Fakulta Riadenia a Informatiky

Žilinská univerzita v Žiline

Vývoj aplikácií pre mobilné zariadenia

Semestrálna práca – hra „Korona Slayer“

Jozef Mikulík

5ZYI21

1. Špecifikácia zadania, definovanie problému

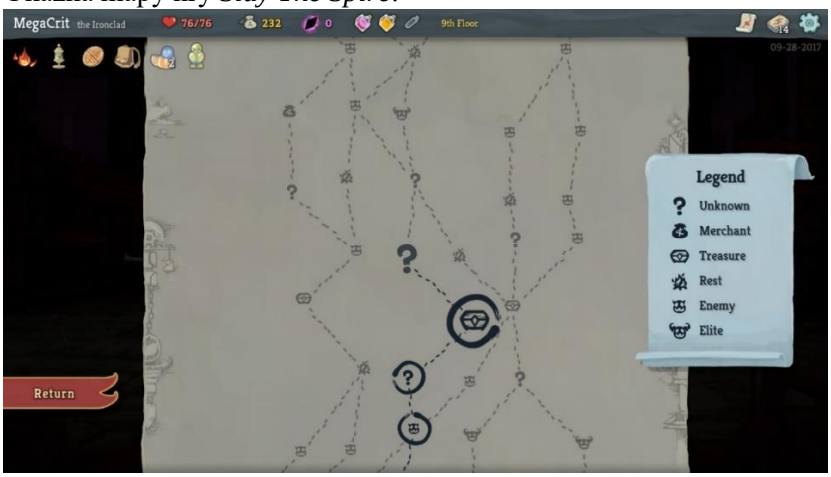
Zadaním mojej semestrálnej práce je hra určená predovšetkým pre mobilné zariadenia. Hra je realizovaná spôsobom ťahov, počas ktorých môže hráč využívať náhodne generované karty na porazenie nepriateľov (chorôb). Hráč vyhráva v prípade, že porazí všetkých nepriateľov rozmiestnených do 10 levelov. Naopak hráč môže prehrať v prípade, že nepriateľ počas jeho ťahu spôsobí hráčovi zranenie kde hráčovi klesnú životy na 0.

2. Prehľad dostupných aplikácií podobného zamerania

2.1 Slay The Spire

Inšpiráciou pre túto mobilnú aplikáciu, bola predovšetkým počítačová hra *Slay The Spire*, ktorá je avšak dostupná aj pre mobilné zariadenia. Cieľom tejto hry je prejsť cez rôzne levely plné nepriateľov za pomoci kariet, ktoré majú rôzne účinky. Celková hra je tvorená z rôznych úrovní, ktoré sú tvorené väčším počtom levelov a cieľom je poraziť „finálneho bossa“ danej úrovne. Hra nie je tvorená iba levelmi s nepriateľmi ale aj z levelov v ktorých si môže hráč zakúpiť nové karty, navýšiť životy a podobne. Boj v určitom levely prebieha ťahovým spôsobom. Hráčovi sú na začiatku jeho ťahu vygenerované karty z jeho balíčku, ktoré môže využiť počas svojho ťahu po použití určitého počtu energie. Na konci ťahu sú zostávajúce karty zahodené a na rade je nepriateľ. V prípade, že hráč vyprázdni celý svoj ťahací balíček sú všetky karty z odhadzovacieho balíčku opäť vrátené do ťahacieho balíčku a boj pokračuje obdobným spôsobom až kým nezomrie nepriateľ alebo daný hráč. V prípade smrti hráča je hra ukončená a užívateľovi nezostáva nič iné iba začať hru znova. V prípade smrti nepriateľa je hráč odmenený novou kartou a môže pokračovať do ďalšieho levelu.

Ukážka mapy hry *Slay The Spire*:



Ukážka boja v hre *Slay The Spire*:

2.2 Dungeon Tales

Ďalšou veľmi podobnou hrou je aj hra *Dungeon Tales*. Kde hráč taktiež prechádza cez rôzne levely v ktorých sa nachádzajú bežný nepriatelia, bossovia, poklady a levely určené na regeneráciu. Podobne je v hre riešený aj boj, ktorý je taktiež realizovaný pomocou kariet, many a obrany/bloku. Hlavným rozdielom medzi týmito hrami je asi grafické prepracovanie kariet a jednotlivých levelov. Pri čom v hre *Dungeon Tales* sa vývojári snažili o väčší 3D zážitok ako v hre *Slay The Spire*.

