

Jetpack Compose ile Android Uygulama Geliştirme Kursu

Depolama İşlemleri

Kasım ADALAN

Elektronik ve Haberleşme Mühendisi

Android - IOS Developer and Trainer

Eğitim İçeriği

- Preferences DataStore
- Room

Preferences DataStore

Preferences DataStore

- Key - Value ilişkisi ile basit verileri kalıcı olarak depolayabiliriz.
- Kullanılan bazı veri türleri : *String,int,double,bool,Set<String>*
- Uygulama silindiğinde veriler silinmektedir.
- Veritabanı üzerindeki gibi detaylı verileri kayıt etmiceksek hızlı bir çözüm için kullanılabilir.

Kurulum

```
dependencies {  
    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.compose.ui:ui:$compose_version"  
    implementation "androidx.compose.material:material:$compose_version"  
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"  
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'  
    implementation 'androidx.activity:activity-compose:1.3.1'  
    testImplementation 'junit:junit:4.+'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"  
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_version"  
    //Preferences DataStore  
    implementation("androidx.datastore:datastore-preferences:1.0.0")  
    // Lifecycle component & Kotlin coroutines components  
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.2.0"  
    api "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.9"  
    api "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9"  
    //ViewModel & Live data  
    implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"  
    implementation "androidx.compose.runtime:runtime-livedata:1.0.0-beta08"  
}
```

```
//Preferences DataStore  
implementation("androidx.datastore:datastore-preferences:1.0.0")  
// Lifecycle component & Kotlin coroutines components  
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.2.0"  
api "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.9"  
api "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9"  
//ViewModel & Live data  
implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"  
implementation "androidx.compose.runtime:runtime-livedata:1.0.0-beta08"
```

Preferences DataStore Sınıfı Oluşturma

*Preferences
DataStore'a erişim
için ds isimli değişken
oluşturuldu.*

```
class AppPref(var context:Context) {  
    val Context.ds : DataStore<Preferences> by preferencesDataStore( name: "bilgiler")  
  
    companion object {  
        val AD_KEY = stringPreferencesKey( name: "AD")  
    }  
  
    suspend fun kayitAd(ad:String){  
        context.ds.edit { it: MutablePreferences  
            it[AD_KEY] = ad  
        }  
    }  
  
    suspend fun okuAd():String {  
        val p = context.ds.data.first()  
        return p[AD_KEY] ?: "isim yok"  
    }  
  
    suspend fun silAd(){  
        context.ds.edit { it: MutablePreferences  
            it.remove(AD_KEY)  
        }  
    }  
}
```

*Kayıt edilecek
dosya adı*

*Kayıt için
kullanılacak KEY
tanımlaması*

Kayıt İşlemi

Okuma İşlemi

*Okuma işlemi yapılırken varsayılan
değer oluşturulabilir.*

Silme İşlemi

Kullanımı

```
@Composable
fun Sayfa() {
    val context = LocalContext.current
    val ap = AppPref(context)

    LaunchedEffect(key1: true){ this: CoroutineScope
        val job: Job = CoroutineScope(Dispatchers.Main).launch {
            ap.kayitAd( ad: "Ahmet")

            ap.silAd()

            val gelenAd = ap.okuAd()
            Log.e( tag: "Gelen Ad",gelenAd)
        }
    }
}
```

