ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu

vývoj aplikácií pre mobilné zariadenia

MOJEKARTY

Vypracoval: Filip Dubovec

Študijná skupina: 5ZYR23

Akademický rok: 2024/2025

V Žiline dňa 7. 6. 2025

Obsah

Obsah obrázkov

Úvod

Mobilné aplikácie dnes často nahrádzajú fyzické peňzenky. Vernostné karty a číselné či QR/Barcode identifikátory dnes využívajú takmer všetky obchody. Mojím cieľom je vytvoriť jednoduchú mobilnú aplikáciu, ktorá umožní uchovávať a spravovať tieto karty digitálne, a teda ich mať neustále po ruke.

Aplikácia **mojeKarty** slúži na uloženie vernostných alebo iných kariet, ktoré obsahujú číselný kód a názov obchodného reťazca. Po kliknutí sa kód zväčší pre jednoduchšie skenovanie v predajni. Aplikácia funguje offline a je zameraná na rýchlosť a jednoduchosť použitia.

Prehľad podobných aplikácií

1. Stocard [deprecated]

Bola jedna z najznámejších aplikácií na správu vernostných kariet. Ponúkala skenovanie, ukladanie aj odporúčania akcií. Mala prepracované UI, ale vyžadovala registráciu.

2. Apple Wallet / Google Wallet

Natívne aplikácie pre smartfóny, ktoré umožňujú uložiť platobné, vernostné aj boarding pass karty. Menej prispôsobiteľné, nie vždy podporujú klasické číselné kódy z obchodov.

3. FidMe

Aplikácia pre digitálnu vernostnú peňaženku. Podporuje aj akcie a ponuky. Obsahuje reklamy a registráciu.

Zhrnutie: moja aplikácia bude jednoduchšia, bez registrácie, bez reklám, zameraná na čisté a rýchle použitie v predajni.

Analýza navrhovanej aplikácie

Používateľské prípady:

- Pridať novú kartu (názov spoločnosti, číslo karty, meno (voliteľné))
- Skenovať kartu kamerou (automatické doplnenie čísla)
- Zobraziť uložené karty
- Kliknutím zväčšiť čiarový (číselný) kód
- Vymazať alebo upraviť kartu

Role: Používateľ (iba jedna rola)

Mimofunkčné požiadavky:

- Offline fungovanie
- Uchovanie dát len v zariadení
- Minimalistický a rýchly dizajn

Použitie aplikácie – use case (opis slovami)

Po spustení aplikácie sa používateľ nachádza na hlavnej obrazovke, kde vidí zoznam všetkých svojich uložených kariet. Každá karta zobrazuje názov spoločnosti a čiarový kód. Používateľ môže jednoduchým kliknutím na vybranú kartu otvoriť jej detail, kde sa čiarový kód zväčší pre jednoduchšie skenovanie v predajni. V tomto rozšírenom zobrazení môže taktiež pridať poznámku o tom, koľko pri nákupe pomocou tejto karty ušetril a koľko bodov získal.

Z hlavnej obrazovky môže používateľ kliknúť na tlačidlo so symbolom „+“ a pridať novú kartu. Zadá názov spoločnosti, číslo karty, prípadne meno, vyberie si farbu a môže naskenovať čiarový kód pomocou kamery.

Ďalej má možnosť prepnúť sa do menu, kde si vie aktivovať tmavý režim, zapnúť alebo vypnúť sledovanie úspor a bodov, a vykonať akcie ako exportovanie, importovanie alebo resetovanie všetkých kariet. Tieto funkcie umožňujú jednoduchú správu údajov, a to bez potreby pripojenia na internet, pretože všetky dáta sú uložené lokálne v zariadení.

Návrh architektúry aplikácie

Aplikácia mojeKarty je navrhnutá ako jednoduchá, offline a používateľsky priateľská aplikácia na správu vernostných kariet. Použil som moedrné komponenty systému Android a dbal som na to, aby bola architektúra aplikácie prehľadná a jednoducho rozšíriteľná.

Technologická architektúra:

- Single-activity architektúra postavená na Jetpack Compose, čo zjednodušuje správu UI a navigáciu medzi obrazovkami.
- Dáta sú ukladané lokálne cez SharedPreferences vo forme serializovaného JSON reťazca, čím sa zabezpečí offline fungovanie a ochrana súkromia (žiadne údaje sa neposielajú na server).
- Na skenovanie čiarových kódov je použitá open-source knižnica ZXing, ktorá umožňuje spoľahlivé rozpoznávanie bežných formátov (napr. EAN-13).

