product_color
var productPriceAdp=productActivity.product_price
var productPiecesAdp=productActivity.product_pieces
var productSizeAdp=productActivity.product_size
var ProductNameAdp = productActivity.product_Name

@SuppressLint("RestrictedApi")
fun setData(nowCeateProduct1: Ürünler, position: Int) {

    productActivity.deleteBasket.visibility=View.VISIBLE

    var ref=FirebaseDatabase.getInstance().reference

    var
sorgu1=ref.child("kullanici").child(FirebaseAuth.getInstance()?.uid.toString()).ch
ild(
    "basket"
).addListenerForSingleValueEvent(
    object : ValueEventListener {
        override fun onDataChange(p0: DataSnapshot) {

            // for (allProductBox in p0?.children) {

            // idRef =allProductBox.key.toString()

            // }

            // if (idRef==nowCeateProduct1.product_id) {

                println("BASKETADAPTERRRRR ID-----" + idRef)
                println("BASKETADAPTERRRRR Product ID-----" +
nowCeateProduct1.product_id)
                productColorAdp.text =
                    "Renk: " + (nowCeateProduct1.product_color).toString()
                ProductNameAdp.text =
                    "Ürün Adı: " +
(nowCeateProduct1.product_name).toString()
                productSizeAdp.text = "Boyut: " +
(nowCeateProduct1.product_size).toString()
                productPiecesAdp.text =
                    "Adet: " +
(nowCeateProduct1.product_pieces).toString()
                productPriceAdp.text =
                    "Fiyat: " +
(nowCeateProduct1.product_price).toString()
            }
        }
    }
)
}

```


kullanıcının veri tabanının alt çocuğu olan basket kısmında gezinerek sepetine eklediği ürünlerin id'lerini elde etmiş oluyoruz ve item sayısını buna göre belirliyoruz, sonrasında ise sepet ekranının kodlarına geçerek sepetteki ürünleri listelemeye başlıyoruz.

```
private fun getAllProductFromActivity() {

    getAllProduct=ArrayList<Ürünler>()
    var ref=FirebaseDatabase.getInstance().reference

    var
    sorgu=ref.child("kullanici").child(kullanici?.uid.toString()).child("basket").addListenerForSingleValueEvent(object:ValueEventListener{
        override fun onDataChange(p0: DataSnapshot) {

            for (allProductBox in p0?.children){

                var idRef=allProductBox.key.toString()

                var
                sorgu=ref.child("product").addListenerForSingleValueEvent(object:ValueEventListener{
                    override fun onDataChange(p0: DataSnapshot) {

                        for (allProductBoxref in p0?.children){

                            var nowProduct = Ürünler()

                            var ObjectMap=(allProductBoxref.getValue() as
HashMap<String,Object>)

                            nowProduct.product_id=ObjectMap.get("product_id").toString()
                            nowProduct.product_name=ObjectMap.get("product_name").toString()
                            nowProduct.product_price=ObjectMap.get("product_price").toString()
                            nowProduct.product_color=ObjectMap.get("product_color").toString()
                            nowProduct.product_size=ObjectMap.get("product_size").toString()
                            nowProduct.product_pieces=ObjectMap.get("product_pieces").toString()
                            nowProduct.ürün_resmi=ObjectMap.get("ürün_resmi").toString()
                            nowProduct.kullanici_id=ObjectMap.get("kullanici_id").toString()

                            Log.e("TEST",nowProduct.product_id+"
"+nowProduct.ürün_resmi)

                            if (idRef==nowProduct.product_id) {
                                getAllProduct.add(nowProduct)
                                myAdapter?.notifyDataSetChanged()
                            }
                        }
                    }
                })
            }
        }
    })
}
```

```

        }
    }
    if(myAdapter==null){
        spetilListele()
    }

    }
    override fun onCancelled(error: DatabaseError) {
        TODO("Not yet implemented")
    }

    })
}
if(myAdapter==null){
    spetilListele()
}

}

override fun onCancelled(error: DatabaseError) {
    TODO("Not yet implemented")
}