Ukážka mapy hry *Dungeon Tales*:

Ukážka boja hry *Dungeon Tales:*

3. Analýza navrhovanej aplikácie

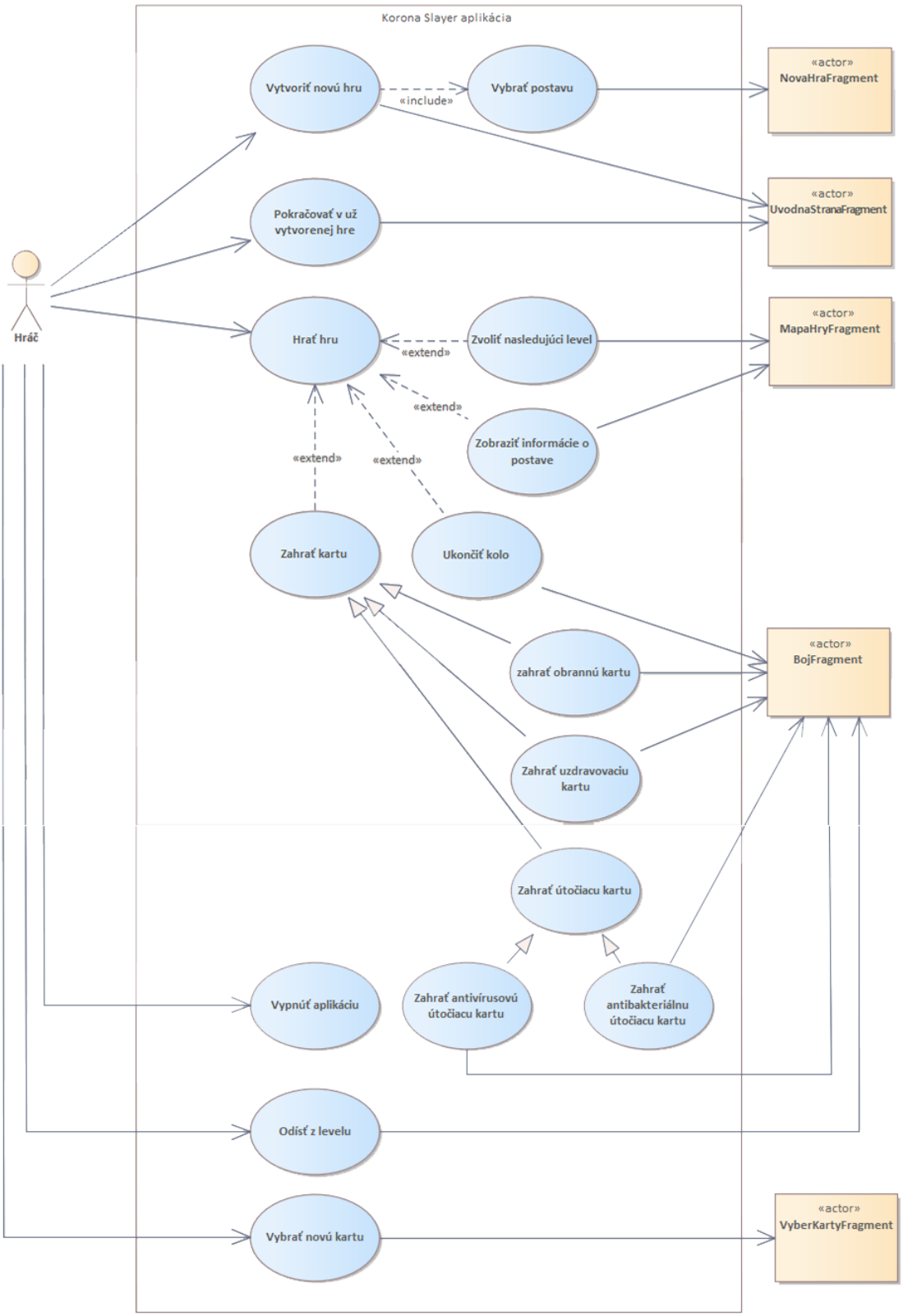
3.1 Priebeh a cieľ hry

Podobne ako Slay the Spire je hra určená pre jedného hráča avšak patrným rozdielom bude, že hra nie je rozdelená do viacerých úrovní preto cieľom hry je poraziť jediného „bossa“ Koronu. Avšak podobne ako v predošlej hre bude musieť hráč počas cesty poraziť aj iné bežnejšie choroby, ktoré sú rozmiestnené do rôznych levelov na základe obtiažnosti. Boj je realizovaný ťahovou formou pomocou kariet, kde je a začiatku kola hráčovi ponúknutých 5 kariet rôznych schopností. Karty sú rozdelené do 3 základných skupín: blokovacie karty (pridávajú obranu proti ďalšiemu útoku nepriateľa), uzdravovacie karty (pridávajú životy avšak nie je možné prekročiť max. životy danej postavy) a útočiace karty (ktoré sú ďalej delené na antibakteriálne alebo antivírusové pričom jednotlivé typy sú účinnejšie proti určitému typu choroby). Po skončení kola hráča vykoná určitú akciu nepriateľ. V prípade výhry hráča sú mu ponúknuté 3 nové karty z ktorých si môže 1 zvoliť, pridať ju do svojho balíčka kariet a pokračovať do ďalšieho levelu. Naopak v prípade smrti hráča je hráčovi ukázané, ktorá choroba ho porazila a môže začať novú hru.