Dátová štruktúra:

Základnou entitou aplikácie je trieda Card, ktorá reprezentuje jednu vernostnú kartu. Každá karta obsahuje identifikačné informácie, vzhľad, ako aj voliteľné údaje o šetrení:

```
data class Card(
    val id: Int,
    val companyName: String,
    val cardNumber: String,
    val holderName: String? = null,
    val color: String = "#2196F3",
    val saved: Double = 0.0,
    val points: Int = 0
)
```

Návrh vzhľadu obrazoviek

Aplikácia obsahuje:

- **Hlavná obrazovka:** zobrazenie kariet v zozname, horné info boxy, tlačidlo pridať kartu, spodná navigácia
- **Obrazovka pridania karty:** formulár s políčkami (spoločnosť, číslo, meno, farba) + možnosť naskenovať číslo
- **Zväčšený čiarový kód**
- **Menu / nastavenia:** prepínače pre tmavý motív, zapisovanie bodov a šetrenia, export/import kariet, zobrazenie verzie aplikácie a autora

Používané komponenty:

Pri návrhu obrazoviek aplikácie som vybral tieto komponenty Jetpack Compose:

- **TextField** – vstupné polia na zadanie názvu spoločnosti, čísla karty, mena a ušetrených údajov.
- **Button** – tlačidlá "Potvrdiť", "Zrušiť" a iné akčné prvky.
- **IconButton** – tlačidlá s ikonou, ako napr. "pridať kartu" (+) alebo "späť" (←).
- **ModalBottomSheet** – zobrazenie pridávacieho formulára ako spodné vyskakovacie okno.
- **Snackbar** – notifikácie o akciách (napr. karta bola pridaná).
- **Switch** – prepínače v menu (napr. tmavý motív, zapisovanie bodov, šetrenie).
- **TopAppBar** – horný panel s názvom aplikácie a ikonami.
- **BottomNavigation** – spodná navigácia s ikonami domov a menu.
- **AlertDialog** – potvrdenie vymazania alebo resetovania kariet.
- **Row** a **Column** – usporiadanie prvkov v riadkoch a stĺpcoch.
- **Spacer** – medzery medzi komponentmi.

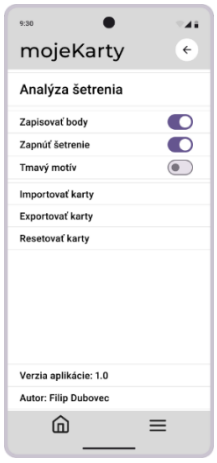
Použité komponenty umožnujú vytvoriť moderné a používateľsky príjemné rozhranie, ktoré bez problémov funguje primárne na dotykových zariadeniach.

Obrázok 1 – Návrh hlavnej obrazovky po spustení aplikácie

Obrázok 2 – Návrh zväčšenia čiarového kódu po kliknutí

Obrázok 3 – Návrh zobrazenia zapisovania údajov po kliknutí

Obrázok 4 – Návrh obrazovky na vkladanie novej karty do aplikácie



Obrázok 5 – Návrh menu, info o aplikácii a pracovanie s kartami

Obrázok 6 – Návrh ukážky tmavej témy

Obrázok 7 – Návrh ukážky tmavej témy pre "Menu"

Skutočný návrh riešenia problému

Aplikácia mojeKarty bola implementovaná ako single-activity Android aplikácia v jazyku Kotlin s využitím frameworku Jatpack Compose. Oproti pôvodnému návrhu sa podarilo implementovať všetky kľúčové plánované funkcionality:

- Pridanie novej karty s číselným kódom a ďalšími údajmi
- Skenovanie karty pomocou fotoaparátu (ZXing)
- Zobrazenie zoznamu uložených kariet
- Zväčšenie čiarového kódu pre jednoduché skenovanie
- Úprava a mazanie existujúcich kariet
- Export a import kariet vo formáte JSON
- Prispôsobovanie témy pre svetlý a tmavý režim
- Možnosť sledovať a zaznamenávať šetrenie a získané body

Celkovo bola aplikácia navrhnutá a implementovaná ako minimalistické, offline riešenie pre správu vernostných kariet s dôrazom na jednoduchosť použitia. Architektúra aplikácie vychádza z princípu jedného zdroja pravdy (**single source of truth**), aby bol stav konzistentný:

- Dáta sú spravované prostredníctvom objektu `StorageManager`, ktorý poskytuje funkcionalitu na načítanie a uloženie zoznamu kariet.
- Stav UI sa v jednotlivých Compose obrazovkách spravuje pomocou lokálneho stavu (remember, rememberSaveable), čo zabezpečuje odolnosť voči prekonfigurovaniu obrazovky.
- Navigácia medzi obrazovkami je riešená cez `Navigation Compose` komponent.

