ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Chef ’s recipies

NÁVRH MOBILNEJ APLIKÁCIE NA UCHOVÁVANIE RECEPOTOV

vypracovali:

fakulta: Riadenia a Informatiky

študijná skupina: 5ZYI21

cvičiaci: Ing. Michal Ďuračík PhD.

termín cvičenia: pondelok 8:00

V Žiline dňa 11.6.2024

Obsah

1. Aplikácie podobného zamerania

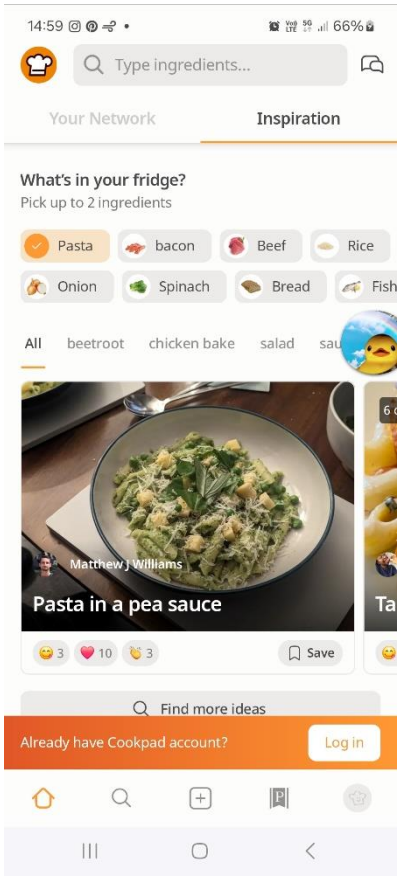
1.1. Cookpad

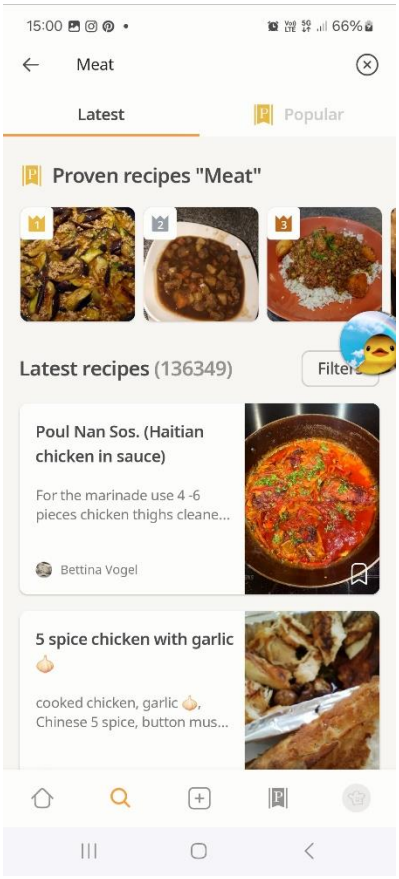
Výhody aplikácie Cookpad:

- Systém kategórií je veľmi dobre spravený kde je možne vyhľadávať podľa napr. ingendiencií
- Aplikácia je viac tvorená ako sociálna sieť kde recept zdieľate s ostatnými ľuďmi pomocou už vstavaných funkcií
- Samotné recepty sú veľmi pekne graficky spracované s väčšinou podstatných informácií

Nevýhody aplikácie Cookpad:

- Nie je možné upravovať recepty kvôli úpravám pre užívateľove podmienky (Iná rúra, špeciálna miska, iné množstvo cukru atď.)
- Veľa funkcionalít je otvorených iba pre prémiových užívateľov
- Aplikácia nemá slovenskú lokalizáciu





