O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKASIYALARINI RIVOJLANTIRISH VAZIRLIGI
MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

INDIVIDUAL LOYIHA

Mavzu: “Wallet” nomli elektron hamyon ilovasini ishlab chiqish..

CSP014 guruh
Bajardi: Jo’rayev Xudoyshukur

Reja:

1. Kirish. Wallet ilovasining maqsad va vazifalari.
2. Asosiy qism
    a) Ilova arxitekturasi va Ilovani ishlab chiqishda ishlatilgan Android texnologiyalar
    b) Ilovaning umumiy imkoniyatlari
3. Xulosa. Dastur yaratilishi borasidagi xulosalar.

KIRISH

Hech kimga sir emaski, mablag’larni to’g’ri sarflay bilish va nazorat qilish muhim ishlar sirasiga kiradi. Bu bilan siz harajatlaringiz qancha qismi qayerga ketyapti. Kimdan qarz olgansiz, kimga qarz bergansiz va hk. larni qayd qilib borishingiz mumkin. Yuqorida keltirilgan ma’lumotlarni shaxsiy daftarchaga yozib borsa ham bo’ladi. Ammo nima uchun texnologiyalar asrida yashayapmiz-u, daftar o’rniga qo’limizdagi mobil telefonlardan foydalanmaymiz?

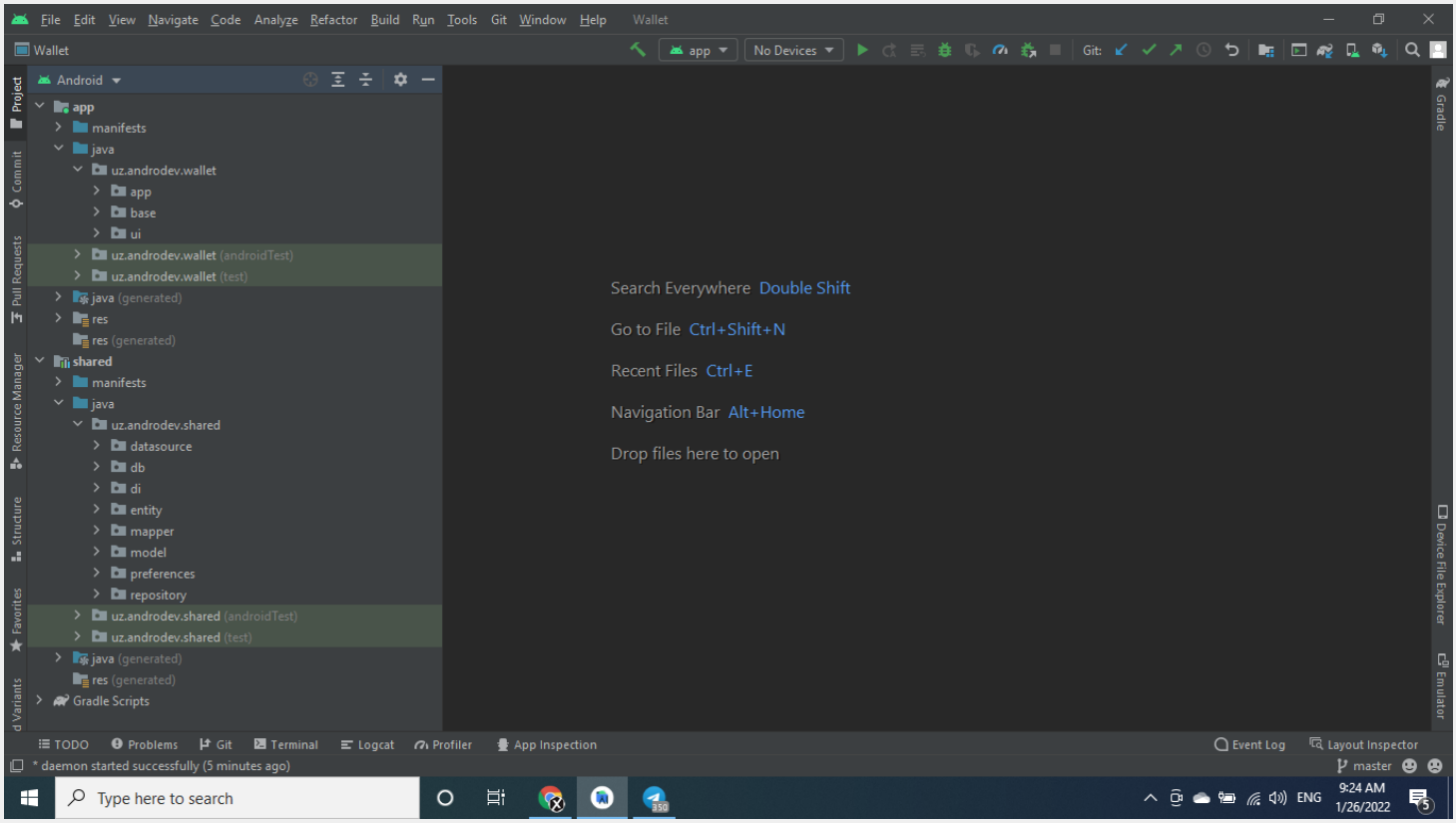
Wallet ilovasi ham ana shu maqsadda ishlab chiqildi. Siz maosh yoki stipendiya olsangiz, nimagadir harajat qilsangiz o’sha zahoti Android tizimida ishlovchi qo’l telefonngizga qayd qilib boraverasiz. Siz kirim yoki chiqim qilsangiz, u haqida kategoriya tanlashingiz va tafsilotlar kiritilishi so’raladi. Bu bilan siz harajatlar tarixini bemalol ko’ra olasiz. Dasturdagi kategoriyalardan tashqari, o’zingiz ham kategoriyalar qo’sha olasiz.