Final Kod :

```
class AppPref(var context:Context) {
    val Context.ds : DataStore<Preferences> by preferencesDataStore( name: "bilgiler")

    companion object {
        val AD_KEY = stringPreferencesKey( name: "AD")
        val YAS_KEY = intPreferencesKey( name: "YAS")
        val BOY_KEY = doublePreferencesKey( name: "BOY")
        val BEKAR_MI_KEY = booleanPreferencesKey( name: "BEKAR_MI")
        val ARKADAS_LISTE_KEY = stringSetPreferencesKey( name: "ARKADAS_LISTE_KEY")
    }

    suspend fun kayitAd(ad:String){
        context.ds.edit { it: MutablePreferences
            it[AD_KEY] = ad
        }
    }

    suspend fun okuAd():String {
        val p = context.ds.data.first()
        return p[AD_KEY] ?: "isim yok"
    }

    suspend fun silAd(){
        context.ds.edit { it: MutablePreferences
            it.remove(AD_KEY)
        }
    }

    suspend fun kayitYas(yas:Int){
        context.ds.edit { it: MutablePreferences
            it[YAS_KEY] = yas
        }
    }

    suspend fun okuYas():Int {
        val p = context.ds.data.first()
        return p[YAS_KEY] ?: 0
    }

    suspend fun kayitBoy(boy:Double){
        context.ds.edit { it: MutablePreferences
            it[BOY_KEY] = boy
        }
    }

    suspend fun okuBoy():Double {
        val p = context.ds.data.first()
        return p[BOY_KEY] ?: 0.0
    }

    suspend fun kayitBekarMi(bekarMi:Boolean){
        context.ds.edit { it: MutablePreferences
            it[BEKAR_MI_KEY] = bekarMi
        }
    }

    suspend fun okuBekarMi():Boolean {
        val p = context.ds.data.first()
        return p[BEKAR_MI_KEY] ?: false
    }

    suspend fun kayitArkadasListe(liste:Set<String>){
        context.ds.edit { it: MutablePreferences
            it[ARKADAS_LISTE_KEY] = liste
        }
    }

    suspend fun okuArkadasListe():Set<String>? {
        val p = context.ds.data.first()
        return p[ARKADAS_LISTE_KEY]
    }
}
```


@Composable

fun Sayfa() {

val context = LocalContext.current

val ap = AppPref(context)

LaunchedEffect(key1: true){ this: CoroutineScope

val job: Job = CoroutineScope(Dispatchers.Main).launch {

//Kayıt İşlemleri

ap.kayitAd(ad: "Ahmet")

ap.kayitYas(yas: 23)

ap.kayitBoy(boy: 1.78)

ap.kayitBekarMi(bekarMi: true)

val liste = HashSet<String>()

liste.add("Mehmet")

liste.add("Zeynep")

ap.kayitArkadasListe(liste)

//Okuma İşlemleri

val gelenAd = ap.okuAd()

val gelenYas = ap.okuYas()

val gelenBoy = ap.okuBoy()

val gelenBekarMi = ap.okuBekarMi()

Log.e(tag: "Gelen Ad",gelenAd)

Log.e(tag: "Gelen Yaş",gelenYas.toString())

Log.e(tag: "Gelen Boy",gelenBoy.toString())

Log.e(tag: "Gelen Bekar Mi",gelenBekarMi.toString())

val gelenListe = ap.okuArkadasListe()

for(a in gelenListe!!){

Log.e(tag: "Gelen Arkadaş",a)

}

}

}

}

Uygulama : Açılış Sayısı



```
class AppPref(var context: Context) {  
    val Context.ds : DataStore<Preferences> by preferencesDataStore( name: "bilgiler")  
  
    companion object {  
        val SAYAC_KEY = intPreferencesKey( name: "SAYAC")  
    }  
  
    suspend fun kayitSayac(sayac:Int){  
        context.ds.edit { it: MutablePreferences  
            it[SAYAC_KEY] = sayac  
        }  
    }  
  
    suspend fun okuSayac():Int {  
        val p = context.ds.data.first()  
        return p[SAYAC_KEY] ?: 0  
    }  
}
```

