ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

SEMESTRÁLNA PRÁCA Z PREDMETU

Vývoj aplikácií pre mobilné zariadenia

QuickFinance

Autor: Jakub Slaninák

Štúdijná skupina: 5ZYR21

Obsah:

1 Popis a analýza riešeného problému

1.1 Špecifikácia zadania

Jedná sa o mobilnú aplikáciu QuickFinance, ktorá slúži na vzdelávanie a zábavu. Je určená na spracovanie osobných financií a bankovníctva. Aplikácia je navrhnutá tak, aby simulovala reálne bankové procesy a poskytla používateľom predstavu o tom, ako by mohli v reálnom svete spravovať svoje financie cez mobilnú aplikáciu. Aplikácia QuickFinance ponúka niekoľko kľúčových funkcií, ktoré simulujú procesy reálneho bankovníctva a poskytujú používateľom predstavu o správe osobných financií.

Správa účtov zahŕňa bezpečný autentifikačný systém, ktorý demonštruje proces prihlásenia do bankovej aplikácie. Používatelia sa môžu prihlásiť a následne si môžu zobraziť aktuálny stav svojho účtu, čo im poskytuje prehľad o zostatku na účte.

Finančný prehľad je zobrazený na domovskej obrazovke, ktorá poskytuje simulovaný prehľad finančného stavu používateľa. Tento prehľad zahŕňa nedávne transakcie a dostupný zostatok.

Transakcie umožňujú používateľom iniciovať nové platby zadaním údajov o príjemcovi a sumy transakcie. Tento proces ukazuje, ako by mohlo vyzerať zadávanie platby v reálnej bankovej aplikácii. Aplikácia taktiež umožňuje zobrazenie histórie transakcií, čo demonštruje spôsob sledovania výdavkov a príjmov.

Celkovo QuickFinance poskytuje používateľom názornú ukážku toho, ako by mohli efektívne spravovať svoje financie pomocou mobilnej aplikácie, zlepšiť prehľad o výdavkoch a príjmoch a zvýšiť finančnú gramotnosť.

1.2 Podobné aplikácie

QuickFinance vs. Tatra banka

QuickFinance je demo aplikácia určená na simuláciu správy osobných financií. Jej hlavnou úlohou je poskytnúť používateľom predstavu o tom, ako by mohli v reálnom svete spravovať svoje financie pomocou mobilnej aplikácie. QuickFinance obsahuje základné funkcie, ako je simulované prihlásenie, zobrazenie fiktívneho zostatku na účte, prehľad finančného stavu a možnosť simulovať transakcie. Celkovo je aplikácia zameraná na vzdelávanie a demonštráciu, nie na spracovanie skutočných finančných operácií.

