Opis rada aplikacije sa slikama:

Aplikacija ima custom ikonicu koja je u nju dodata kao Image Asset, a prilikom samog dodavanja override-ala je default-nu ikonicu koju ima svaka Android aplikacija koja se kreira koristeći Android Studio.

Klikom na ikonicu aplikacije na mobilnom uređaju gdje je instalirana, aplikacija se pokreće i otvara se početna aktivnost pod nazivom „UvodnaActivity“ koja podešava contentView na fragment koji odgovara ovoj aktivnosti tj. na fragment „activity\_uvodna“.

Kako je postignuto da se otvori prvo aktivnost „UvodnaActivity“, a ne „MainActivity“ koja bude kreirana prilikom kreiranja samog Empty project-a? Tako što smo u manifest fajl „AndroidManifest“ zamijenili aktivnosti „MainActivity“ i „UvodnaActivity“, a samom tom zamjenom je na uvodnoj aktivnosti atribut „android:exported“ dobio vrijednost „true“, a isti atribut main aktivnosti je uzeo vrijednost „false“.

Fragment „activity\_uvodna“ biva prikazan na ekranu 4000 milisekundi tj. 4 sekunde, nakon čega se iz uvodne aktivnosti prelazi na main aktivnost tj. „activity\_main“ preko Intent-a i Handler-a. Također se unutar handler-a pozove funkcija finish() koja uništava uvodnu aktivnost nakon prelaska na novu aktivnost, tj. neće biti moguće klikom na dugme nazad iz main aktivnosti doći u uvodnu aktivnost tj. njen odgovarajući fragment.

Prilikom pokretanja main aktivnosti contentView se setuje na „activity\_main“ fragment, te potom se između ostaloga uštimava side menu tj. njegovo ponašanje, preciznije šta se dešava klikom na koji menu\_item. Također, imamo funkciju onSupportNavigateUp() koja klikom na neki od item-a menija skače sa trenutno otvorenog fragmenta na kliknuti fragment.

Po default-u se prilikom kreiranja main aktivnosti prikazuje „fragment\_lista\_gradova“ gdje se u traci na vrhu tj. app bar-u, sa lijeve strane nalazi tzv. „Hamburger menu“, klikom na te tri crtice sa lijeve ivice ekrana se proširuje ka desno side menu koji ima tri opcije, odlazak na fragment liste gradova tj. „fragment\_lista\_gradova“, koji je po defaultu otvoren, zatim na fragment dodaj grad tj. „fragment\_dodaj\_grad“ i fragment o aplikaciji tj. „fragment\_o\_aplikaciji“ . Pomenuti meni se mogao prikazati i swipe-om tj. povlačenjem sa lijeve ivice ekrana prema desno. Meni se može uvući nazad lijevo, klikom desno od menija na „prazan“ prostor, ili swipe-om tj. povlačenjem, ali u ovom slučaju prema lijevoj ivici ekrana, ili klikom na ikonicu strijelice lijevo koja stoji sada umjesto hambruger ikonice u traci na vrhu aplikacije. Fragment liste gradova, po default-u prikaže listu od 6 gradova, preciznije svaka kartica prikazuje naziv, latitudu i longitudu grada.

Klikom na neku od kartica otvara se fragment „fragment\_detalji\_grada“ koji prikazuje detalje kliknutog grada, ovom fragmentu proslijeđen je preko safeargs-a grad na čiju karticu smo kliknuli, te niz svih preostalih gradova. Ovo proslijeđivanje custom klase, u našem slučaju klase „Grad“, omogućeno je koristeći ključne riječi @Parcelize iznad same deklaracije klase. Fragment detalji grada prikazuje neke dodatne informacija o gradu na čiji okvir se pritisnulo u fragmentu liste gradova, sve informacije koje se prikazuju su: naziv, država, latituda, longituda i opis grada. Ispod navedenih atributa grada se prikazuju i dva dugmeta, prvo „PRIKAŽI NA MAPI“ i drugo „IDI NA ALGORITAM“. Također, u desnom dijelu gornje trake se prikazuje ikonica za dijeljenje, klikom na nju se može bilo kojom aplikaciju koja podržava „share“ opciju podijeliti predefinisana poruka, a njen sadržaj je naziv grada sa latitudom i longitudom.