Uygulama : Açılış Sayısı



```
@Composable
fun Sayfa() {
    val context = LocalContext.current
    val ap = AppPref(context)
    val sayac = remember { mutableStateOf( value: 0) }

    LaunchedEffect( key1: true){ this: CoroutineScope
        val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope
            var gelenSayac = ap.okuSayac()
            sayac.value = ++gelenSayac
            ap.kayitSayac(gelenSayac)
        }
    }

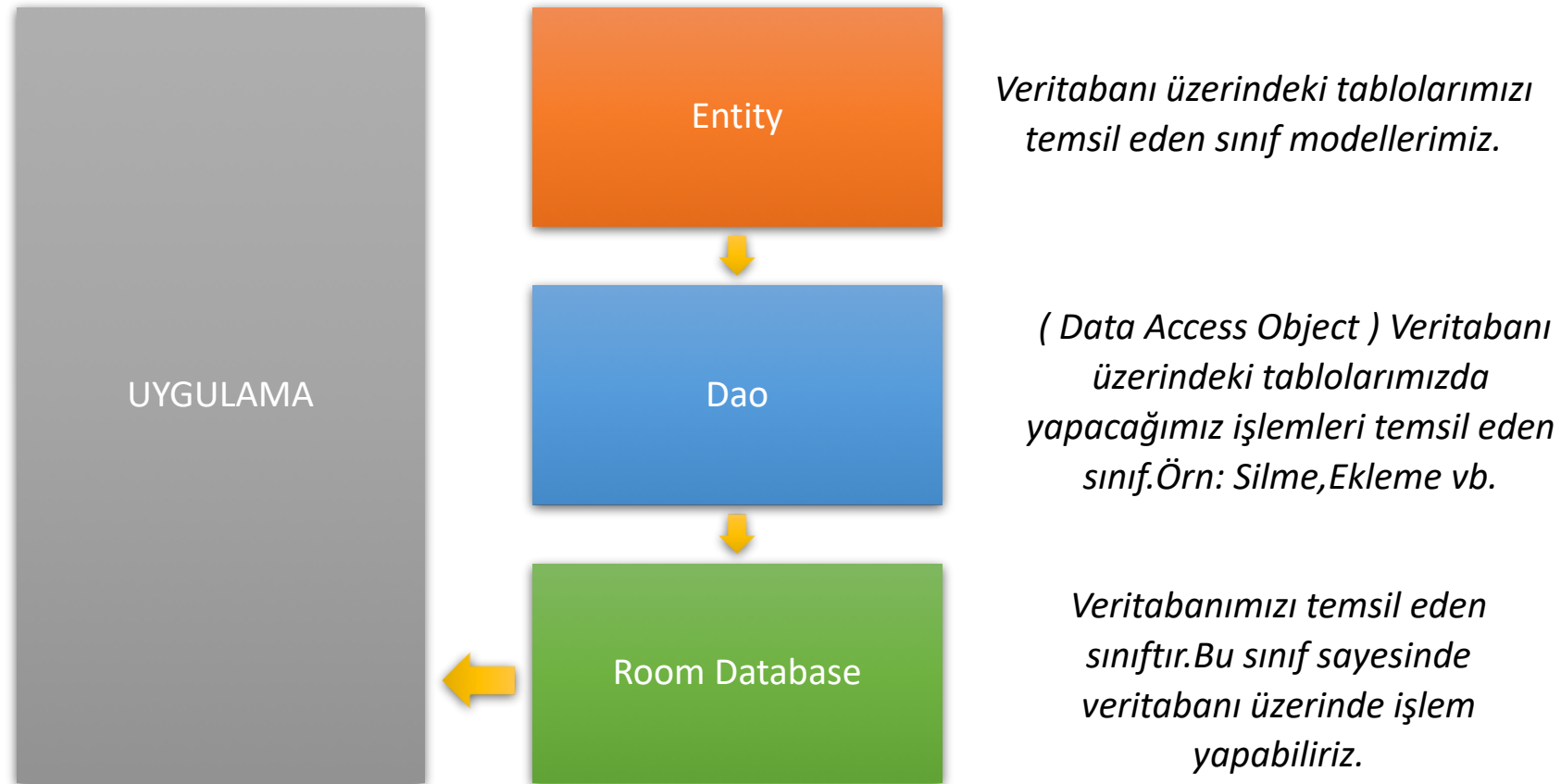
    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text( text: "Açılış Sayısı : ${sayac.value}",fontSize = 36.sp)
    }
}
```

Room

Room Kütüphanesi

- SQLite veritabanı üzerinde çalışan bir kütüphanedir.
- Androdin kendi kütüphanesidir.
- Nesne tabanlı şekilde çalışmaktadır.
- SQLite kullanımını oldukça kolaylaştıran bir yapıdır.
- Room kütüphanesini Asenkron çalışması için Coroutine yapısıyla kullanmaktayız.