Pri návrhu riešenia bol kladený dôraz na:

- Prehľadnosť používateľského rozhrania
- Rýchlu použiteľnosť bez nutnosti internetu
- Minimálne požiadavky na povolenia a bezpečné uchovávanie dát

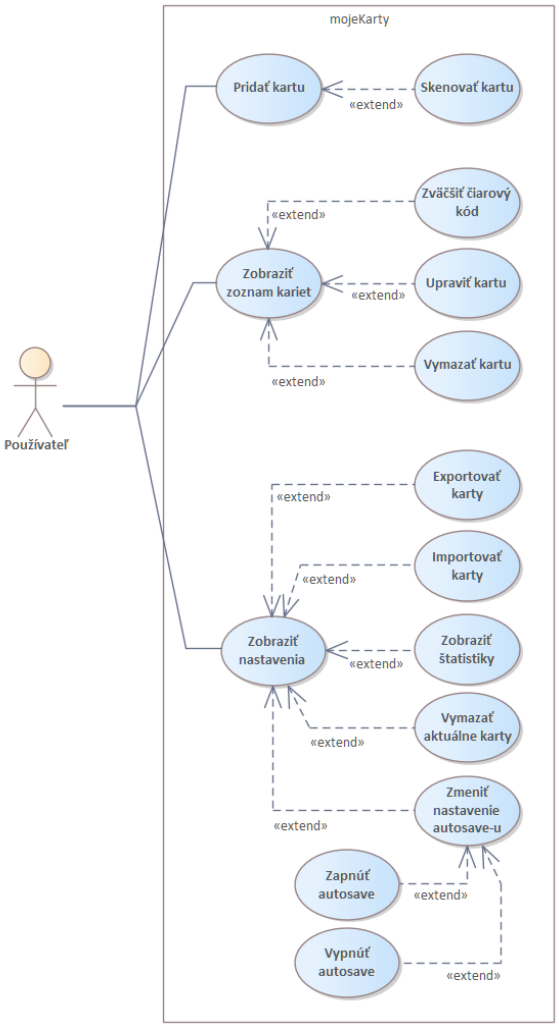
Krátka analýza

Používateľské scenáre (príklady použitia)

1. **Pridanie novej karty**
   - Používateľ klikne na ikonu „+“. Vyplní názov obchodu a číslo karty. Môže použiť tlačidlo „Sken“ pri poli číslo (zapne sa kamera a po úspešnom nasnímaní kód vyplní pole automaticky (1). Nastaví farbu karty z farebnej palety. Nakoniec potvrdí tlačidlom „Pridať“. Aplikácia túto kartu uloží do zoznamu a (ak je zapnutý auto-save) okamžite aj do súboru.
2. **Zobrazenie karty**
   - Na hlavnej obrazovke („cards“ záložka) vidí používateľ zoznam kariet. Každý prvok ukazuje firmu a číslo. Kliknutím na kartu sa otvorí **náhľad** karty na celú obrazovku (`PreviewCardScreen`), kde sa zreteľne vidí čiarový kód a detaily. Náhľad zavrie dotykom. Pri otvorení náhľadu sa zároveň inkrementuje počítadlo „použití“ danej karty.

3. **Úprava alebo vymazanie karty**
   - Na obrazovke zoznamu dlhý stisk na kartu vyvolá dialóg s možnosťami „Upraviť“ a „Vymazať“. Kliknutím na **Upraviť** sa kartová položka predvyplní do formulára `AddCardScreen` (so zmenou názvu na „Upraviť kartu“). Po úpravách sa karta uloží späť do zoznamu. Kliknutím na Vymazať sa položka odstráni zo zoznamu a uloženého JSON súboru, používateľ uvidí krátke upozornenie pomocou SnackBar o zmazaní.
4. **Export / Import kariet**
   - V *Nastaveniach* stlačí používateľ „Exportovať“ a zobrazí sa systémový dialóg na uloženie súboru. Zvolí miesto a potvrdí, čím sa zoznam kariet uloží vo formáte JSON. Import funguje naopak, výber súboru JSON v SAF dialógu načíta dáta a doplní ich do aktuálneho zoznamu. Ak bol import úspešný, zoznam kariet sa prečíta pomocou Gson a preloží do pamäte.
5. **Zobrazenie štatistík**
   - Záložka **Štatistiky** sa nachádza v Nastaveniach. Aplikácia zobrazuje všetky karty a u každej počítadlo, koľko krát bol jeho náhľad otvorený. Používateľ tak vidí najčastejšie používané karty.