Klikom na prvo dugme otvara se novi fragment i to „fragment\_grad\_na\_mapi“, koji prikazuje unutar aplikacije Google Map-u i to fokusira je tako da je u samoj njenoj sredini stoji crveni marker, a koordinate markera su latituda i longituda grada iz kojeg smo stisnuli dugme. Nazad na fragment o detaljima grada možemo doći stiskanjem dugmeta „nazad“ na dnu mobitela ili stiskanjem ikonice nazad lijevo u appbar-u na vrhu aplikacije. Po mapi se može zumirati kao i na svakoj Google Mapi, također moguće ju je i rotirati.

Klikom na dugme „IDI NA ALGORITAM“ otvara se algoritam fragment tj. „fragment\_algoritam“ kojem se također proslijeđuju grad i gradovi koje smo prihvatili u fragmentu o detaljima grada. Algoritam Fragment prikazuje polazni grad, a to je grad iz čijih smo detalja došli, te ispod toga imamo dugme „POKRENI ALGORITAM“. Klikom na navedeno dugme, pokreće se brute-force algoritam koji računa najkraću rutu za obliazak svih preostalih gradova iz grada iz čijih detalja smo došli. Nakon što se algoritam izvrši, tj. pronađe najbolju rutu, pojavi se TextView koji prikazuje tu rutu, a dugme „POKRENI ALGORITAM“ nestaje, a pojavljuje se novo dugme „PRIKAŽI RUTU“.

Klikom na dugme „PRIKAŽI RUTU“ otvara se novi fragment „fragment\_najkraca\_ruta“, koji otvara mapu u aplikaciji i iscrtava pronađenu najkraću rutu, pri čemu je polazni grad označen crvenim markerom.

Fragment „fragment\_dodaj\_grad“ sadrži 5 TextView-ova koji predstavljaju atribute instance klase Grad, te ispod njih dugme „DODAJ GRAD“. Nakon što se popuni svih 5 input polja, tapkanjem dugmeta „DODAJ GRAD“ u globalnu varijablu „gradovi“ koja se nalazi u klasi „GlobalnaKlasa“, doda se novi grad. Ukoliko neko polje nije popunjeno, prikazat će se adekvatna poruka na ekranu, tek kada sva polja budu popunjena, stiskanjem dugmeta omogućeno je dodavanje grada u globalnu varijablu „gradovi“ klase „GlobalnaKlasa“. Također, stiskanjem dugmeta se sakriva tastatura koja je ostala nakon unošenja teksta u navedena input polja.

Fragment „fragment\_o\_aplikaciji“ sadrži osnovne informacije o aplikaciji, može mu se pristupiti iz side menu-a o kojem smo govorili na početku.

Opis arhitekture aplikacije:

Aplikacija ima jedan manifest fajl pod nazivom “AndroidManifest”, dvije aktivnosti i 7 fragmenata koji se nazivaju “UvodnaActivity” i “MainActivity”, a fragmenti: “AlgoritamFragment”, “DetaljiGradaFragment”, “DodajGradFragment”, “GradNaMapiFragment”, ”ListaGradovaFragment”, ”NajkracaRutaFragment”, ”OAplikacijiFragment”. Svaki od navedenih aktivnosti i fragmenata imaju odgovarajuće layout .xml fajlove. Klase: “Grad” i “GlobalnaKlasa”, te jedan adapter “GradAdapter” koji se koristi za prikazivanje liste gradova u RecyclerView-u za “ListaGradovaFragment”. Dodate su neke ikonice i slike za ljepši izgled aplikacije. Čitava aplikacija je prevedena i na engleski jezik pomoću string resource fajla, također bitno je napomenuti da čitava aplikacija radi i u portrait i landscape mode-u. Svi text-ovi, fontovi i dimenzije su smješteni u res folderu, ništa nije hard-codirano.

Jedna je navigacija pod nazivom “navigation” i koristi se za navigaciju između fragmenata kroz čitavu aplikaciju, zajedno sa side menu-om.