Room Kütüphanesi Temel Yapıları



1

Kurulum

```
//Room
apply plugin: 'kotlin-kapt'
```














```
apply plugin: "kotlin-kapt"
```

```
dependencies {
    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.compose.ui:ui:$compose_version"
    implementation "androidx.compose.material:material:$compose_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_version"
    //ViewModel
    implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"
    //Live data
    implementation "androidx.compose.runtime:runtime-livedata:1.0.0-beta08"
    //Room
    implementation "androidx.room:room-runtime:2.3.0"
    kapt "androidx.room:room-compiler:2.3.0"
    //Coroutine
    implementation "androidx.room:room-ktx:2.3.0"
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.3.1"
}
```

```
//ViewModel
implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"
//Live data
implementation "androidx.compose.runtime:runtime-livedata:1.0.0-beta08"
//Room
implementation "androidx.room:room-runtime:2.3.0"
kapt "androidx.room:room-compiler:2.3.0"
//Coroutine
implementation "androidx.room:room-ktx:2.3.0"
implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.3.1"
```

2

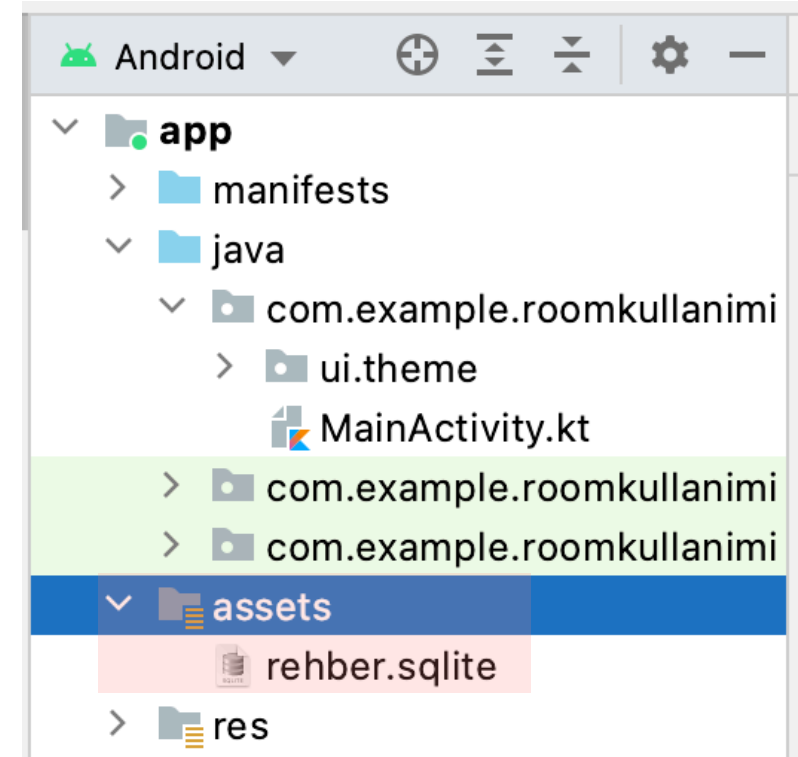
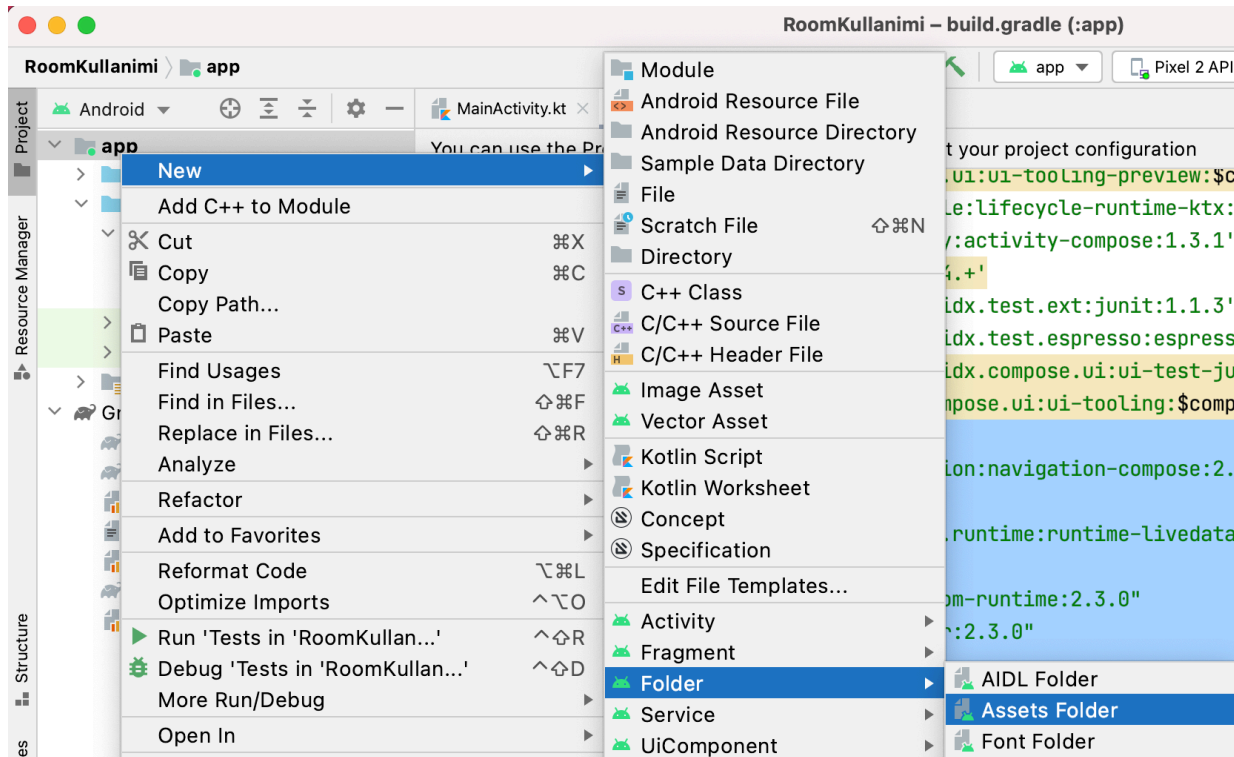
DB Browser ile Veri tabanı Oluşturma