Use Case Diagram



Obrázok 8 - Use Case Diagram pre aplikáciu mojeKarty

Návrh riešenia aplikácie

Použité technológie

- **Kotlin**
  - programovací jazyk s rozhraním Android, používa sa pre celú logiku aplikácie. Kotlin ponúka coroutines a mnoho rozšírení pre Android, vďaka čomu je vývoj prehľadnejší (2).
- **Jetpack Compose**
  - moderný UI toolkit od Google pre deklaratívne vytváranie používateľského rozhrania. Compose zjednodušuje tvorbu používateľského rozhrania pre Android (2). Používa Material Design 3 komponenty a témovanie (3).
- **Android Navigation**
  - knižnica pre navigáciu medzi obrazovkami (composable) pomocou `NavController` a `NavHost` (4). Umožňuje definovať trasy („cards“, „add“, „settings“, „stats“) a jednoducho medzi nimi prepínať.
- **Gson**
  - knižnica od Google na serializáciu/deserializáciu objektov do JSON. Používame ju pre ukladanie zoznamu kariet do interného súboru vo formáte JSON (5).
- **ZXing-Android-Embedded**
  - knižnica pre skenovanie (a generovanie) čiarových a QR kódov. V aplikácii slúži na čítanie kódu kamerou (trieda ScanContract). ZXing je open-source knižnica na spracovanie čiarových kódov (6) (7).
- **ColorPicker Compose**
  - knižnica na výber farieb v Compose (dotykom na palete). Požíva sa pri pridávaní karty pre nastavenie farby pozadia karty (8).

Architektúra aplikácie

Aplikácia využíva single activity architektúru. Hlavnú aktivitu (`MainActivity`) deklaruje Jetpack Compose prostredie a obsahuje `NavHost` pre navigáciu medzi obrazovkami (cards/list, add, settings, stats). Podporuje sa *unidirectional data flow* -> stav aplikácie sa drží v jednom zdroji (single source of truth) a prenáža sa do UI komponentov (9) (10). Udalosti zo UI (pridanie, úprava, mazanie kariet) sa posielajú späť a menia dátový model.

V tejto verzii je zoznam kariet spravovaný cez `CardListViewModel`, ktorý drží stav zoznamu kariet ako `StateFlow`. UI ho odoberá pomocou `collectAsState()`, čím sa zabezpečuje konzistentný tok dát. Stav nastavenia `autoSaveEnabled` je aktuálne ešte spravovaný v Compose, v prípade budúceho rozšírenia by bolo vhodné preniesť ho tiež to `ViewModel`-u a využiť plné MVVM podľa odporúčaní (9) (10).

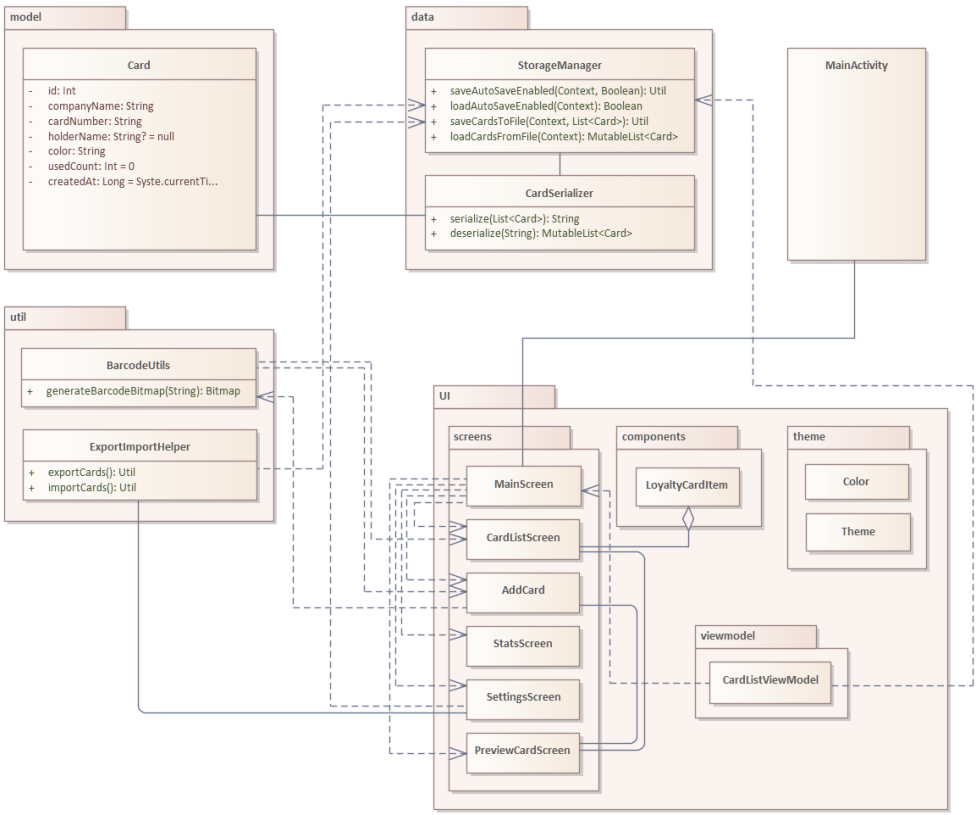
Dátová vrstva je jednoduchá: trieda `StorageManager` spravuje ukladanie a načítanie zoznamu kariet. Ukladá ich do interného úložiska (súbor `cards.json` v adresári `filesDir` (11), ktorý je súkromný pre aplikáciu (11) (12)) a používa JSON formát so serializérom `CardSerializer` (ktorý využíva Gson (5)) Nastavenia (napr. `autoSaveEnabled`) sa ukladajú pomocou `SharedPreferences` (uľahčuje ukladanie jednoduchých kľúč-hodnota párov (13)). V architektúre aplikácie sú oddelené dátové modely, ukladacia logika a používateľské rozhranie. Kód je rozdelený do balíčkov model, data a ui – dátové modely (`Card`), ukladacia logika (`StorageManager`, `CardSerializer`) (9) (10).