})
}

```

(Sepetim Sayfası Ürün Listeleme Kod Kısmı)

Ve burada ise tekrardan kullanıcının veri tabanında gezip id'leri alıp sonradan ürünler veri tabanına erişip bu id'ler ile sorgu yaparak alınan id'lere ait bilgileri kullanıcının sepet sayfasına yansıtıyoruz, yani özet olarak:

Kullanıcının kendi sepetine eklediği ürünlerin id kısmını yine kullanıcıya ait olan veri tabanında basket adında bir çocuk veritabanı oluşturularak alınan id leri buraya kaydediyoruz, sonrasında ise bu idleri sepet sayfasına gelindiğinde tekrar çekip ürünler veri tabanında bu idlerin özelliklerini kullanıcının sepet sayfasında kendisine yasıtıyoruz.

```

productActivity.deleteBasket.setOnClickListener {

    val builder = AlertDialog.Builder(itemView.context)
    builder.setTitle("Emin misiniz?")
    builder.setMessage("Ürünü Sepetinizden Silmek İstiyor Musunuz?")
    builder.setPositiveButton("Evet") { dialogInterface: DialogInterface, i: Int -
>

        FirebaseDatabase.getInstance().reference.child("kullanici").child(
            kullanici!!.uid
        ).child("basket").child(nowCeateProduct1.product_id.toString())
            .removeValue()
    }
}

```

```

var intent = Intent(itemView.context, BasketPage::class.java)
(itemView.context as Activity).finish()
itemView.context.startActivity(intent)

}
builder.setNegativeButton("Hayır") { dialogInterface: DialogInterface, i: Int
->

}
builder.show()
}

```

(Ürünleri Sepetten Silme İşlemi)

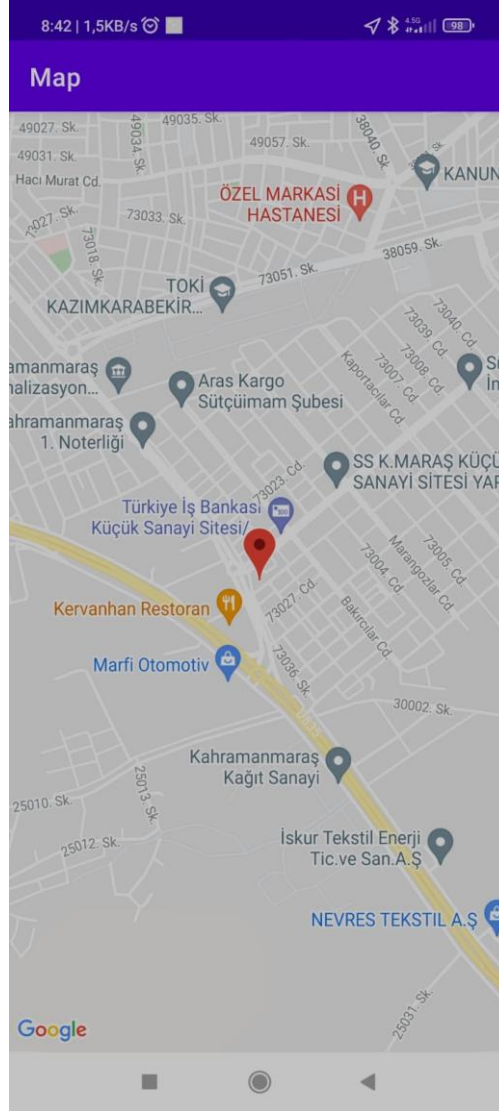
Burada ise sepetim sayfasında üzerine tıklanan ürüne bir alert dialog göndererek bu ürünü sepetten silip isteyip istemediğini soruyoruz, eğer cevap evet ise kullanıcı veritabanından bu ürünün idsini silerek gösterimden alıyoruz.

Bu işlem ile birlikte kullanıcı ve satıcı işlemlerimiz bitiyor ve son olarak kurye işlemlerimize devam ediyoruz...



(Kurye Kullanıcı Ayarları Kısmı “Konum Bildir Butonu”)

Sistemimize Kayıtlı olan kurye seviyesindeki kullanıcılar ise diğerlerinden farklı olarak konum bildirme yetkisine sahip olacaktır, Google maps api ile elde ettiğimiz bu işlemi kuryelerin anlık konum bildirmesi ve son mevcut konumu bildirmesi gibi işlemlerde bize yardımcı olup ulaşılabilirliği arttıracaktır.



(Konum Bildir Kismına Tıklanınca Açılan Ekran)

```
addLocation.setOnClickListener {  
    var intent=Intent(this,MapsActivity::class.java)  
    startActivity(intent)  
}
```

(Konum Bildir Butonu Kod Bloğu)

Butona tıklandıktan sonra geçilecek olan MapActivity Sayfasında öncelikle konum izinlerinin alınıp alınmadığını sorgulamamız gerekecektir.