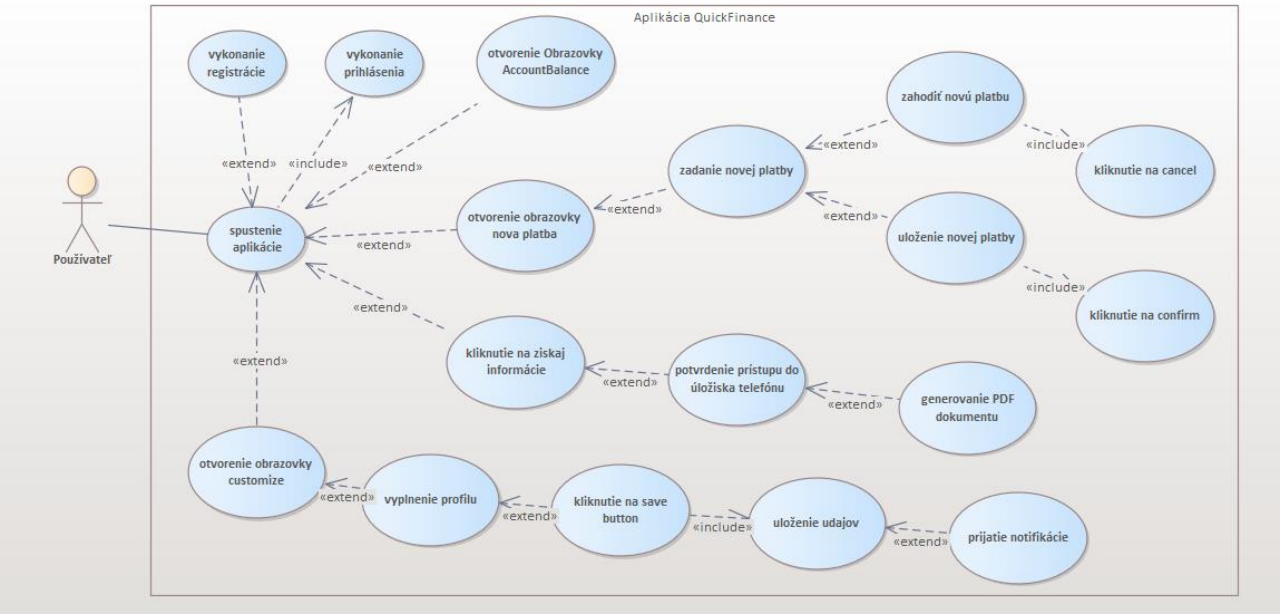
Na druhej strane, Tatra banka je profesionálna banková aplikácia poskytujúca plnohodnotné bankové služby. Používatelia môžu spravovať svoje reálne bankové účty, vykonávať finančné transakcie, investovať, žiadať o úvery a spravovať svoje poistenia. Aplikácia ponúka vysokú úroveň bezpečnosti, vrátane viacúrovňového overovania, šifrovania údajov a biometrickej autentifikácie. Tatra banka aplikácia je pravidelne aktualizovaná a podporovaná tímom odborníkov, čo zaručuje jej spoľahlivosť a bezpečnosť pre používateľov.

QuickFinance vs. George Slovensko

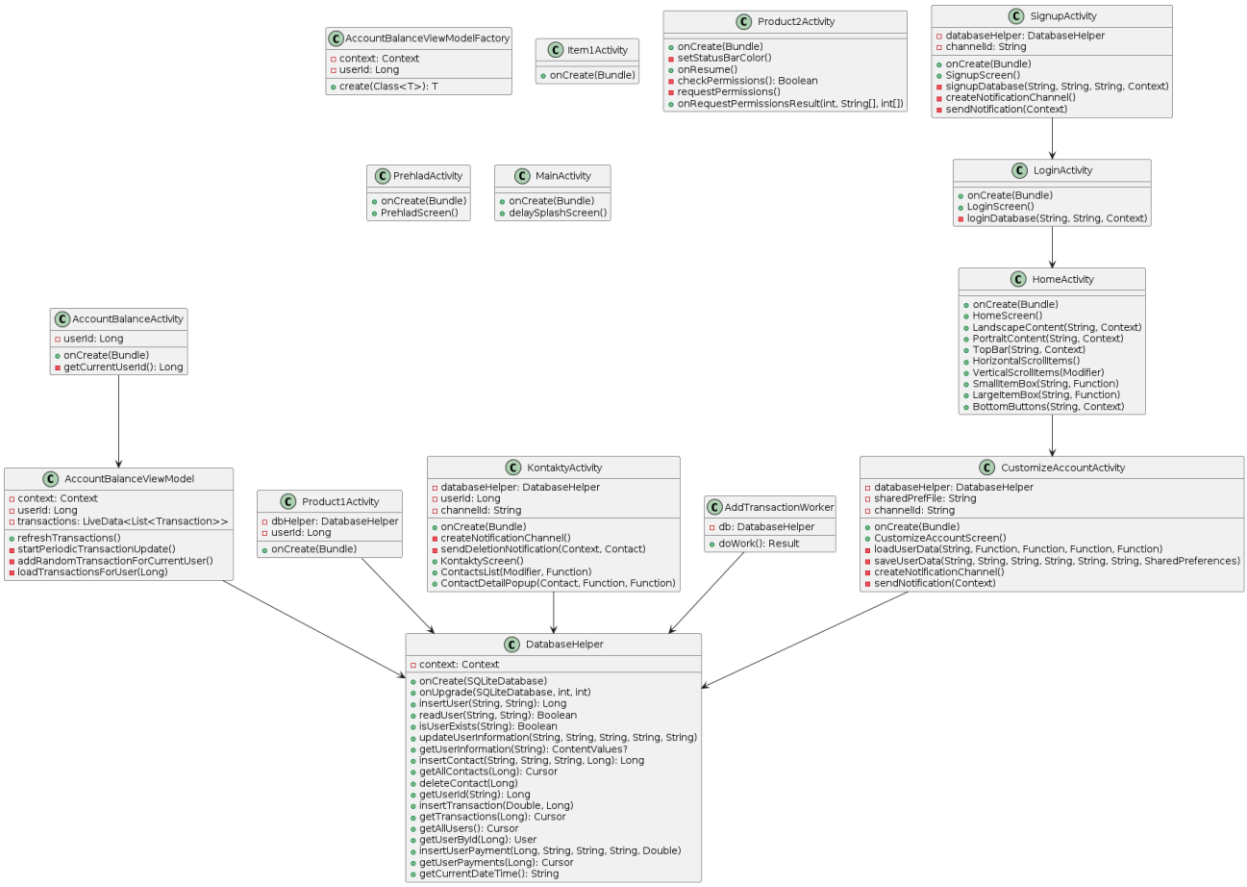
George Slovensko je pokročilá banková aplikácia, ktorá ponúka širokú škálu bankových služieb a produktov. Používatelia môžu spravovať svoje bankové účty, vykonávať okamžité platby, investovať do rôznych finančných nástrojov, platiť faktúry a sledovať svoje finančné výdavky pomocou prepracovaných analytických nástrojov. Aplikácia George je známa svojím moderným dizajnom a intuitívnym používateľským rozhraním. Poskytuje vysokú úroveň bezpečnosti, vrátane šifrovania údajov a biometrického overovania. Je podporovaná veľkým tímom vývojárov a IT odborníkov, ktorí zaisťujú jej neustálu aktualizáciu a údržbu, čo zaručuje spoľahlivosť a bezpečnosť pre používateľov.