Architektúra a triedny model

Aplikácia je implementovaná ako single-activity, a jej architektúra sa skladá hlavne z následovných častí:

- **Dátová trieda Card**
  - Základná entita obsahuje pole `id`, `companyName`, `cardNumber`, voliteľné `holderName`, `color`, `usedCount` a `createdAt`.
- **StorageManager**
  - Objekt zodpovedá za ukladanie a načítavanie dát aplikácie. Používa interný JSON súbor (resp. SharedPreferences) na trvalé uloženie zoznamu kariet a nastavení (napr. auto-save). Na serializáciu sa využíva knižnica Gson (trieda CardSerializer) (5) (13)
- **UI vrstvy (Compose)**
  - Viaceré `@Composable` funkcie implementujú obrazovky: `CardListScreen`, `AddCardScreen`, `PreviewCardScreen`, `SettingsScreen`, `StatsScreen` a pod. Triedy ako `MainActivity` spúšťajú tieto Composable komponenty a spravujú navigáciu (pomocou `NavController`). Pre správu zoznamu kariet je použitá samostatná ViewModel trieda CardListViewModel (14). Ostatné stavy (napríklad nastavenia) sú spravované priamo v Compose (napr. pomocou `remember` a `mutableStateListOf`)
- **Návrh obrazoviek**
  - Hlavná obrazovka obsahuje zoznam kariet a spodnú navigáciu, pridať kartu sa rieši vo formulári, detail v novom okne (Navigation composable) atď.

Diagram tried



Obrázok 9 - Diagram tried pre aplikáciu mojeKarty

Popis použitých komponentov

Card – dátová trieda reprezentujúca vernostnú kartu. Obsahuje vlastnosti: `id` (UUID), `companyName` (názov obchodu), `cardNumber` (číslo karty), `holderName` (meno držiteľa, nepovinné), `color` (hex farba) a štatistiky (`usedCount`, `createdAt`). Každý objekt sa serializuje do JSON pomocou Gson (5).

StorageManager – objekt so statickými funkciami na prácu s úložiskom. Metódy `loadCardsFromFile()` a `saveCardsToFile()` zapisujú/čítajú celý zoznam kariet do interného súboru (cesta `context.filesDir`). Používa `CardSerializer` (so Gson) na konverziu zoznamu na JSON. Tiež spravuje `SharedPreferences` pre nastavenie `autoSaveEnabled`. (Oficiálne dokumenty hovoria, že pre malé množstvo nastavení je `SharedPreferences` vhodný nástroj (13) (15).)

CardSerializer – pomocná trieda, ktorá convertuje `List<Card>` na JSON a späť pomocou Gson knižnice. Zabezpečuje, aby JSON bol prehľadný (indentovaný). Údaje sa tak bezpečne ukladajú na disk.

BarcodeUtils – utilitná trieda na generovanie čiarového kódu (formát `CODE_128`) z textu (čísla karty). Používa `MultiFormatWriter` z knižnice ZXing: táto trieda dokáže zakódovať text do bitovej mapy čiarového kódu (16). Vygenerovaný kód sa potom zobrazuje na obrazovke pomocou `Bitmap` (7).