```
if
(ContextCompat.checkSelfPermission(this,Manifest.permission.ACCESS_FINE_LOCATION)!
= PackageManager.PERMISSION_GRANTED){
    //izin verilmemiş
    ActivityCompat.requestPermissions(this,
    arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),1)
}else{
    //izin zaten verilmiş

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,1,1f,locationL
istener)
}
```

(Konum izinlerinin alınıp alınmadığı sorgusu)

Bu sorguyu yaptıktan sonra ise alınan konumu firebase veri tabanına herkesin ulaşabileceği bir şekilde eklemek için sisteme giriş yapmış olan mevcut kuryenin veri tabanından kullanıcı adını çekip lokasyon veri tabanında kuryenin kullanıcı adı ile bir çocuk oluşturup kuryenin konumunu oraya kaydedeceğiz.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_maps)
    // Obtain the SupportMapFragment and get notified when the map is ready to be
    used.
    val mapFragment = supportFragmentManager
        .findFragmentById(R.id.map) as SupportMapFragment
    mapFragment.getMapAsync(this)

    var referans1 = FirebaseDatabase.getInstance().reference
    var sorgu = referans1.child("kullanici")
        .orderByKey()
        .equalTo(kullanici?.uid)

    sorgu.addListenerForSingleValueEvent(object : ValueEventListener {
        override fun onDataChange(p0: DataSnapshot) {

            for (singleSnapshot in p0.children) {

                var readuser = singleSnapshot.getValue(Kullanici::class.java)
                kullaniciAdi=readuser?.kullanici_adi.toString()
            }
        }
    })
}
```

```

    }

    }
    override fun onCancelled(p0: DatabaseError) {
        TODO("Not yet implemented")
    }
    })
}

```

(Giriş Yapmış olan mevcut kuryenin kullanıcı adını aldık)

```

override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap

    locationManager = getSystemService(Context.LOCATION_SERVICE) as
    LocationManager

    locationListener = object : LocationListener{
        override fun onLocationChanged(location: Location) {
            //lokasyon veya konum değişince yapılacak işlemler

            mMap.clear()
            val guncelKonum = LatLng(location.latitude,location.longitude)

            FirebaseDatabase.getInstance().reference
                .child("location")
                .child(kullaniciAdi)
                .child("latitude")
                .setValue(location.latitude)

            FirebaseDatabase.getInstance().reference
                .child("location")
                .child(kullaniciAdi)
                .child("longitude")
                .setValue(location.longitude)

            mMap.addMarker(MarkerOptions().position(guncelKonum).title("Buradasınız"))
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(guncelKonum,15f))

        }
    }
}

```

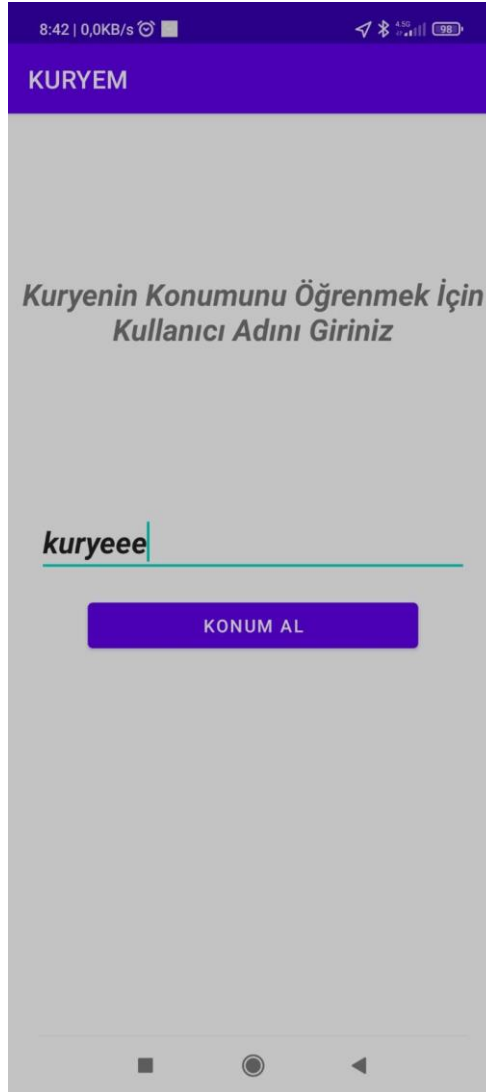
(Konum Alma ve Kaydetme işlemleri kod bloğu)