Dasturdan asosiy maqsad, pul mablag’larini boshqarish. Dasturning kelgusi verisyalari ishlab chiqilsa, statistika uchun turli graflar, chartlar bo’lishi mumkin.

ASOSIY QISM

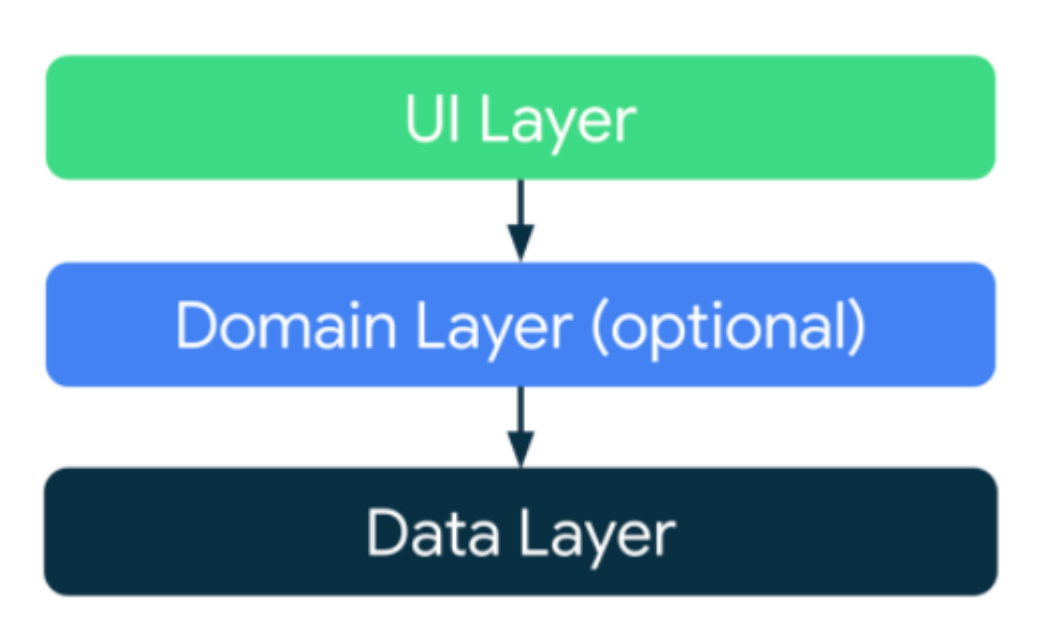
Wallet ilovasi Andorid Studio IDE yordamida kotlin dasturlash tilida yaratildi. Ilovaning asosiy arxitekturasi clean architecture tamoyillariga asoslangan. Dasturda 2 ta layer (qatlam) dan iborat:

1. Presentation layer (app module)
2. Data layer (shared module)

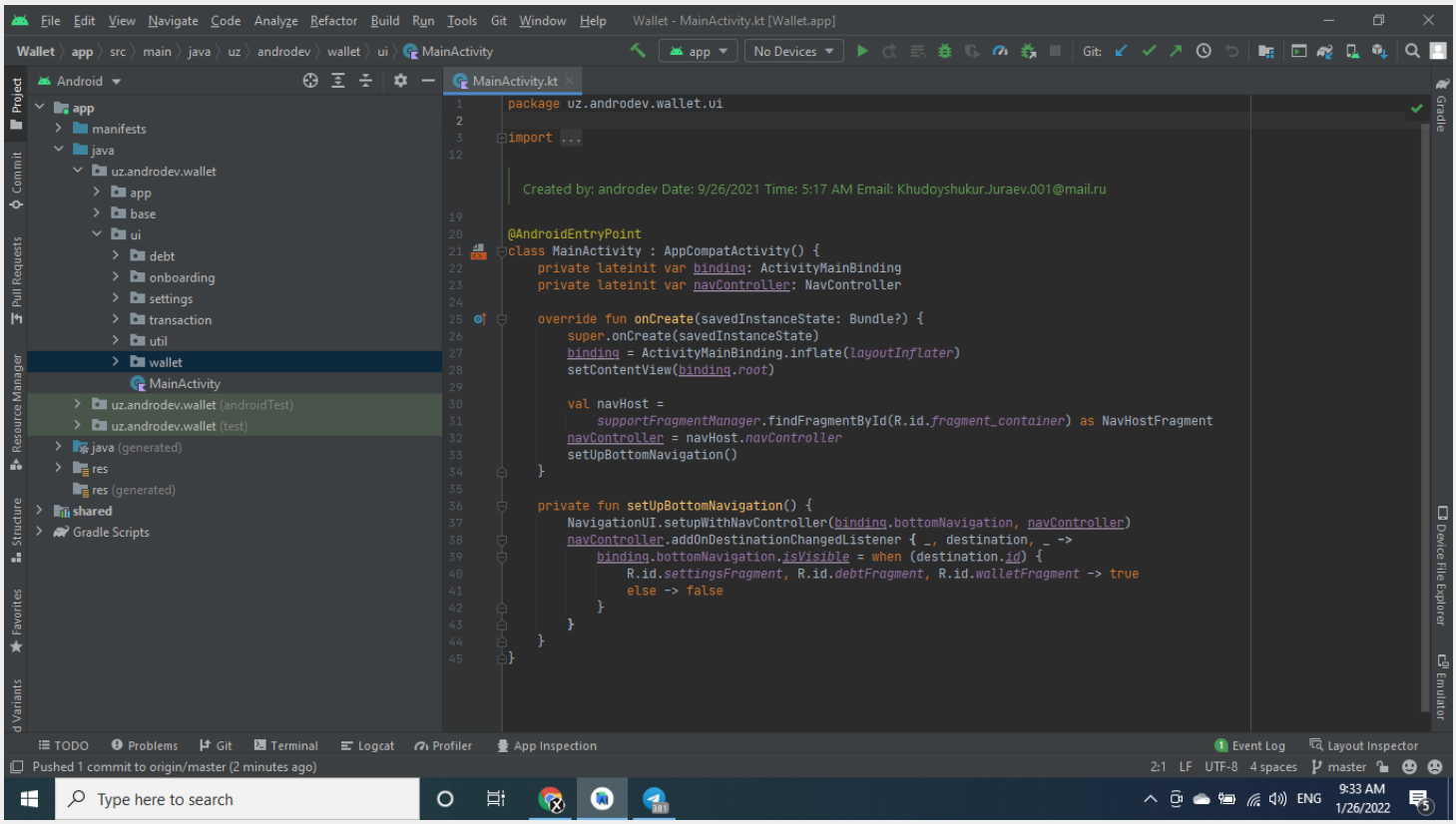


1-rasm. Ilovaning asosiy arxitekturasi papkalar asosida ko'rsatilgan.

Presentation layerda foydalanuvchi interfeysiga aloqador qismlar saqlanadi. Bu qismdan olingan event (masalan server bilan aloqa uchun biror knopkani bosish) data layerga uzatiladi. Presentation layer MVVM (Model-View-ViewModel) pattern bo'yicha qurildi. Dastur SingleActivity tamoyilida ishlaydi. Ya'ni dastruda yagona MainActivity mavjud. Dasturning barcha sahifalari Fragment lardan iborat. MainActivityning Fragment Containerida logikaga qarab sahifalar almashtiriladi.

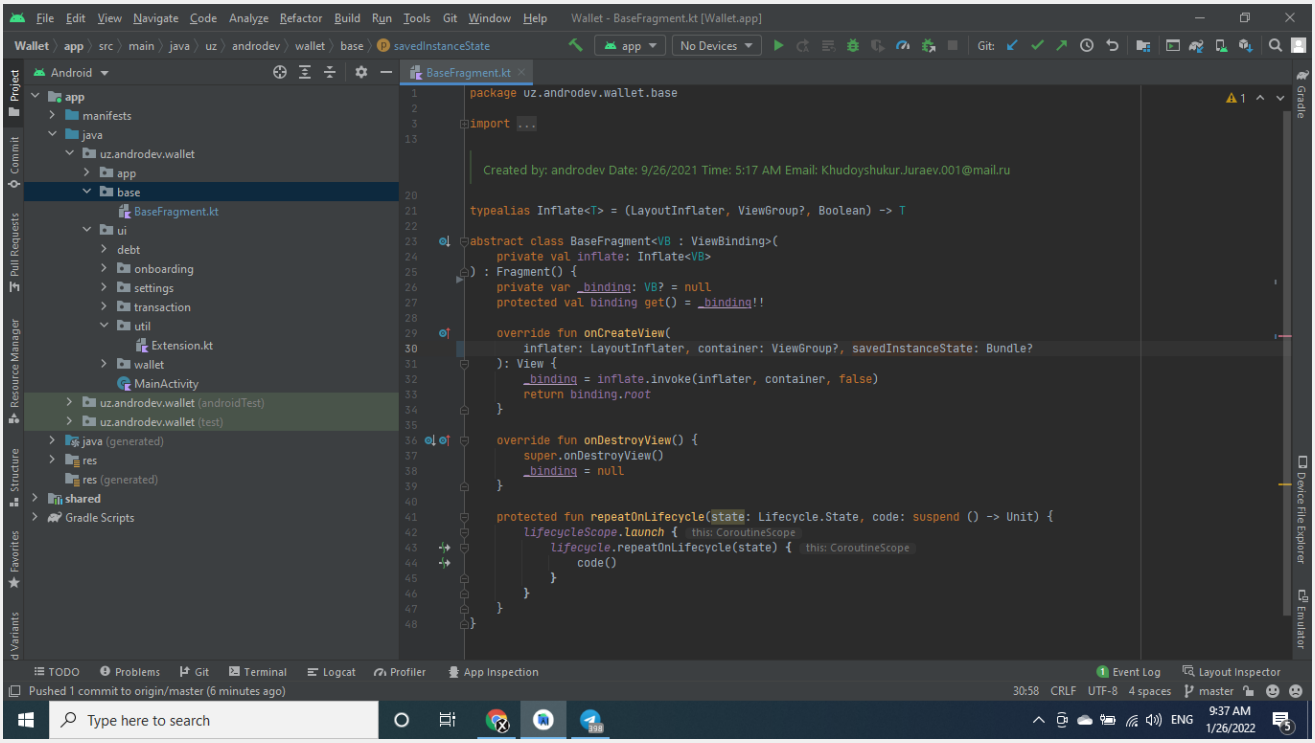


Dasturda yuqoridagi google tomonidan taklif etilgan arxitektura ishlatildi. Dasturimiz juda murakkab bo'lmaganligi uchun Domain layer ishlatilmadi.



2-rasm. Ilovadagi MainActivity ko'rsatilgan.

Yuqorida aytilganidek presentation layer sahifalari Fragmentlardan iborat. Dasturda view lardan foydalanish uchun ViewBinding API ishlatilgan va barcha umumiy fragmentlar uchun BaseFragment.kt abstract klasi yaratilgan.