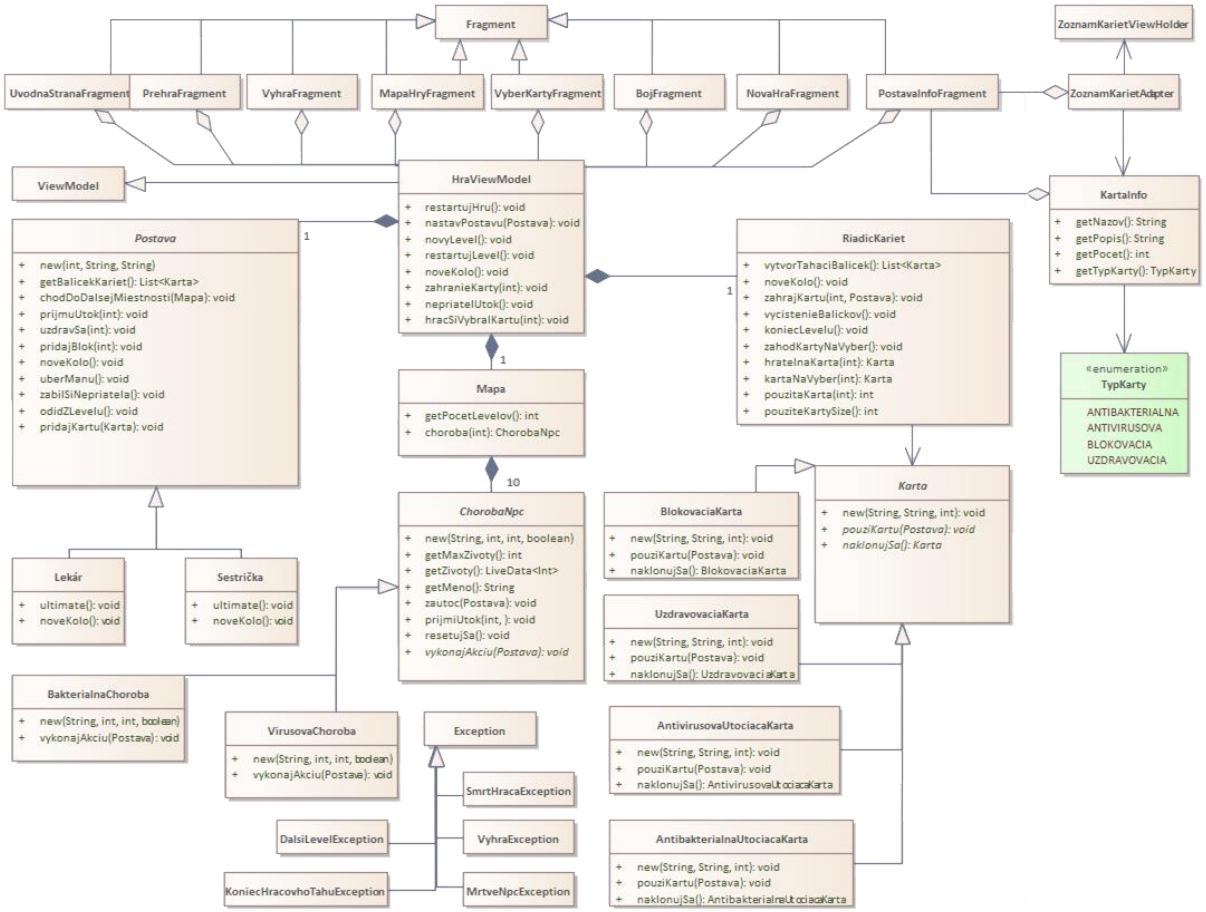
3.2 Ovládanie

Po spustení aplikácie je hráčovi zobrazená úvodná obrazovka, kde si môže hráč zvoliť vytvoriť novú hru a neskôr v prípade, že už má hru vytvorenú je na danej obrazovke zobrazené aj tlačidlo pre pokračovanie v hre. V prípade, že sa hráč rozhodne vytvoriť novú hru aj v prípade, že už má hru vytvorenú bude táto hra zrušená a vyresetovaná. Pri vytváraní novej hry sú hráčovi zobrazené 2 postavy za ktoré môže hráč hrať. O postave sú taktiež vypísané základné informácie ako sú maximálne životy, mana a pasívna schopnosť danej postavy. Po vybratí postavy je hráč presmerovaný na obrazovku mapy, ktorá obsahuje tlačidlá predstavujúce jednotlivé levely. Tlačidlá sú rozlíšené farbou podľa toho či hráč level už prešiel/neprešiel a aký level ho momentálne čaká. Na jednotlivých tlačidlách sú názvy chorôb v daných leveloch a obrázky reprezentujúce typ nepriateľa (bakteriálna alebo vírusová choroba). Na obrazovke je taktiež tlačidlo schopné presmerovať hráča na fragment zobrazujúci základné informácie o jeho postave a o aktuálnom balíčku kariet. Po vybratí levelu je hráč presmerovaný na fragment, kde prebieha základ celej hry. Hráč začína každé kolo s 5-timi náhodne vygenerovanými kartami z ťahacieho balíčka a max. manou špecifickú pre danú postavu. Hráčove kolo končí v prípade minutia many alebo predčasne stlačením tlačidla „Ukončiť kolo“. Po skončení hráčovho kola je na rade nepriateľ, ktorý zvyčajne spôsobí hráčovi určité zranenie na základe jeho obtiažnosti. V hre ako už bolo spomenuté sú 2 typy nepriateľov. Hlavným rozdielom medzi nimi je ich správanie pokiaľ ich životy klesnú pod 30% maximálnych životov. Jeden typ má zvýšený útok o 20%, druhý je schopný uzdraviť sa. V prípade že hráčovi počas nepriateľovho ťahu klesnú životy na 0 hráč prehráva a je presmerovaný na fragment zobrazujúci túto skutočnosť. Z tohto fragmentu sa môže hráč dostať buď na úvodnú obrazovku alebo priamo k výberu