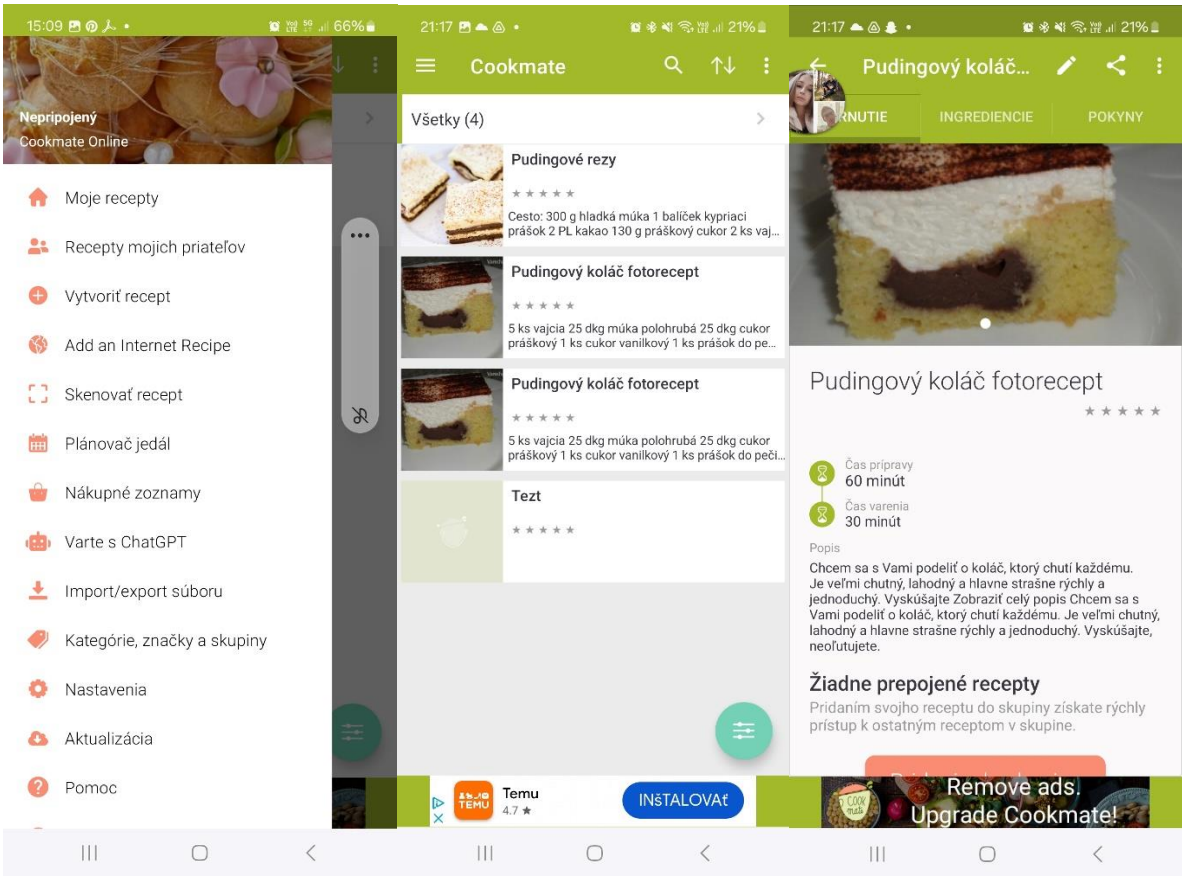
1.2. Cookmate

Výhody aplikácie Cookmate:

- Aplikácia Cookmate Excelu hlavne v rozmanitosti funkcií ktoré ponúka. Od vytvorenia receptov ručne alebo importovaním z podporovaných webstránok cez kategorizáciu jedál podľa upravitelných tagov až po vytváranie nákupného zoznamu z zoznamu ingrediencií ba dokonca sa zviezla s relatívne novým trendom ChatGPT a existuje možnosť variť teda vytvárať recept s jeho pomocou
- Aplikácia síce má Premium ale zamknuté funkcionality výrazne neobmedzujú užívateľa teda aplikácia sa dá používať aj bez prémiového účtu
- Existuje možnosť zdieľať recepty s priateľmi/blízkymi priamo skrz aplikáciu

Nevýhody aplikácie Cookmate:

- Vzhľad aplikácie nie je až tak lákavý hlavne najdôležitejšia časť recepty. Taktiež je cítiť akúsi nekonzistentnosť medzi jednotlivými obrazovkami.
- V postupoch pri receptoch chýbajú rozšírené možnosti ich tvorenia. V aplikácií je možne zadávať iba čistý text + odkazy. Veľmi chýbajú minimálne obrázky pre každý krok postupu
- Človek sa stráca dakedy čo sa v aplikácií dá robiť a kde danú funkcionalitu nájsť



2. Analýza navrhovanej aplikácií

2.1. Funkcie

Aplikácia musí mať niekoľko funkcií. Hlavnou funkciu je zobrazenie receptov. Aplikácie ho musí byť schopná správne a jednoducho zobraziť aby nemohli vzniknúť nejasnosti. Následne aplikácia musí byť schopná uložiť recepty do nejakej formy trvalého úložiska ako taktiež načítavať dáta z tohto úložiska. Aplikácia ďalej na základe analýzy iných aplikácií by mohla mať systém tagov kde tagy bude môcť vytvárať používateľ. Pomocou nich bude môcť následne filtrovať recepty.

Aplikácia bude mať aj akýsi užívateľský systém aby bolo možne identifikovať autora receptu.