2 Návrh riešenia problému

2.1 Diagram prípadov použitia



2.2 Diagram tried



3 Popis implementácie

3.1 Komplexné obrazovky

Obrazovky ako AccountBalanceActivity, Product1Activity KontaktyActivity, HomeActivity, LoginActivity a SignupActivity demonštrujú unikátne funkcie aplikácie, ako sú správa transakcií, zadávanie platieb, správa kontaktov, prihlásenie a registrácia.

3.2 Komponenty LifeCycle, ViewModel, WorkManager

Použitie komponentov ako ViewModel a WorkManager je demonštrované v AccountBalanceActivity a AddTransactionWorker.

LifecycleKomponent Lifecycle je implicitne používaný v ComponentActivity, čo je základná trieda pre aktivity v Jetpack Compose, ktorá poskytuje podporu pre lifecycle komponenty.

ViewModelKomponent

ViewModel je použitý na správu dát pre užívateľské rozhranie a jeho životný cyklus je nezávislý na aktivitách alebo fragmentoch. V mojom kóde je ViewModel použitý takto: V triede AccountBalanceViewModel:

```
class AccountBalanceViewModel(private val context: Context, private val userId: Long) : ViewModel() {
    private val db = DatabaseHelper(context)
    private val _transactions = MutableLiveData<List<Transaction>>()
    val transactions: LiveData<List<Transaction>> = _transactions
```

WorkManagerKomponent

WorkManager je použitý na plánovanie periodických úloh na pozadí. V kóde je WorkManager použitý takto:

```
class AddTransactionWorker(context: Context, workerParams: WorkerParameters) : Worker(context, workerParams) {
    private val db = DatabaseHelper(context)

    jakubJKS
    override fun doWork(): Result {
        val cursor = db.getAllUsers()
        with(cursor) { this: Cursor
            while (moveToNext()) {
                val userId = getLong(getColumnIndexOrThrow(DatabaseHelper.COLUMN_ID))
                val randomAmount = Random.nextDouble( from: 10.0,  until: 150.0)
                val user = db.getUserById(userId)
                if (user.createdAt <= db.getCurrentDateTime()) {
                    db.insertTransaction(randomAmount, userId)
                }
            }
            close()
        }
        return Result.success()
    }
}
```