Opis funkcionalnosti pojedinačnih klasa

Klasa Grad:

class Grad (var naziv: String, var longituda: String, var latituda: String, var drzava: String, var opis: String)

Je implementirana na način da se sastoji od pet atributa: ‘naziv’ – naziv grada, ‘longituda’ – longitude grada, ‘latituda’ – latitude grada, ‘drzava’ – označava državu grada i ‘opis’ – označava opis grada, odnosno neke informacije o samom objektu tipa ‘Grad’. Klasa sadrži samo konstruktor koji prima svih pet ovih parametara. Ne postoji konstruktor bez parametara. Ova klasa je implementirana iz razloga što je potrebno da se napravi lista objekata tipa ‘Grad’ i ta lista se koristi u fragmentu ‘ListaGradovaFragment.kt’ gdje se prikazuju svi gradovi iz liste.

GradAdapter.kt (kotlin fajl)

Pošto se prikazuje lista gradova, iz tog razloga potreban je Adapter jer bez njega ne možemo raditi sa listama koje imamo u Recycler View-u. Adapter upravlja elementima liste, pa tako u Adapteru postavljamo Koliko elemenata želimo predstaviti u listi, način na koji će se elementi prikazivati, postavljamo layout koji je jedan element liste. Ovdje je također implementirana i klasa “CityViewHolder” koja nasljeđuje RecyclerView.viewHolder. Ova klasa “CityViewHolder” nam čuva podatke koji se nalaze u tom jednom elementu liste u Recyclew View-u (podatke koje sadrži jedan list item). Klasa “GradAdapter” nasljeđuje RecyclerView.Adapter<GradAdapter.CityViewHolder> iz razloga jer je to Adapter klasa. Metoda “onCreateViewHolder” vraća CityViewHolder koji prima view a to je zapravo taj jedan list item.

Metoda “onBindViewHolder” – tu postavljamo vrijednosti na elemente koje smo definirali u klasi “CityViewHolder”. Ova metoda prima holder I position.

Metoda “getItemCount()” – ova metoda vraća koliko će biti prikazano list item-a na Recycler View-u, a to je u ovom slučaju dužina liste “listaGradova.size”.

Klasa GlobalnaKlasa: (Proširuje tj. nasljeđuje klasu Application)

Kao što samo ime klase govori ova klasa je globalna, tj. čuva određene podatke kojim se pristupa kroz čitavu aplikaciju. Preciznije sadrži companion object unutar kojeg se nalazi varijabla gradovi koja je tipa MutableList<Grad>, ova varijabla služi da se čuva niz gradova koje trenutno imamo u aplikaciji. Prilikom svakog pokretanja aplikacije popuni se sa 6 gradova, tj. inicijalizira se sa 6 gradova, to se dešava u ovoj klasi, no kroz aplikaciju moguće je ovaj niz gradova proširiti sa novim gradovima, to se radi kroz fragment za dodavanje grada pod nazivmo “DodajGradFragment”. Mana ovog pristupa čuvanja gradova u aplikaciji je da prilikom izlaska iz aplikacije svi gradovi koji su dodati kroz pomenuti fragment za dodavanje, bivaju izbrisani, tj. ne budu sačuvani. To znači da opet kad se uđe u aplikaciju niz gradovi se inicijalizira sa standardnih 6 gradova o kojima je već bilo priče i onda se na taj niz može opet kroz fragment za dodavanje dodati novi grad tj. novi gradovi, ako se želi dodati više gradova.

Opis opštih koncepata Android frameworka

Activity

Activity klasa služi za iscrtavanje korisničkog interfejsa i prima događaje od korisnika (event listenere). U principu aktivnost nam predstavlja jedan ekran koji vidimo.

Fragment

Fragment nastaje sa razvojem tableta. Dio interfejsa koji se može ponovo koristiti i svaki fragment ima svoj layout, lifecycle i sam se bavi svojim elementima. Svaki fragment mora biti hostiran na jednoj aktivnosti. U principu fragmenti i aktivnosti su dosta slični, samo što fragmente mi više kontrolišemo.