MainActivity – vstupná aktivita aplikácie, ktorá nastaví Compose UI. Vytvorí navigačný ovládač (`NavController`) a prepne tému (`Theme.MojeKarty`). Rieši zároveň uzamykanie orientácie (napr. obrazovku „pridať kartu“ alebo „štatistiky“ má uzamknutý portrét, inak povolený aut. rotácia).

MainScreen (composable) – hlavné rozhranie aplikácie po spustení. Obsahuje `Scaffold` s vrchnou lištou (názov aplikácie) a spodnou navigačnou lištou (`BottomAppBar` (17)) s ikonami pre jednotlivé obrazovky (zoznam kariet, štatistiky, pridať novú kartu, nastavenia). V strede zobrazuje príslušný `NavHost` pre vybranú záložku. Zdôrazňujeme, že navigácia v Compose sa realizuje pomocou `NavHost` a `NavController` (4).

CardListScreen – displej so zoznamom všetkých kariet. Zoznam je komponenta `LazyColumn` a každý prvok zobrazuje `LoyaltyCardItem`. Krátky klik na kartu otvorí náhľad (`PreviewCardScreen`), dlhý klik vyvolá dialóg (ponuku akcií): `Upraviť` alebo `Vymazať kartu`. Týmto sa implementuje úprava a mazanie. Zoznam kariet je držaný na jednom mieste v pamäti a akcie v dialógu (9) (10).

LoyaltyCardItem (composable) – komponenta zobrazujúca jednu kartu v zozname. Používa `Card` z `Material3`, nastaví pozadie podľa `card.color`, vypíše názov firmy, meno držiteľa a posledné 4 číslice karty. Taktiež zobrazuje text „Použitá X-krát“ s počítadlom. V spodnej časti vykreslí čiarový kód pomocou `Canvas` a bitmapy z `BarcodeUtils` (16).

AddCardScreen – obrazovka pre pridanie alebo editáciu karty. Obsahuje textové polia (`OutlinedTextField`) pre názov firmy, číslo karty a meno držiteľa. Polia sú obohatené o tlačidlo na skenovanie (ikonka skenera, spustenie `ScanContract` z ZXing) – po úspešnom skene sa číslo karty automaticky zapíše (1). Ďalej obsahuje farebný výber: používateľ vyberá z palety farebnej schémy (knižnica `ColorPicker` Compose (8)). Pod formulárom sa zobrazuje ukážka karty v komponente `PreviewCardScreen`.

PreviewCardScreen – jednoduchá kompozičná obrazovka, ktorá zobrazí vybranú kartu vo väčšom zobrazení (karta je stredovým prvkom). Umožňuje zblízka vidieť čiarový kód a ostatné údaje. Kliknutie kdekoľvek zatvorí tento „plnú obrazovku“.

CardListViewModel – trieda `ViewModel`, ktorá spravuje stav zoznamu kariet v aplikácii. Zoznam kariet drží ako `MutableStateFlow`, ktorý sledujú UI komponenty. `ViewModel` implementuje *unidirecitonal data flow* – akcie z UI (pridanie, zmena, vymazanie karty) aktualizujú stav kariet v cards. Pri zapnutom autoSaveEnabled sa zmeny automaticky ukladajú (14).

SettingsScreen – obrazovka nastavení. Obsahuje prepínač (`Switch`) pre možnosť automatické ukladanie. Ak je automatické ukladanie zapnutý, každá zmena zoznamu kariet sa hneď zapíše do súboru. Ak je vypnutý, objaví sa tlačidlo „`Uložiť zmeny`“, ktorým používateľ manuálne uloží súbor. Ďalej má tlačidlá Exportovať (otvorí dialóg pre vytvorenie súboru JSON v inom úložisku) a Importovať (otvorí dialóg výber súboru). Implementácia využíva `rememberLauncherForActivityResult` s kontraktmi `CreateDocument` a `OpenDocument` (18) (bez potreby explicitného povolenia (19)). Nakoniec tlačidlo „`Vymazať všetky karty`“ s potvrdením – zmaže všetky uložené karty zo zoznamu.

StatsScreen – štatistiky používania. Zobrazuje zoznam kariet s počtom „kliknutí“ (zobrazení náhľadu) pre každú z nich. Využíva stĺpcový pohľad (`Column`) rozdelený do riadkov: vľavo názov a dátum pridania karty, vpravo počet využití.