Konumu alıp kuryenin kullanıcı adı ile location veri tabanına kuryenin kullanıcı adı altında latitude ve longitude kısımlarına enlem ve boylam bilgilerini kayıt ediyoruz bu kısım location listener ile yapıldığı için

harita açık kaldığı sürece anlık konum takibi yapabileceğiz. Peki Müşteri bu konumu nasıl görecektir?

```
R.id.getLocation -> {  
    var intent=Intent(this,getLocationActivity::class.java)  
    startActivity(intent)  
}
```

Ana menümüzde bulunan sağ üst kısımdaki menüde konum öğren kısmının işlemleri bizi getLocationActivityye yönlendirecek



(Konum Öğren Sayfası Tasarımı)

Bu sayfada konumunu öğrenmek istediğimiz kuryenin kullanıcı adını yazarak o kullanıcıya ait son konum bilgisini alabileceğiz. Peki nasıl?

```

override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap
    locationManager = getSystemService(Context.LOCATION_SERVICE) as
LocationManager
    println("-----Aktivitiden Gelen Veri----->" + kuryeUserName)

    locationListener = object : LocationListener {
        override fun onLocationChanged(location: Location) {
            //lokasyon veya konum değişince yapılacak işlemler
            println("----->Locationn"+location.latitude.toString())
            println(location.longitude.toString())

            mMap.clear()

            var referans = FirebaseDatabase.getInstance().reference

            var sorgu = referans.child("location")
                .orderByKey()
                .equalTo(kuryeUserName)

            sorgu.addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(p0: DataSnapshot) {

                    for (singleSnapshot in p0.children) {

                        var readuser =
singleSnapshot.getValue(Locations::class.java)

                        longituderef= readuser?.longitude!!
                        latituderef=readuser?.latitude!!
                        guncelKonum= LatLng(latituderef,longituderef)

mMap.addMarker(MarkerOptions().position(guncelKonum).title(kuryeUserName+"
Konumu"))

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(guncelKonum,15f))
                    }
                }
                override fun onCancelled(p0: DatabaseError) {
                    TODO("Not yet implemented")
                }
            })
            // Add a marker in Sydney and move the camera
        }
    }

    if
(ContextCompat.checkSelfPermission(this,Manifest.permission.ACCESS_FINE_LOCATION)!
= PackageManager.PERMISSION_GRANTED){
        //izin verilmemiş
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),1)

    }else{
        //izin zaten verilmiş
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,1,1f,locationL
istener)
    }
}
}
}

```

(Kurye Konumu öğreneceğimiz Kod Bloğu)

Burada konum bildir sayfası ile benzer bir yapı kullanıldı fakat fark olarak konum bilgisini haritadan değilde kuryemizin kullanıcı adına ait olan location üyesi veritabanından çektik, Firebase veri tabanında oluşturduğumuz location kısmının çocuğu olan konumu öğrenilmek istenen kuryenin kullanıcı adının altında konum bilgileri saklıyorduk, bu sayfada ise bu konum bilgilerini çekip haritada kullanıcıya sunmuş olduk.

Projemizi Konum bildirme kısmı ile Bitirmiş olup Eksik kısımları günden güne geliştirmeye devam edeceğim bir proje haline geldi, umarım anlatabilmişimdir.

Teşekkür Ederim

Ahmet Esat Kopar