Data Binding

Korisitmo ga da bi povezali komponente iz layouta. da bi izbjegli stalno korištenje findViewById, imamo opciju da uključimo “binding” u gradle fajlu – buildFeatures { databinding true }. Ono što još moramo uraditi da bi mogli koristiti “binding” jeste da cijeli layout ubacimo u tag “<layout>”. Nakon toga u “Mainu” napravimo objekat koji će nam predstavljati binding. Npr. “private lateinit var binding: ActivityMainBinding” – korisitmo camel case notaciju i komponentama iz layouta pristupamo na način “binding.idKomponente”.

SafeArgs

Prosljeđivanje podataka između fragmenata. Navigacija omogućava da priložimo podatke operaciji navigacije definiranjem argumenata za odredište. Na primjer, odredište korisničkog profila može uzeti argument korisničkog ID-a da odredi kojeg će korisnika prikazati. Općenito, treba se snažno preferirati prosljeđivanje samo minimalne količine podataka između fragmenata. Na primjer, trebali biste proslijediti ključ za dohvaćanje objekta umjesto samog objekta, jer je ukupan prostor za sva sačuvana stanja ograničen na Androidu. Ako trebate proslijediti velike količine podataka, razmislite o korištenju ViewModel-a kao što je opisano u Dijeli podatke između fragmenata.

ViewBinding

ViewBinding (Povezivanje View-ova) je feature koja omogućuje lakše pisanje koda koji je u interakciji s pogledima (view-ovima). Jednom kada je povezivanje pogleda omogućeno u modulu, on generira klasu vezanja za svaku XML datoteku izgleda koja je prisutna u tom modulu. Instanca klase povezivanja sadrži direktne reference na sve poglede koji imaju ID u odgovarajućem izgledu. U većini slučajeva, vezanje pogleda zamjenjuje findViewById. findViewById izvor je mnogih bugova s kojima se suočavaju korisnici u Androidu. Lako je proslijediti id koji nije u trenutnom izgledu - stvarajući null i rušenje. A budući da nema ugrađenu sigurnost tipa, lako je poslati kod koji poziva findViewById<TextView>(R.id.image). View binding zamjenjuje findViewById sažetom, sigurnom alternativom.

Prednosti View Bindinga nad findViewById izvorom su:

Sigurno za tip jer su svojstva uvijek ispravno upisana na temelju pogleda u izgledu. Dakle, ako stavite TextView u izgled, vezanje pogleda će izložiti svojstvo TextView.

Null-safe za izglede definirane u više konfiguracija. Povezivanje pogleda će otkriti je li pogled prisutan samo u nekim konfiguracijama i stvoriti svojstvo @Nullable.

A budući da su generirane klase vezanja redovite Java klase s Kotlin-prijateljskim bilješkama, možemo koristiti povezivanje pogleda iz programskog jezika Java i Kotlina.

Recycler View

Je view u kojem se prikazuju elementi određene liste. Da bi koristili Recycler View potreban je i LayoutManager koji upravlja tim elementima. Recycler View prikazuje samo onoliko elemenata liste koliko se može prikazati u jednom momentu na ekranu, a mi kada skrolamo po ekranu, onaj view koji izlazi iz vidnog polja se uništi a kreira se novi view (list item). Layout Manager definira kako ćemo predstaviti listu, hoće li to biti horizontalno, vertikalno, ili kao grid. Da bi ovo sve radilo potreban nam je Adapter koji služi da bi radili sa listama. Adapter je klasa koja upravlja sa elementima liste.

ConstraintLayout

Omogućava nam stvaranje velikih i složenih izgleda s ravnom hijerarhijom pogleda (bez ugniježđenih grupa pogleda). Sličan je RelativeLayoutu po tome što su svi pogledi raspoređeni prema odnosima između srodnih pogleda i roditeljskog izgleda, ali je fleksibilniji od RelativeLayouta i lakši za korištenje s Android Studioovim uređivačom izgleda.

Lifecycle

Sastoji se od 7 metoda. Prva metoda je “onCreate”, ostale metode su: “onResume”, “onPause”, “onStart”, “onRestart”, “onStop”, “onDestroy”.