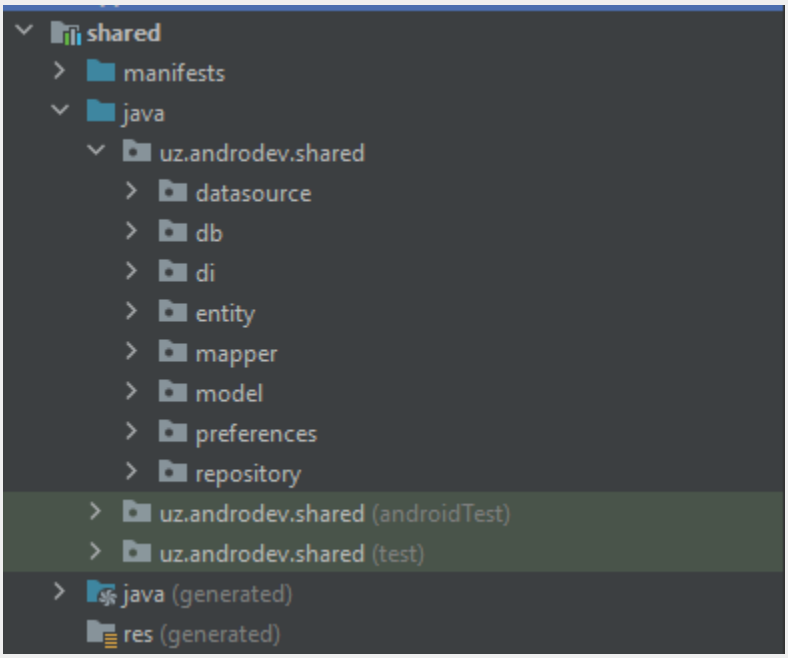
3-rasm. BaseFragment.kt klasi

Presentation layerda ishlatilgan fragmentlardan biri TransactionFragment dir. U orqali siz kirim va chiqimlarni kiritasiz. Ushbu fragment BottomSheetDialogFragment dan nasl olgan holda yaratildi:

```kotlin
/**
 * Created by: androdev
 * Date: 9/26/2021
 * Time: 5:17 AM
 Email: Khudoyshukur.Juraev.001@mail.ru
 */

@AndroidEntryPoint
class TransactionFragment : BottomSheetDialogFragment() {

    private var _binding: FragmentTransactionBinding? = null
    private val binding get() = _binding!!

    private val viewModel: TransactionViewModel by viewModels()
    private val navArgs by navArgs<TransactionFragmentArgs>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        viewModel.fetchCategories(navArgs.transactionType)
        observeUiState()
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentTransactionBinding.inflate(inflater, container, attachToParent: false)
        return binding.root
    }
```