novej postavy. V prípade, že hráč ďalej žije sa celý proces opakuje. Ak naopak je víťazom hráč a nenachádza sa v poslednom levely je teda presmerovaný na fragment, kde si môže vybrať jednu z 3 náhodne vygenerovaných kariet. Po výbere karty je karta pridaná do jeho základného balíčka a hráč je opäť presmerovaný na fragment mapy. V prípade, že porazil posledného nepriateľa je hráč presmerovaný na fragment zobrazujúci výhru hráča odkiaľ je možné prejsť podobne ako pri prehre buď na úvodnú obrazovku alebo na fragment výberu novej postavy.

3.3 Diagram prípadov použitia



4. Návrh architektúry aplikácie



HraViewModel – Slúži na riadenie hry a uchovávanie stavov jednotlivých prvkov.

Mapa – Slúži pre vygenerovanie a uchovávanie zoznamu jednotlivých NPC. Počas hry je trieda využívaná na vracanie inštancií NPC pre požadovaný level.

ChorobaNpc – Abstraktná trieda slúžiaca pre uchovávanie a vracanie hodnôt základných atribútov jednotlivých chorôb. Taktiež slúži pre implementáciu základnej logiky NPC.

BakterialnaChoroba/VirusovaChoroba – Triedy dediace z triedy ChorobaNpc. Rozdiel medzi danými triedami je v implementácií logiky pre metóda *vykonajAkciu()*.

Postava – Abstraktná trieda slúžiaca pre uchovávanie a vracanie hodnôt atribútov pre jednotlivé typy postáv. Okrem iného implementuje taktiež rovnaké správanie pre jednotlivé typy postáv.

Lekar/Sestricka – Triedy dediace z triedy Postava. Rozdiel v správaní týchto tried je v metóde *noveKolo()* kde sú implementované pasívne schopnosti daných postáv.

RiadicKariet – Trieda slúžiaca na celkovú implementáciu hracej logiky s kartami. Slúži pre generovanie kariet pre hráča v kolách, riadenie zahadzovacieho a ťahacieho balíčka kariet a generovanie kariet na konci víťazného kola hráča. Počas kôl slúži taktiež na využívanie schopností jednotlivých kariet.

Karta – Abstraktná trieda slúžiace pre uchovávanie, vracanie hodnôt atribútov a implementáciu základnej logiky všetkých typov kariet.