Padding

Príklad použitia padding

```
@Composable
fun PortraitContent(username: String, context: Context) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) { this: ColumnScope
        TopBar(username, context)
        Spacer(modifier = Modifier.height(16.dp))
        Divider(color = Color.White, thickness = 1.dp)
        Spacer(modifier = Modifier.height(16.dp))
        HorizontalScrollItems()
        Spacer(modifier = Modifier.height(16.dp))
        VerticalScrollItems(modifier = Modifier.weight(1f))

        BottomButtons(username, context)
    }
}
```

3.3 Notifikácie

Použitie notifikácií je demonštrované v časti aplikácie `KontaktyActivity`, kde sa notifikácia odošle pri odstránení kontaktu.

```
private fun sendDeletionNotification(context: Context, contact: Contact) {
    val builder = NotificationCompat.Builder(context, channelId)
        .setSmallIcon(R.drawable.ic_notification) // Ensure you have an icon drawable
        .setContentTitle("Contact Deleted")
        .setContentText("Deleted contact: ${contact.firstname} ${contact.lastname}")
        .setPriority(NotificationCompat.PRIORITY_HIGH) // Set high priority to show heads-up notification
        .setVisibility(NotificationCompat.VISIBILITY_PUBLIC) // To ensure it appears on lock screen

    with(NotificationManagerCompat.from(context)) { this: NotificationManagerCompat
        if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.POST_NOTIFICATIONS) == android.content.pm.
            notify(contact.id.toInt(), builder.build())
        } else {
            // Request permission from the user
            androidx.core.app.ActivityCompat.requestPermissions(
                activity: this@KontaktyActivity,
                arrayOf(android.Manifest.permission.POST_NOTIFICATIONS),
                requestCode: 1
            )
        }
    }
}
```

3.4 Použitie externého frameworku / knižnice

androidx.compose.material3 je externá knižnica pre tvorbu používateľského rozhrania. androidx.work pre WorkManager. A nakoniec androidx.lifecycle pre ViewModel.

Surface
Surface je kontajnerový komponent, ktorý poskytuje rôzne štýly povrchov v rámci aplikácie. V kóde je použitý na zobrazenie obsahu v aplikáciách.

```
Surface(modifier = Modifier.fillMaxSize(), color = Color(0xFF121212)) {
    PdfGeneratorScreen() // alebo iné obrazovky
}
```

Text
Text komponent je používaný na zobrazenie textu.

```
Text(
    text = "PDF Generator",
    fontSize = 24.sp,
    fontWeight = FontWeight.Bold,
    color = Color.White
)
```

Button
Button komponent je používaný na zobrazenie tlačidiel, ktoré reagujú na kliknutie.

```
Button(
    onClick = {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            if (activity?.let { checkPermissions(it) } == true) {
                scope.launch {
                    generatePDF(context)
                }
            } else {
                activity?.let { requestPermissions(it) }
            }
        }
    },
    colors = ButtonDefaults.buttonColors(containerColor = Color.Green)
) {
    Text("Generate PDF")
}
```

MaterialTheme
MaterialTheme je používaný na definovanie globálnych štýlov a farieb pre komponenty v rámci aplikácie.

```
QuickFinanceTheme {
    Surface(modifier = Modifier.fillMaxSize(), color = Color(0xFF121212)) {
        AccountBalanceScreen(userId)
    }
}
```

Column
Column je layout kontajner, ktorý zoraďuje svoje deti vertikálne.

```
Column(
    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    Text(
        text = "PDF Generator",
        fontSize = 24.sp,
        fontWeight = FontWeight.Bold,
        color = Color.White
    )
    Spacer(modifier = Modifier.height(16.dp))
    Button(
        onClick = {
            // akcia po kliknutí
        },
        colors = ButtonDefaults.buttonColors(containerColor = Color.Green)
    ) {
        Text("Generate PDF")
    }
}
```

Spacer
Spacer je komponent používaný na pridanie prázdneho priestoru medzi komponentmi.