 Tablo Oluştur	 Index Oluştur	 Tabloyu Düzenle	 Tabloyu Sil	»
İsim	Tip	Şema		
▼  Tablolar (2)				
▼  kisiler	CREATE TABLE "kisiler" ("kisi_id" INTEGER NO			
 kisi_id	INTEGER	"kisi_id" INTEGER NOT NULL		
 kisi_ad	TEXT	"kisi_ad" TEXT NOT NULL		
 kisi_tel	TEXT	"kisi_tel" TEXT NOT NULL		
>  sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)			
 İndisler (0)				
 Görünümler (0)				
 Tetikleyiciler (0)				

NOT : Tablo alanları NOT NULL olmalıdır.

3

Veri tabanını Assets Dosyasına Aktarma



4

Veritabanı Tablosu için Entity Oluşturma

Sınıfın veritabanında
temsil edeceği tablo adı

```
@Entity(tableName = "kisiler")
```

Veri tabanı tablo alan adı

```
data class Kisiler(@PrimaryKey(autoGenerate = true)
@ColumnInfo(name = "kisi_id") @NotNull var kisi_id: Int,
@ColumnInfo(name = "kisi_ad") @NotNull var kisi_ad: String,
@ColumnInfo(name = "kisi_tel") @NotNull var kisi_tel: String): Serializable {
}
```

Primary Key ve Auto Increment özellikleri

Veritabanındaki alanlar NOT NULL özelliğine sahip olmalıdır. Aksi halde hata alırız.
Veritabanı alanında String ifade not null ise entity içinde @NotNull ifadesi yazmalıyız
primitif tiplere @NotNull ifadesi gerekli değildir ama veritabanı modelinizin böyle olduğu görebilmek için primitif tiplere @NotNull ifadesi ekleyebilirsiniz.

kisiler		CREATE TABLE "kisiler" ("kisi
kisi_id	INTEGER	"kisi_id" INTEGER NOT NULL
kisi_ad	TEXT	"kisi_ad" TEXT NOT NULL
kisi_tel	TEXT	"kisi_tel" TEXT NOT NULL

5

Dao Interface Sınıfı

Dao Notasyonu

Metodun
veritabanında
yapacağı işlem türü.