BlokovaciaKarta – Slúži pre navýšenie obrany(bloku) hráča na ďalšie kolo nepriateľa.

UzdravovaciaKarta – Slúži pre navýšenie životov postavy až do max. počtu životov danej postavy.

AntibakterialnaUtociacaKarta – Slúži na útok proti nepriateľom. Navýšený útok proti chorobám bakteriálneho typu.

AntivirusovaUtociacaKarta - Slúži na útok proti nepriateľom. Navýšený útok proti chorobám vírusového typu.

UvodnaObrazovkaFragment – Fragment úvodnej obrazovky, kde si môže hráč zvoliť pokračovať vo vytvorenej hre ak taká existuje alebo vytvoriť novú hru.

NovaHraFragment – Fragment slúžiaci na vytvorenie novej hry. Hráčovi sú zobrazené rôzne typy postáv z ktorých si môže vybrať. Každá z týchto postáv má rôzne max. životy, energiu a pasívnu schopnosť.

MapaHryFragment – Fragment zobrazujúci jednotlivé levely, typy chorôb v daných leveloch aj s ich názvami. Červenou farbou sú zvýraznené už porazené choroby, zelenou nasledujúci možný level, bielou levely, ktoré budú dostupné v budúcnosti. V pravom hornom rohu sa nachádza ikonka, pomocou ktorej sa dokáže hráč dostať na fragment zobrazujúci informácie o jeho postave a balíčku kariet.

PostavaInfoFragment – Fragment zobrazujúci základné informácie o postave a balíčku kariet.

BojFrament – Fragment zobrazujúci práve prebiehajúci boj. V hornej časti sú zobrazené informácie o počte životov a energie hráča. V strednej časti je zobrazené meno, obrázok a aktuálne životy nepriateľa. V spodnej časti sú zobrazené hrateľné karty a tlačidlo na ukončenie ťahu.

VyberKartyFragment – Fragment slúžiaci hráčovi na výber novej karty po úspešne vyhratom levely. Na fragmente je zobrazený informatívny text a 3 tlačidla s názvami náhodne vygenerovaných kariet zo zoznamu vzorov kariet.

PrehraFragment – Fragment slúžiaci na informovanie hráča o jeho prehre. Na fragmente je zobrazený informatívny text aj s názvom nepriateľa, ktorý hráča porazil a 2 tlačidlá slúžiace na presmerovanie hráča buď na úvodnú obrazovku alebo na fragment NovaHra.

VyhraFragment – Fragment slúžiaci na informovanie hráča o jeho výhre. Na fragmente je zobrazený informatívny text a podobne ako pri fragmente určeného pri prehre 2 tlačidlá odkazujúce na úvodnú stranu a na fragment novej hry

TypKarty – Enum trieda predstavujúce jednotlivé typy kariet.

KartaInfo – Data trieda slúžiace pre uchovávanie a vracanie základných informácií o určitej karte, o jej počte v hráčovom balíčku a o type karty.

ZoznamKarietAdapter – Trieda slúžiaca pre zobrazenie zoznamu kariet tvoreného pomocou inštancií triedy *KartaInfo* v RecyclerView.

5. Popis Implementácie

5.1 Data Binding

Data binding som využil skoro na každom fragmente spolupracujúcom s ViewModelom. Využil som to predovšetkým pre zjednodušenie hľadania referncií na jednotlivé „views“ a zároveň pre definovanie dát v samotnom XML. Ako dáta som v XML využíval predovšetkým definovanie hodnoty ViewModelu vďaka čomu som vedel získať potrebné texty ako boli názvy postáv, chorôb, životy hráča, nepriateľa a podobne.

Ukážka definovania Data bindingu vo fragmente:

```
private lateinit var binding: FragmentBojBinding
...
override fun onCreateView(...) {
...
binding = DataBindingUtil.inflate(inflater, R.layout.fragment_boj, container,false)
binding.hraViewModel = viewModel
...
}
```

Ukážka využitie Data bindingu v XML:

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="hraViewModel"
            type="fri.uniza.sk.mikulik6.koronaSlayer.HraViewModel" />
    </data>

    <TextView
        ...
        android:text="@{@string/manaHraca(hraViewModel.hrac.mana)}"
        .../>
</layout>
```

5.2 LiveData

LiveData som využil predovšetkým v triedach spolupracujúcich s BojFragment, kde je potrebné často upravovať hodnoty many, bloku, životov hráča a nepriateľa.

Ukážka definovania LiveData v triede Postava:

```
abstract class Postava(val meno: String, val pasivnaSchopnost: String, pZdravie: Int) {
...
    private val balicekKariet   = mutableListOf<Karta>()
    val maxZdravie: Int         = pZdravie
    val maxMana: Int            = 3

    private var _zdravie                = MutableLiveData(maxZdravie)
    val zdravie: LiveData<Int>
        get() = _zdravie
    private var _mana                   = MutableLiveData(maxMana)
    val mana: LiveData<Int>
        get() = _mana
    private var _blok                   = MutableLiveData(0)
    val blok: LiveData<Int>
        get() = _blok
...
}
```

Ukážka definovania lifeCycleOwner v triede BojFragment:

```
class BojFragment : Fragment(){

    override fun onCreateView(...) {
        binding = DataBindingUtil.inflate(inflater, R.layout.fragment_boj, container,false)
        binding.lifecycleOwner = viewLifecycleOwner
        ...}
...
}
```

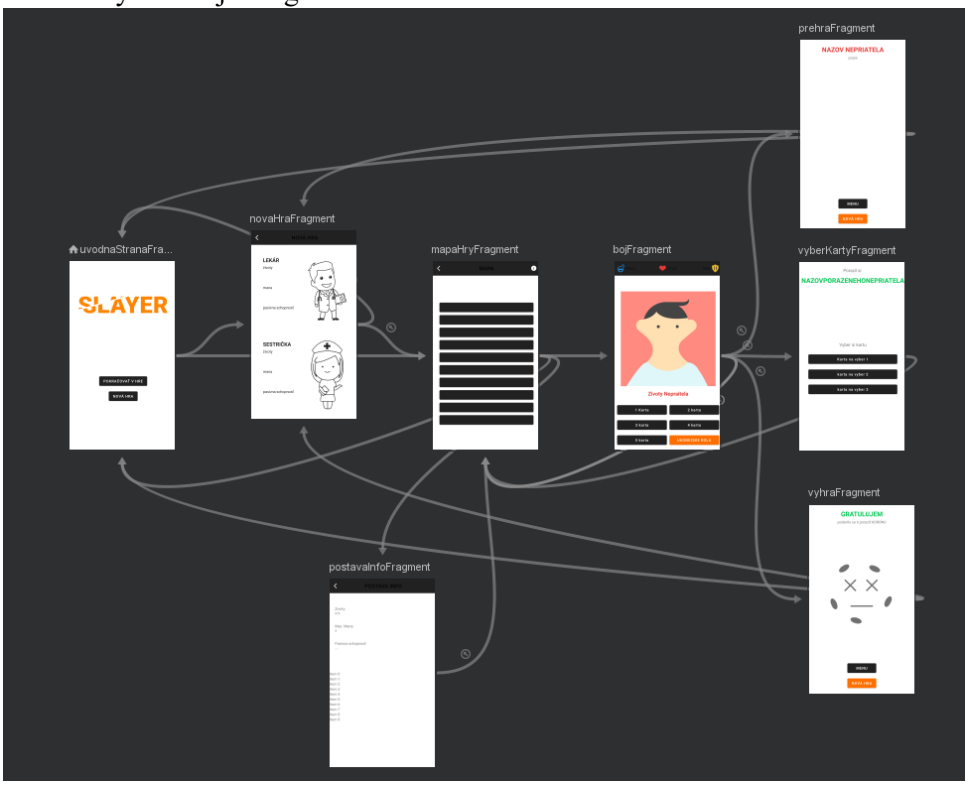
Ukážka využitie LiveData vo fragment_boj.xml:

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="hraViewModel"
            type="fri.uniza.sk.mikulik6.koronaSlayer.HraViewModel" />
    </data>

    <TextView
        ...
        android:text="@{@string/manaHraca(hraViewModel.hrac.mana)}"
        .../>
</layout>
```

5.3 Navigation

Navigation som využil predovšetkým pre navigovanie medzi fragmentami aplikácie a pre definovanie samotných prechodov.

Ukážka vytvorenej navigácie:



Ukážka použitia prechodu medzi fragmentami:

```
class UvodnaStranaFragment : Fragment() {
...
    override fun onCreateView(...) {
        ...
        binding.pokracovatVHreTlacidlo.setOnClickListener {
         findNavController().navigate(R.id.action_uvodnaStranaFragment_to_mapaHryFragment)
        }
        ...
    }
...
}
```

5.4 ViewModel

ViewModel som v mojej aplikácií využil ako takzvaný „*SharedViewModel*“. Túto techniku som zvolil z dôvodu, že viacero fragmentov využíva tie isté objekty prípadne aj rovnaké alebo veľmi podobné metódy ViewModelu. V prípade využitia štandardu a vytvorenia ViewModelu pre každý fragment samostatne by sa jednoznačne zvýšila duplicita kódu a náročnosť kvôli posielaniu rovnakých objektov medzi jednotlivými fragmentami.