2.2. Výkonnosť

Výkonnosť v takomto type aplikácie nehrá veľkú úlohu. Jediná požiadavka je aby grafické rozhranie aplikácie prebiehalo plynule. To znamená aby aplikácia bola neustále responzívna aj počas dlhodobých operáciách ako napríklad načítaní dát z dlhodobého úložiska alebo sťahovania nejakého iného obsahu z internetu.

2.3. Vzhľad

Aplikácia by mala byť veľmi príjemná na vzhľad. Technický vzhľad v takomto type aplikácií užívateľov odrádza od používania. Taktiež celý design by mal byť responzívny teda mal by prispôsobiť svoj vzhľad na základe orientácie displeja.

3. Návrh architektúry aplikácie

3.1. Popis

Moja aplikácia bude zodpovedná za vytváranie ukladanie a ukazovanie receptov. Recepty bude možne zdieľať pomocou správy ktorá bude môcť byť následne importovaná do aplikácie. Recepty bude možne filtrovať na základe užívateľom vytvorených tagov ako aj preddefinovaných hodnôt (napr. Čas varenia, počet hviezdičiek, ingrediencií). Aplikácia bude fungovať iba lokálne a k zdieľaniu bude potrebné exportované dáta poslať cez inú aplikáciu (ako napr. Mail alebo Messenger).

3.2. Dátová časť

Dáta sa budú ukladať v lokálnej databáze pomocou knižnice Room. V databáze bude uložený obsah receptov ako aj samotné recepty. Používateľove nastavenia ako aj jeho meno s cestou k jeho profilovej fotografií bude uložený v kľúčom definovanej databáze s pomocou knižnice DataStore.

3.3. BackEnd

Backend bude riešený cez architektúru MVVM ako viewmodel triedy pričom využijeme vstavanú Android implementáciu. Vo viewmodeloch budeme ukladať stav aplikácie a pomocou neho komunikovať s databázou. Budeme preferovať politiku 1 viewmodel ku 1 view(teda k jednej obrazovke).

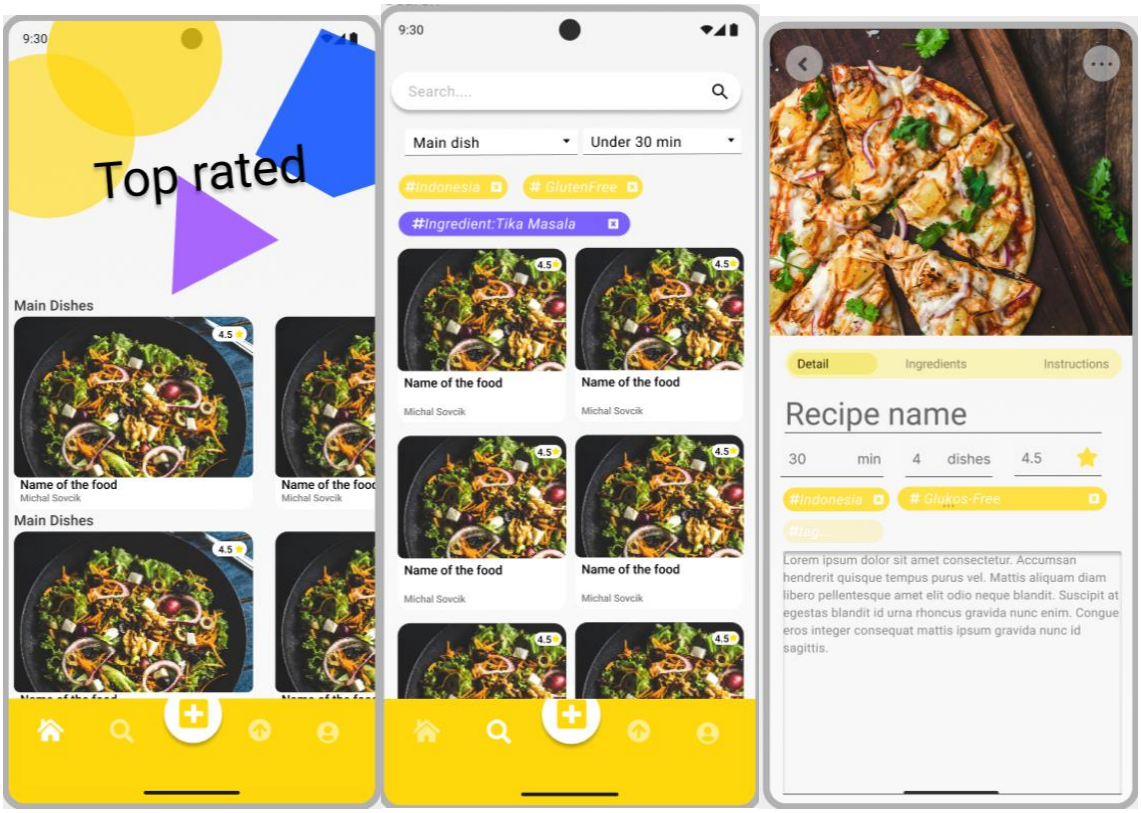
V jednom viewmodely budeme taktiež využívať internetovú komunikáciu pomocou courotines aby sme zozbierali dáta z stránky.