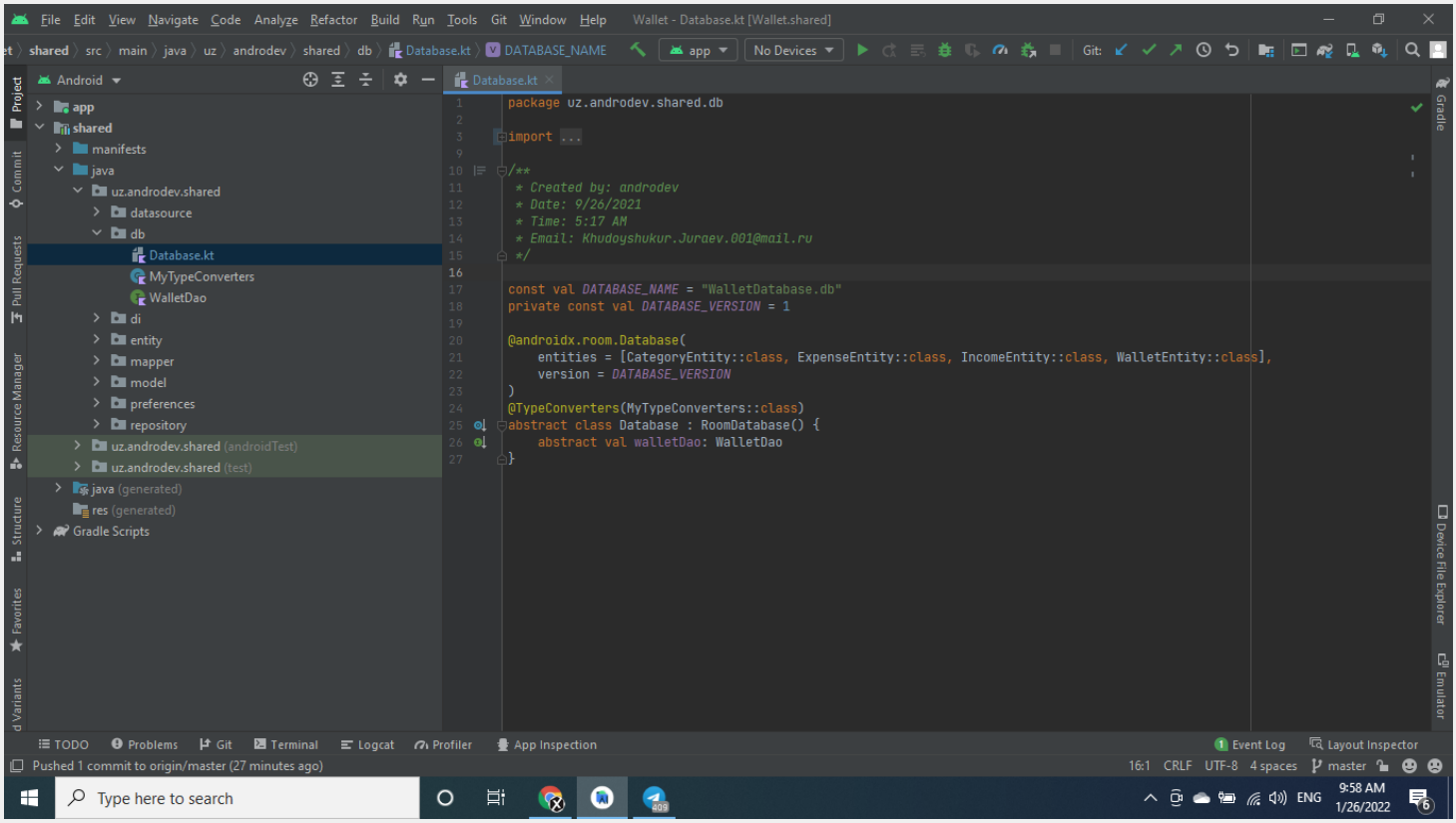
4-rasm. TransactionFragment.kt

Data layer o’zidan repositorylarni chiqaradi. Repositorylar orqali dasturda biz presentation layerda turib dataga murojaat qilamiz. Ya’ni datani o’zgartirishimiz, ko’rishimiz, o’chirishimiz va yangi data qo’shishimiz mumkin. Bizning ilovamizda data layer network bilan aloqa qilmaydi. Ma’lumotlarni serverda saqlamaymiz. Ma’lumotlar lokal SqlLite bazada saqlanadi. Biz to’g’ridan to’g’ri SqlLite bazaga murojaat qilmay, o’rniga SqlLite ustiga abstrakt qilingan Room ma’lumotlar bazasini ishlatdik.



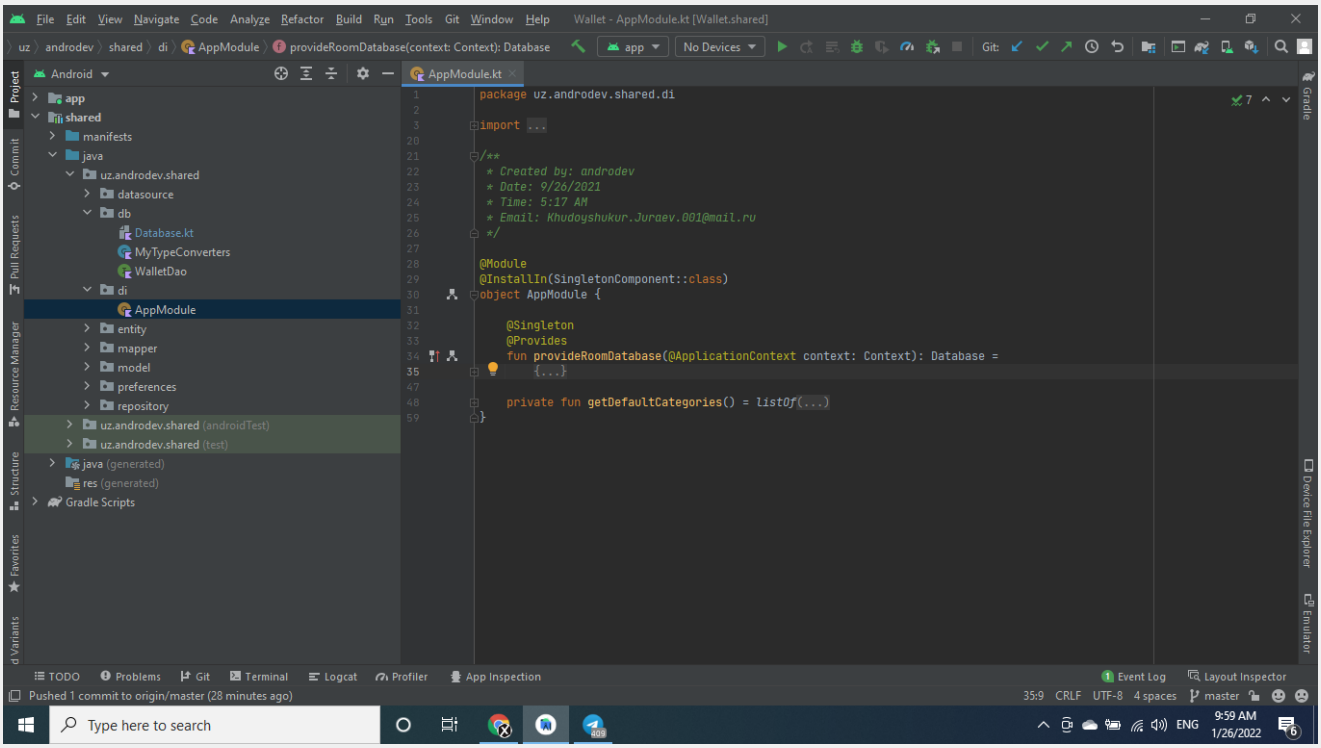
5-rasm. Data layer ko’rsatilgan.

5-rasm da data layer ko’rsatilgan.
Repository papkasida repositorylar saqlanadi.
Db papkasida Room database ga oid klasslar bo’ladi
Entity papkasida database jadvallari saqlanadi
Modelda presentationga chiqariladigan data class lar bo’ladi
Mapperda Data layerdagi entitylar va presentation layer data classlari o’rtasida munosabat o’rnatiladi.
Datasource room database bilan ishlashni abstraktsiyalaydi
Di da dependency injection ga oid classlar bo’ladi.