Ukážka inicializácie ViewModelu v *UvodnaStranaFragment*:

```
class UvodnaStranaFragment : Fragment() {
...
    private val viewModel: HraViewModel by activityViewModels()
...
}
```

5.5 RecyclerView

RecyclerView som využil na fragmente *PostavaInfoFragment* na zobrazenie zoznamu kariet v hráčovom balíčku. Pre ich zobrazenie využívam triedy *ZoznamKarietAdapter* slúžiaci pre vytvorenie a nastavenie textov a farieb jednotlivých „*itemov*“. Taktiež využívam triedu *ZoznamKarietViewHolder,* ktorá uchováva referencie na jednotlivé pohľady každého „*itemu*“.

5.6 LifeCycles

LifeCycles som využil na každom fragmente kde som v metóde onCreateView definoval jednotlivé onClickListenery, nastavoval farby a texty tlačidiel, prípadne kontroloval či je uložená inštancia stavu daného pohľadu podľa čoho som vykonával v tejto metóde rozdielne akcie.

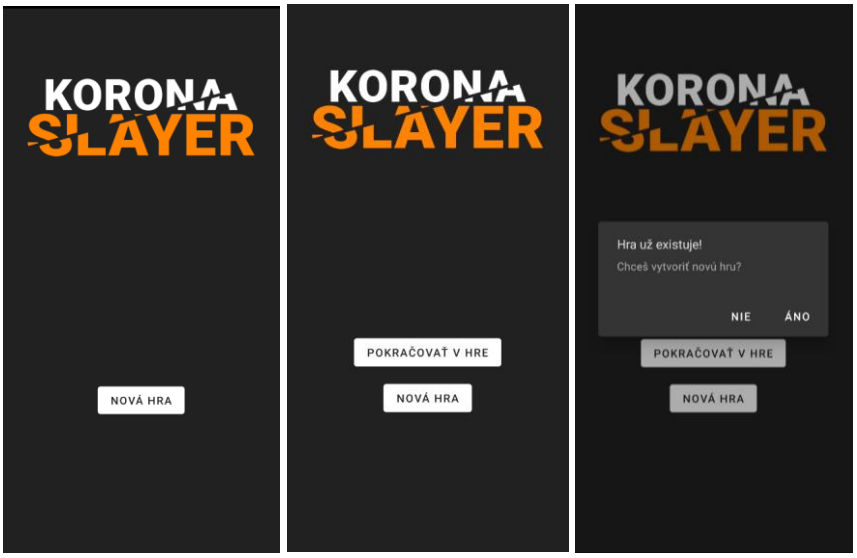
5.7 Aktivity a fragmenty

V aplikácií mám vytvorených celkovo 8 fragmentov, z ktorých sú niektoré iba informatívne v iných je naopak implementovaná hlavná logika celej aplikácie. Pre väčšinu fragmentov som taktiež vytváral landscape XML verzie pre krajšie zobrazenie.

5.7.1 UvodnaStranaFragment

Fragment, ktorý je hráčovi zobrazený ako prvý. Pri spustení aplikácie fragment obsahuje logo hry a 1 tlačidlo umožňujúce vytvorenie novej hry. Po vytvorení hry a vrátení sa opäť na tento fragment je tu zobrazené ja tlačidlo umožňujúce pokračovať v už vytvorenej hre. V prípade, že je hra vytvorené a hráč sa rozhodne vytvoriť novú je mu zobrazené dialógové okno, kde sa môže hráč rozhodnúť či chce vyresetovať už vytvorenú hru alebo chce svoje rozhodnutie zobrať späť.

Ukážka obrazoviek:



5.7.2 NovaHraFragment

Fragment slúžiaci pre zvolenie postavy. Na fragmente sú zobrazené 2 klikacie ConstrainLayouty zobrazujúce základné informácie o každej postave. Horný AppBar taktiež obsahuje tlačidlo umožňujúce návrat na predošlú obrazovku.