Bir kere çalışır ve sonuç
döner

```

@Dao
interface KisilerDao {

    @Query(value: "SELECT * FROM kisiler")
    suspend fun tumKisiler(): List<Kisiler>

    @Insert
    suspend fun kisiEkle(kisiler: Kisiler)

    @Update
    suspend fun kisiGuncelle(kisiler: Kisiler)

    @Delete
    suspend fun kisiSil(kisiler: Kisiler)

    @Query(value: "SELECT count(*) FROM kisiler WHERE kisi_ad=:kisi_ad")
    suspend fun kayitKontrol(kisi_ad:String): Int

    @Query(value: "SELECT * FROM kisiler WHERE kisi_id=:kisi_id")
    suspend fun kisiGetir(kisi_id:Int): Kisiler

    @Query(value: "SELECT * FROM kisiler WHERE kisi_ad like '%' || :aramaKelimesi || '%'"')
    suspend fun kisiArama(aramaKelimesi:String): List<Kisiler>

    @Query(value: "SELECT * FROM kisiler ORDER BY RANDOM() LIMIT 1")
    suspend fun rasgele1KisiGetir(): List<Kisiler>
}

```

- Veritabanı üzerinde işlem yapacak metodlarımızı oluşturduğumuz sınıftır.
- İşlemleri asenkron olarak yapabilmek için Coroutine kullanılmıştır.

Veritabanı sınıfı oluşturma

Veritabanı üzerinde çalışabilmek için bu sınıfı oluşturmamız gerekir.

Bu sınıfa istediğimiz kadar entity ekleyebiliriz.

```
@Database(entities = [Kisiler::class], version = 1)
abstract class Veritabani : RoomDatabase() {
    abstract fun kisilerDao(): KisilerDao

    companion object {
        var INSTANCE: Veritabani? = null

        fun veritabaniErisim(context: Context): Veritabani? {
            if (INSTANCE == null){

                synchronized(Veritabani::class){

                    INSTANCE = Room.databaseBuilder(context.applicationContext,
                        Veritabani::class.java,
                        name: "rehber.sqlite")
                        .createFromAsset( databaseFilePath: "rehber.sqlite")
                        .build()

                }

            }

            return INSTANCE
        }
    }
}
```

← *entities kısmına başka sınıfları ekleyebiliriz.*

← *Veritabanı üzerinde işlem yapabilmek için dao sınıfına erişmeliyiz.*

← *Veritabanı üzerinde çalışmak için nesnemiz.*

← *Veritabanı adı*

← *Asset dosyasındaki veritabanı adı*

Asset dosyasındaki veritabanı alınır ve otomatik olarak telefona kopyalanır.

Veritabanına Erişim

```
@Composable
fun Sayfa() {
    val context = LocalContext.current
    val vt = Veritabani.veritabaniErisim(context)!!

    LaunchedEffect( key1: true){ this: CoroutineScope
        tumKisiler(vt)
    }
}
```

```
fun tumKisiler(vt:Veritabani){
    val job: Job = CoroutineScope(Dispatchers.Main).launch {
        val liste = vt.kisilerDao().tumKisiler()

        for (k in liste){
            Log.e( tag: "*****", msg: "*****")
            Log.e( tag: "Kişi id",k.kisi_id.toString())
            Log.e( tag: "Kişi ad",k.kisi_ad)
            Log.e( tag: "Kişi tel",k.kisi_tel)
        }
    }
}
```

Select - Veri Okuma

*suspend olduğu için
veri okuma işlemi
gerçekleşir ve işlemi
biter.*

```
@Query( value: "SELECT * FROM kisiler")  
suspend fun tumKisiler(): List<Kisiler>
```

```
fun tumKisiler(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch {  
        val liste = vt.kisilerDao().tumKisiler()  
  
        for (k in liste){  
            Log.e( tag: "*****", msg: "*****")  
            Log.e( tag: "Kişi id",k.kisi_id.toString())  
            Log.e( tag: "Kişi ad",k.kisi_ad)  
            Log.e( tag: "Kişi tel",k.kisi_tel)  
        }  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Insert – Veri Kaydı