ExportImportHelper – pomocný `Composable`, ktorý vytvára dve `ActivityResultLauncher` – jeden pre `ACTION_CREATE_DOCUMENT` (export JSON) a druhý pre `ACTION_OPEN_DOCUMENT` (import JSON). Vracia lambda funkcie `onExport()` a `onImport(...)` na volebnú inicializáciu akcií. Pomocou nich SettingsScreen navolá zapísanie/odčítanie JSON cez systémový súborový dialóg (20). Dokumentácia SAF opisuje, že tento postup funguje bez požiadania extra povolení, lebo súbor vyberá užívateľ (19) (21).

Popis implementácie podľa hodnotených častí

Táto kapitola sumarizuje, akým spôsobom aplikácia mojeKarty spĺňa jednotlivé hodnotené kritériá.

Zložitosť aplikácie (max. 40 bodov)

Všeobecné požiadavky

- **Otočenie displeja:** Aplikácia správne reaguje na otočenie displeja. Dôležité stavy (formulár, zoznam kariet, zobrazovanie čiarového kódu) sú spravované pomocou `rememberSaveable` a `mutableStateListOf`.
- **Zdroje:** Všetky texty, obrázky a farby sú uložené v strings.xml alebo ako resource.

Obrazovky

- Hlavná obrazovka (zoznam kariet)
- Pridane / editovanie karty
- Zväčšený čiarový kód (náhľad)
- Nastavenia
- Štatistiky používania

Využité AndroidX komponentov

- ViewModel: použitý (CardListViewModel pre správu zoznamu kariet)
- Navigation použité (Navigation Compose)
- Lifecycle nie je použitý
- Paging, Room, WorkManager nie je použitý

Widget

- Nepoužité.

Notifikácie

- Nepoužité.

Externé frameworky / knižnice

- ZXing (čiarové kódy)
- Gson (serializácia)
- ColorPicker Compose

Service / Broadcast receiver / Content provider

- Nepoužité.

Použitie senzora

- Použitie fotoaparátu cez ZXing contract.

Sieťová komunikácia

- Nepoužitá (offline aplikácia).

Použitá programátorská technika (max. 30 bodov)

Práca s Git serverom

- Git bol používaný, projekt má štrukturované commity (viď repozitár).
- Commity majú správne názvy, časové rozloženie práce snáď značné.

Návrh aplikácie, architektúra

- Oddelenie dátové logiky (`StorageManager`, `CardSerializer`, `CardListViewModel`) od UI vrstvy (Compose obrazovky).
- Použitá je jednoduchá *unidirectional data flow* architektúra (9) (10) (22) (23), stav zoznamu kariet je spracovaný prostredníctvom CardListViewModel.
- Nie je implementovaná plná clean architecture, ostatné stavy (napr. nastavenia) sú spravované priamo v Compose. MVVM čiastočne využité pre správu zoznamu kariet.

Dodržiavanie coding standards

- Identifikátory a názvy sú správne.
- Výnimky sú ošetrené tam, kde je to potrebné.
- Kód je rozčlenený do balíčkov (model, data, ui).
- Duplicitný kód by nemal byť prítomný.

Správne uplatnenie OOP

- Dáta sú reprezentované cez `data class Card`.
- `StorageManager` je singleton (object).
- UI vrstvy používajú Compose `@Composable` komponenty.

Možné rozšírenia a známe limity

- **Limitovaný dátový model**
  - V aplikácií sa všetky karty ukladajú len lokálne (bez cloudových služieb). V budúcnosti by sa dal pridať sync cez Firebase/Cloud, aby karty boli dostupné na viacerých zariadeniach. Tiež je možné pre komplexnejšie dáta použiť Room databázu namiesto ručného JSON súboru.
- **Optimalizácia úložiska**
  - Momentálne používame `SharedPreferences` pre nastavenia a textový JSON súbor pre karty. Ako odporúčanie ANDROID dev sa dá namiesto SharedPreferences použiť DataStore (Kotlin korutiny/Flow) (13) (24), ktorý je modernejší a spoľahlivejší. Pri veľkom počte kariet by bolo lepšie riešenie s databázou.
- **Robustnosť a vzory**
  - Aktuálne je `ViewModel` použitý pre správu zoznamu kariet, ostatné stavy nie -> v budúcnosti by bolo vhodné doplniť MVVM vrstvu (napríklad SettingsViewModel) a viac oddeliť logiku aplikácie od používateľského rozhrania (9) (10). Ďalej chýbajú jednotkové testy.
- **Používateľské vylepšenia**
  - Momentálne nie je možnosť filtrovať či vyhľadávať karty, chýba lokalizácia do iných jazykov. Budúce rozšírenia by mohli priniesť widgety zobrazujúce obľúbené karty a notifikácie, prípadne integráciu s QR kódmi okrem 1D čiarových kódov.
- **Výkon**
  - Zápis do súboru sa deje na hlavnom vlákne. Ak by kariet bolo veľmi veľa, mohlo by to spomaliť UI. Rozšírením by bolo ukladanie asynchrónne (cez coroutines alebo WorkManager).
- **Bezpečnosť**
  - Aplikácia momentálne nešifruje údaje. Citlivé informácie ako vernostné čísla by mohli byť chránené.