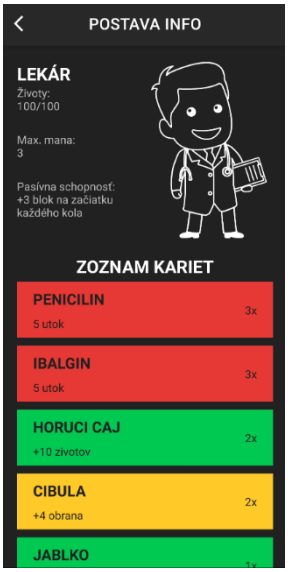
Ukážka obrazovky:

5.7.3 MapaHryFragment

Fragment slúžiaci hráčovi pre výber levelu. Na fragmente je zobrazených 10 tlačidiel, pre každý level jedno. V hornom AppBare je umiestnené tlačidlo umožňujúce návrat na predošlú obrazovku a tlačidlo umožňujúce presmerovanie na PostavaInfoFragment. V prípade, že hráč klikne na červená tlačidlo je mu zobrazený „Toast“, ktorý ho informuje o už prejdení daného levelu. Po kliknutí na biele tlačidlo je hráč informovaný, že ešte musí prejsť určité levely aby sa k nemu dostal a po kliknutí na zelené tlačidlo je hráč presmerovaný na BojFragment.

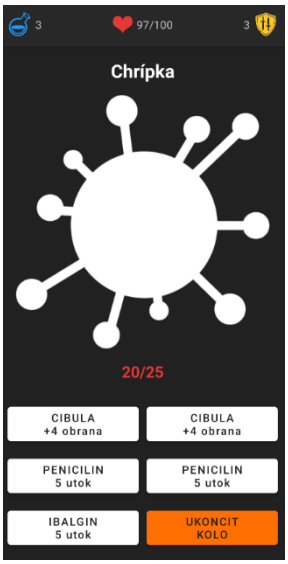
5.7.4 PostavaInfoFragment

Fragment slúžiaci na zobrazenie základných informácií o postave a taktiež o zozname kariet v hráčovom balíčku. V hornom AppBare je okrem iného umiestnené tlačidlo umožňujúce presmerovanie na predošlú obrazovku. Zoznam kariet umiestnený v spodnej časti je realizovaný pomocou RecycleView.



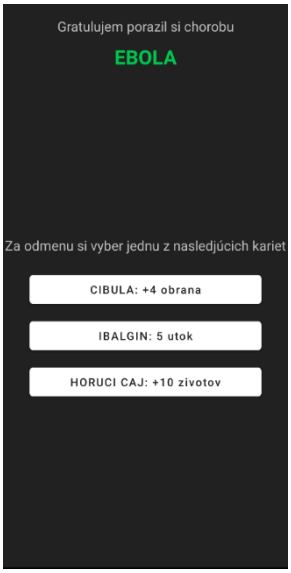
5.7.5 BojFragment

Fragment slúži pre implementáciu základnej logiky hry. V hornom AppBare sú zobrazené základné štatistiky o postave hráča, koľko má many, životov a bloku. V nižšej časti je zobrazené meno choroby, ilustračný obrázok reprezentujúci typ choroby a aktuálne/maximálne životy choroby. V samotnej spodnej časti je zobrazených 5 tlačidiel kariet a 1 tlačidlo určené na predčasné ukončenie kola. V prípade smrti hráča je hráč presmerovaný na PrehraFragment naopak pri výhre hráča v prípade, že sa nachádza v poslednom levely je hráč presmerovaný na VyhraFragment. Pokiaľ hráč vyhrá a nenachádza sa v poslednom levely je presmerovaný na VyberKartyFragment. V prípade stlačenia spätného tlačidla je hráč upozornený formou dialógového okna na skutočnosť že bude vrátený na mapu a životy nepriateľa budú obnovené na jeho maximálnu hodnotu.



5.7.6 VyberKartyFragment

Fragment slúžiaci na výber novej karty. Na fragmente je zobrazený informatívny text o porazenej chorobe a o možnosti výberu novej karty. V strednej časti sú zobrazené 3 tlačidlá reprezentujúce nové karty z ktorých si hráč môže vybrať. Po výbere karty je hráč presmerovaný opäť na MapaHryFragment. Pokiaľ hráč stlačí spätné tlačidlo je upozornený formou dialógového okna na skutočnosť že žiadna karta nebude pridaná do jeho balíčka kariet a je na ňom či sa rozhodne fragment naozaj opustiť alebo zostať a vybrať si jednu z kariet.



5.7.7 PrehraFragment

Jednoduchý fragment zobrazujúcu informáciu o prehre a o chorobe ktorá hráča premohla. Okrem iného je vyobrazený aj obrázok mŕtvej postavy hráča a 2 tlačidlá umožňujúce návrat hráča na úvodnú obrazovku alebo presmerovanie na výber postavy.

5.7.8 VyhraFragment

Fragment podobný fragmentu zobrazujúcemu prehru. Taktiež zobrazuje informatívny text, ilustračný obrázok a 2 tlačidlá s rovnakou funkcionalitou.