```
@Insert  
suspend fun kisiEkle(kisiler: Kisiler)
```

```
fun ekle(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope  
        val yeniKisi = Kisiler( kisi_id: 0, kisi_ad: "Mehmet", kisi_tel: "33333")  
        vt.kisilerDao().kisiEkle(yeniKisi)  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

*Kişi id önemli değildir veritabanı
kisi_id'yi otomatik oluşturur.
Oyüzden öylesine 0 yazıldı.*

Update – Veri Güncelleme

```
@Update  
suspend fun kisiGuncelle(kisiler: Kisiler)
```

```
fun guncelle(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope  
        val kisi = Kisiler( kisi_id: 3, kisi_ad: "Yeni Mehmet", kisi_tel: "99999")  
        vt.kisilerDao().kisiGuncelle(kisi)  
    }  
}
```



kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Güncelleme yaparken nesnenin
kisi_id kısmına güncelleme
yapılacak kisi_id eklenir.
Diğer alanlara güncellemek
istediğimiz alanlar yazılır.

Delete – Veri Silme

```
@Delete  
suspend fun kisiSil(kisiler: Kisiler)
```

```
fun sil(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope  
        val kisi = Kisiler( kisi_id: 3, kisi_ad: "Yeni Mehmet", kisi_tel: "99999")  
        vt.kisilerDao().kisiSil(kisi)  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Silme yaparken nesnenin kisi_id kısmına silme yapılacak kisi_id eklenir. Diğer alanlara silme işleminde önemli değildir.

Select - Kayıt Kontrol

```
@Query( value: "SELECT count(*) FROM kisiler WHERE kisi_ad=:kisi_ad")
suspend fun kayitKontrol(kisi_ad:String): Int
```

Sorgu içinde parametre kullanma : ve daha sonra metod parametre adı yazılarak ile yapılır.

```
fun kontrol(vt:Veritabani){
    val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope
        val sonuc = vt.kisilerDao().kayitKontrol( kisi_ad: "ahmet")
        Log.e( tag: "Sonuç",sonuc.toString())
    }
}
```

Girilen isimden tabloda kaç tane olduğunu gösterir.

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Select – Bir Kayıt Getirme

```
@Query( value: "SELECT * FROM kisiler WHERE kisi_id=:kisi_id")  
suspend fun kisiGetir(kisi_id:Int): Kisiler
```

```
fun getir(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch {  
        val k = vt.kisilerDao().kisiGetir( kisi_id: 1)  
        Log.e( tag: "Kişi id",k.kisi_id.toString())  
        Log.e( tag: "Kişi ad",k.kisi_ad)  
        Log.e( tag: "Kişi tel",k.kisi_tel)  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Select – Arama İşlemi

```
@Query(value: "SELECT * FROM kisiler WHERE kisi_ad like '%' || :aramaKelimesi || '%'")  
suspend fun kisiArama(aramaKelimesi:String): List<Kisiler>
```

```
fun ara(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope  
        val liste = vt.kisilerDao().kisiArama( aramaKelimesi: "z")  
  
        for (k in liste){  
            Log.e( tag: "*****", msg: "*****")  
            Log.e( tag: "Kişi id",k.kisi_id.toString())  
            Log.e( tag: "Kişi ad",k.kisi_ad)  
            Log.e( tag: "Kişi tel",k.kisi_tel)  
        }  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Select – Rasgele ve Sınırlı Sayıda

```
@Query(value: "SELECT * FROM kisiler ORDER BY RANDOM() LIMIT 1")  
suspend fun rasgele1KisiGetir(): List<Kisiler>
```

```
fun rasgele(vt:Veritabani){  
    val job: Job = CoroutineScope(Dispatchers.Main).launch {  
        val liste = vt.kisilerDao().rasgele1KisiGetir()  
  
        for (k in liste){  
            Log.e( tag: "*****", msg: "*****")  
            Log.e( tag: "Kişi id",k.kisi_id.toString())  
            Log.e( tag: "Kişi ad",k.kisi_ad)  
            Log.e( tag: "Kişi tel",k.kisi_tel)  
        }  
    }  
}
```

kisiler	Türü
kisi_id	int
kisi_ad	String
kisi_tel	String

Teşekkürler...



kasım-adalan



kasimadalan@gmail.com



kasimadalan