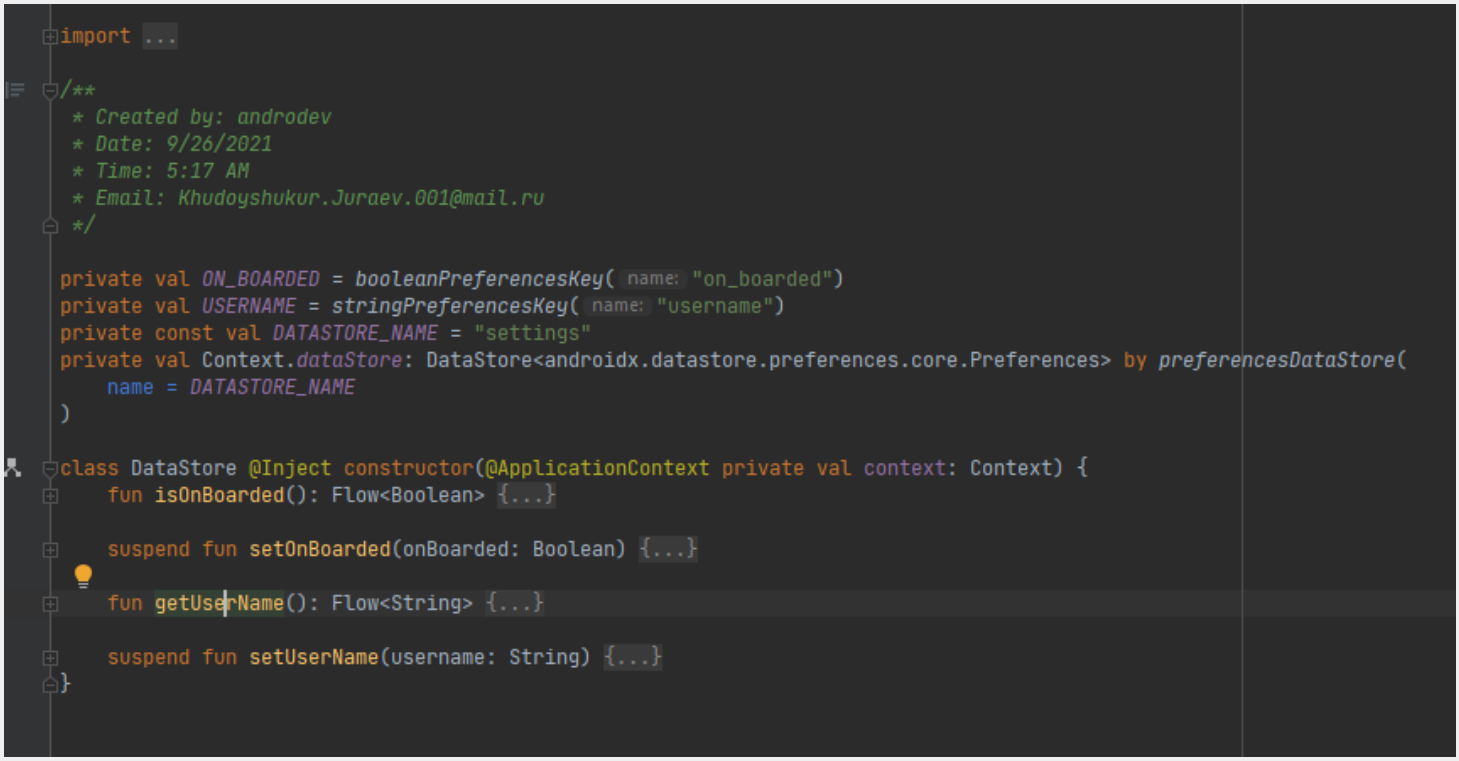
6-rasm. Database

Dasturda dependency injection uchun Hilt kutubxonasi ishlatilgan. AppModule.kt klasida provider funksiyalar yozilgan.



7-rasm. Dependency injection @Module

Dasturdagi kichik ma’lumotlarni saqlash uchun Jetpack DataStore ishlatilgan:

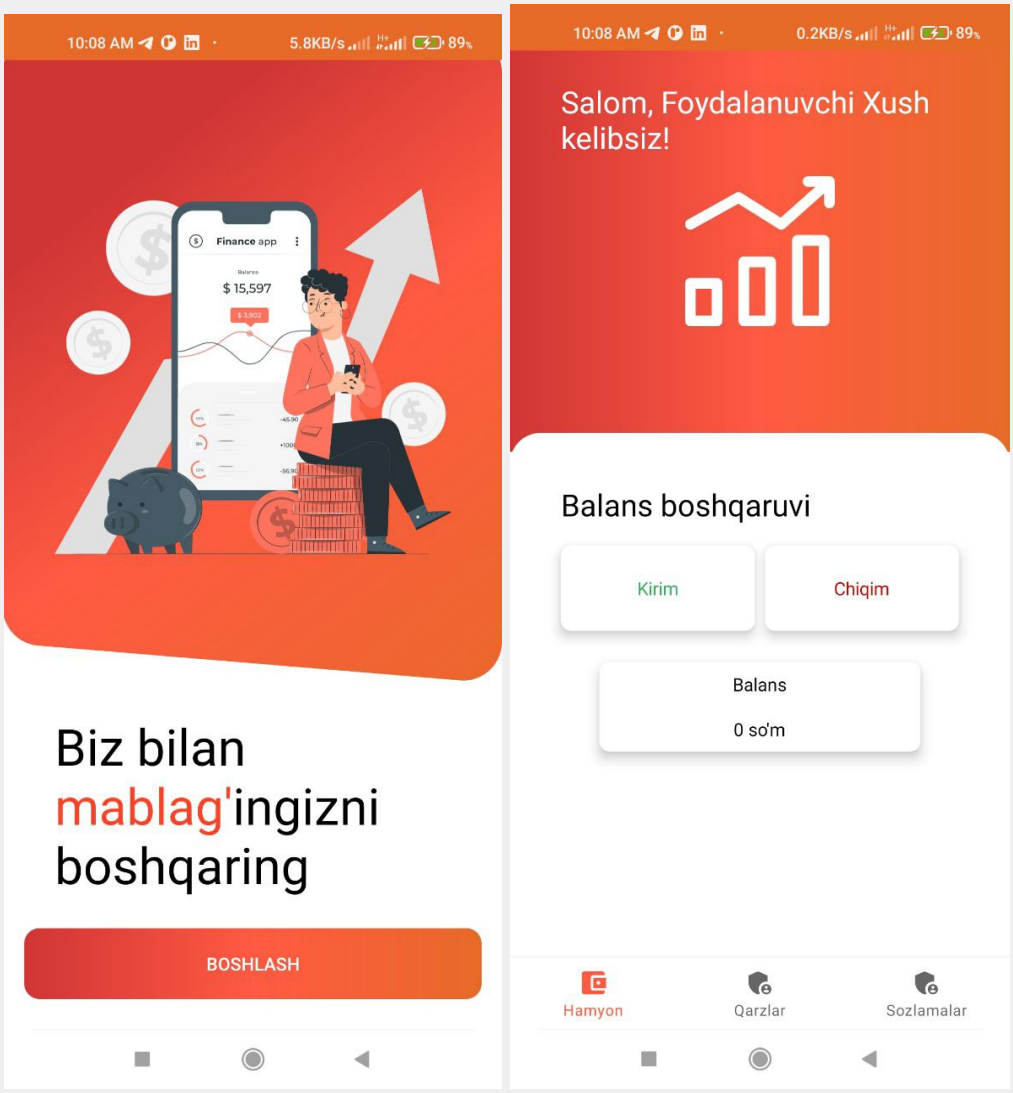


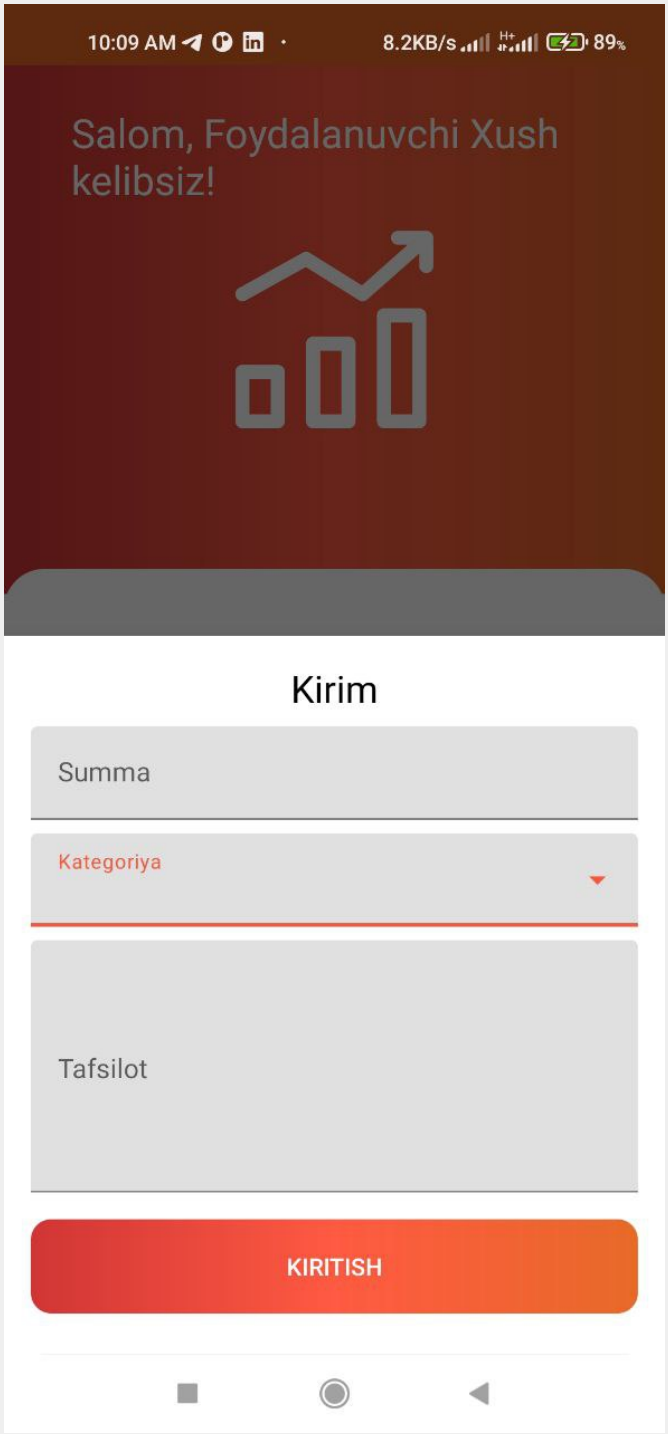
```
import ...

/**
 * Created by: androdev
 * Date: 9/26/2021
 * Time: 5:17 AM
 * Email: Khudoyshukur.Juraev.001@mail.ru
 */

private val ON_BOARDED = booleanPreferencesKey( name: "on_boarded")
private val USERNAME = stringPreferencesKey( name: "username")
private const val DATASTORE_NAME = "settings"
private val Context.dataStore: DataStore<androidx.datastore.preferences.core.Preferences> by preferencesDataStore(
    name = DATASTORE_NAME
)

class DataStore @Inject constructor(@ApplicationContext private val context: Context) {
    fun isOnBoarded(): Flow<Boolean> {...}

    suspend fun setOnBoarded(onBoarded: Boolean) {...}

    fun getUserName(): Flow<String> {...}

    suspend fun setUserName(username: String) {...}
}
```