Bibliografia

1. **Künneth, Thomas.** Integrating ZXing Android Embedded in a Compose app. *DEV.to.* [Online] 11 2021. https://dev.to/tkuenneth/integrating-zxing-android-embedded-in-a-compose-app-5ela.

2. **Developers, Android.** Jetpack Compose UI toolkit for Android. *Android Developers.* [Online] https://developer.android.com/jetpack/compose.

3. —. Compose Material 3. *Android Developers.* [Online] https://developer.android.com/jetpack/androidx/releases/compose-material3.

4. —. Navigation with Jetpack Compose. *Android Developers.* [Online] https://developer.android.com/develop/ui/compose/navigation.

5. **Tutorialspoint.** Convert Java Object to JSON using Gson Library. *TutorialsPoint .* [Online] https://www.tutorialspoint.com/how-to-convert-java-object-to-json-using-gson-library.

6. **(GitHub), ZXing/zxing.** ZXing ("Zebra Crossing") barcode processing library. *GitHub.* [Online] https://github.com/zxing/zxing.

7. **Scott.** Generate QR Code using ZXing Library. *Medium.* [Online] https://medium.com/@copy2sim/generate-qr-code-using-zxing-library-751ca3d76792.

8. **skydoves.** colorpicker-compose (Kotlin Multiplatform Color Picker library). *GitHub.* [Online] https://github.com/skydoves/colorpicker-compose.

9. **Developers, Android.** Guide to app architecture. *Android Developers.* [Online] https://developer.android.com/topic/architecture.

10. —. Architecting your Compose UI. *Android Developers.* [Online] https://developer.android.com/develop/ui/compose/architecture.

11. —. Access app-specific files. *Android Developers.* [Online] https://developer.android.com/training/data-storage/app-specific.

12. **GeeksforGeeks.** Internal Storage in Android with Example. *GeeksforGeeks.* [Online] 17. 9 2024. https://www.geeksforgeeks.org/internal-storage-in-android-with-example/.

13. **Developers, Android.** Save simple data with SharedPreferences. *Android Developers.* [Online] https://developer.android.com/training/data-storage/shared-preferences.

14. —. ViewModel overview (App architecture). *Android Developers.* [Online] https://developer.android.com/topic/libraries/architecture/viewmodel.

15. **GeeksforGeeks.** Shared Preferences in Android with Examples. *GeeksforGeeks.* [Online] https://www.geeksforgeeks.org/shared-preferences-in-android-with-examples/.

16. **Geeksforgeeks.** How to Generate Barcode in Android? *Geeksforgeeks.* [Online] 28. 4 2025. https://www.geeksforgeeks.org/how-to-generate-barcode-in-android/.

17. **Yadav, Santosh.** Bottom Navigation Bar in Jetpack Compose. *Medium.* [Online] https://medium.com/@santosh_yadav321/bottom-navigation-bar-in-jetpack-compose-5b3c5f2cea9b.

18. **Styl, Alex.** Ask other apps for photos, files and more using ActivityResultContracts. *Compose Tutorials.* [Online] https://composables.com/jetpack-compose-tutorials/activityresultcontract.

19. **Developers, Android.** Access documents and other files from shared storage. *Android Developers.* [Online] https://developer.android.com/training/data-storage/shared/documents-files.

20. —. Open files using the Storage Access Framework. *Android Developers.* [Online] https://developer.android.com/guide/topics/providers/document-provider.

21. **jroddev.** Reading and Writing files in Android 10+ Scoped Storage. *jroddev.* [Online] https://blog.jroddev.com/reading-and-writing-files-in-android-10/.

22. **Developers, Android.** Android Developers. *Android Developers.* [Online] https://developer.android.com/jetpack/compose/architecture.

23. **Waghmare, Mayur.** Jetpack Compose: Best Practices. *Medium.* [Online] 9 2024. https://medium.com/mobile-innovation-network/jetpack-compose-best-practices-7a25e6c182ac.

24. **Developers, Android.** App architecture – Data layer: DataStore. *Android Developers.* [Online] https://developer.android.com/topic/libraries/architecture/datastore.

25. **GeeksforGeeks.** Barcode Generation in Android using ZXing. *GeeksforGeeks.* [Online] https://www.geeksforgeeks.org/generate-barcode-in-android-using-zxing/.