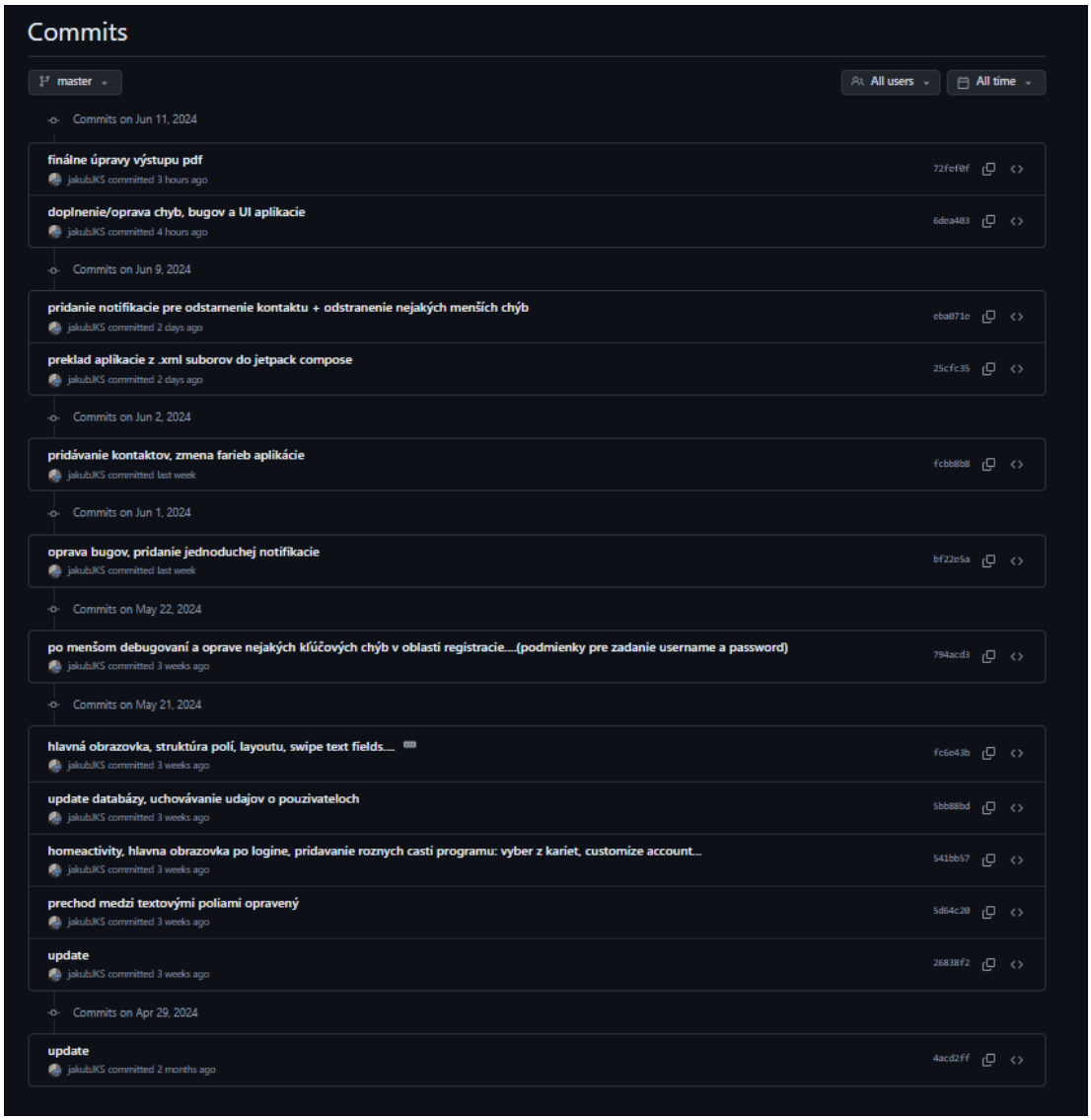
```
Spacer(modifier = Modifier.height(16.dp))
```

3.5 Využitie komponentu service (služby)

Workmanager je komponent služby používaný na periodické úlohy

```
// Plánovanie periodickej úlohy
val workRequest = PeriodicWorkRequestBuilder<AddTransactionWorker>(15, TimeUnit.MINUTES)
    .build()
WorkManager.getInstance(this).enqueueUniquePeriodicWork(
    "AddTransactionWork",
    ExistingPeriodicWorkPolicy.KEEP,
    workRequest
)
```

3.6 Práca s GIT-om



4 Zoznam použitých zdrojov

https://www.geeksforgeeks.org