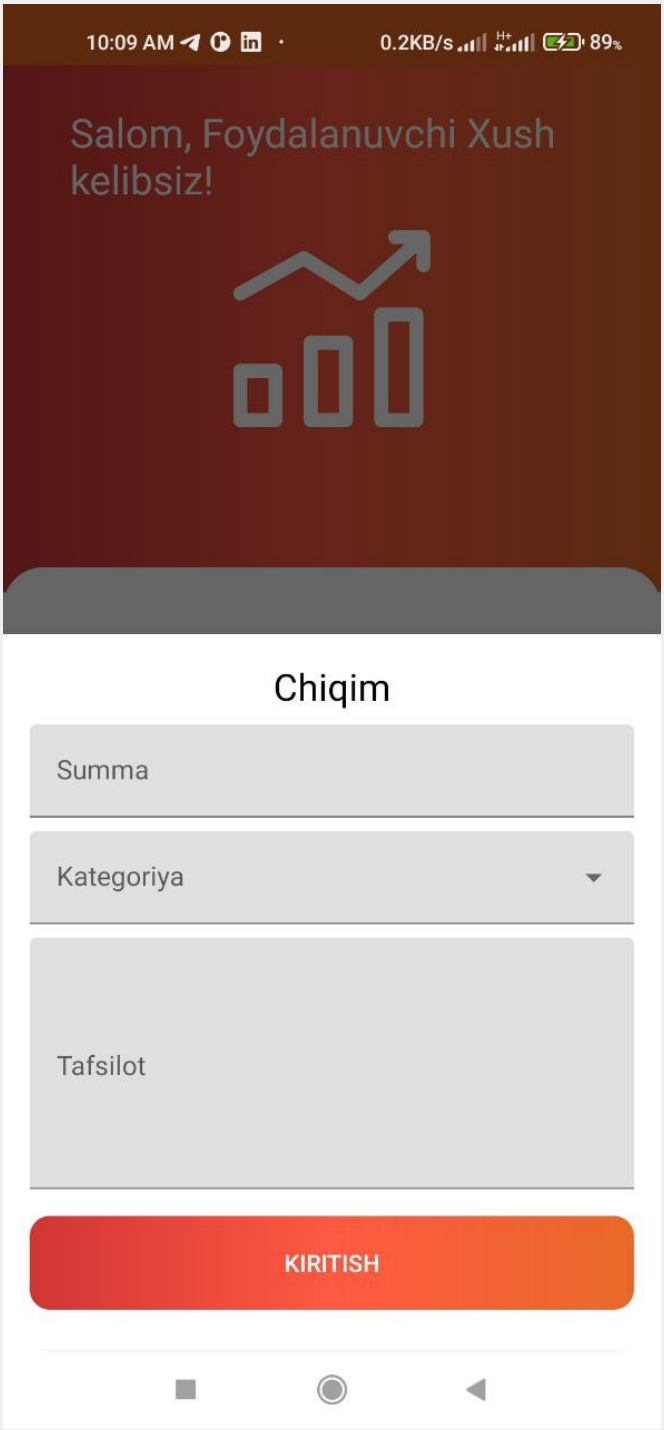
8-rasm. Jetpack DataStore

Dasturning umumiy imkoniyatlari:

Dasturda siz kirim va chiqim lar qo’shishingiz mumkin:




10:09 AM
8.2KB/s
89%
Salom, Foydalanuvchi Xush kelibsiz!
Kirim
Summa
Kategoriya
Tafsilot
KIRITISH


10:09 AM
0.2KB/s
89%
Salom, Foydalanuvchi Xush kelibsiz!
Chiqim
Summa
Kategoriya
Tafsilot
KIRITISH

10:10 AM 0.0KB/s 89%

Qarzlar

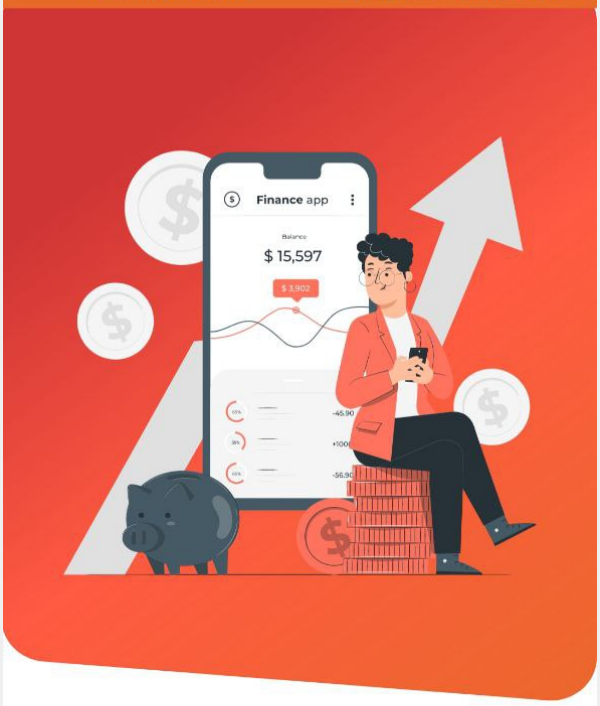
Akmal

200,000 so'm

26 yanvar, 2022

Hamyon Qarzlar Sozlamalar

10:09 AM 0.0KB/s 89%



Foydalanuvchi

Ismni o'zgartirish

Kategoriya qo'shish

Hamyon Qarzlar Sozlamalar

XULOSA

Dasturni yaratish davomida clean architecture tamoyillari qo’lladim. Fragment lifecycle, Activity lifecycle, debug qilish, Kotlin Flow lar bilan ishlash borasida ko’plab bilimlarimni mustahkamladim.

Ushbu dasturni internetda server bilan yangilab boradigan qilsa ham bo’ladi. Firebase eng yaxshi yechim bo’ladi bu masalada.