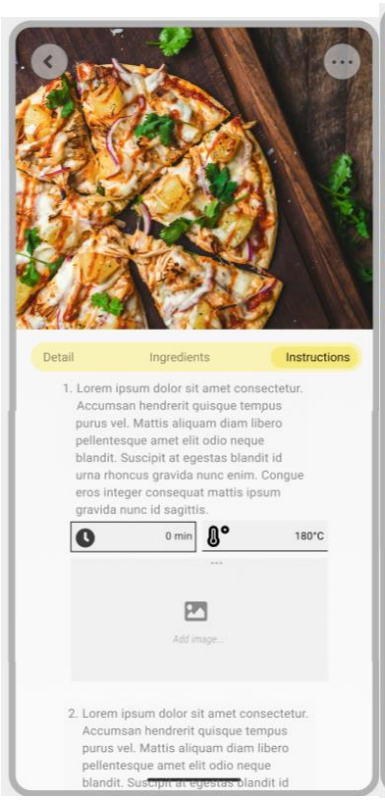
3.4. Front End (Obrazovky)

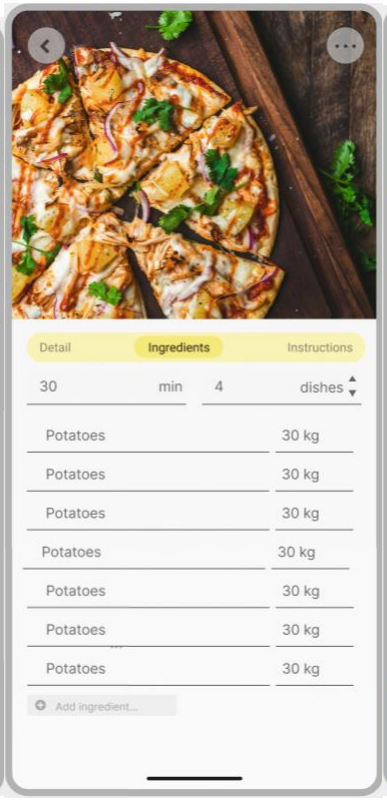
V aplikácií budeme mať dokopy 5 obrazoviek pričom 4 z nich budú navigovateľné cez spodnú lištu s navigáciou a 5 bude slúžiť na zobrazenie detailu receptu. Obrazovky budú nasledovné:

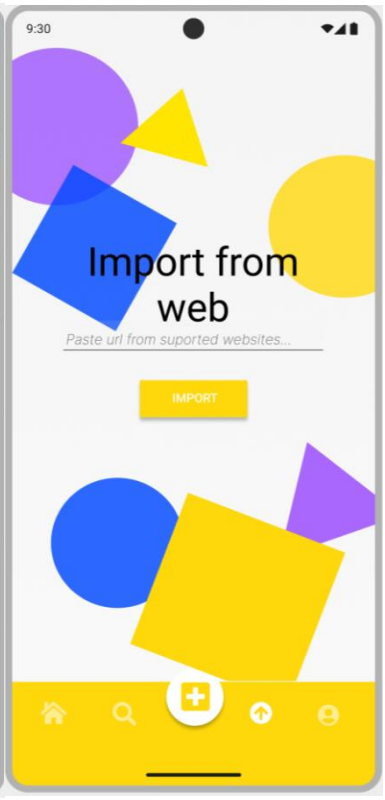
- Domov – Štartovná obrazovka na ktorej bude aplikácia nastavená keď sa prvý krát spustí. Budú na nej naposledy zapísané recepty v jednotlivých kategóriách.
- Hľadaj – obrazovka na ktorej bude možné vyhľadávať recepty na základe ich tagu, meno, typu jedla, ingrediencií a času vyhotovenia
- Import – bude obrazovka na ktorej bude možné importovať recept z iných podporovaných stránkach. (Táto funkcionalita možno nebude v výslednej aplikácií implementovaná)
- Profil – v tejto obrazovke používateľ nastavuje svoje meno, profilovú fotku a iné nastavenia ktoré modifikujú chod aplikácie. Taktiež môžeme na tejto obrazovke vidieť koľko receptov sme už celkovo vytvorili/naimportovali.
- Recept – Obrazovka ktorá ukazuje vybraný recept so všetkými jeho časťami akými sú detail receptu, ingrediencie receptu a postup pri recepte

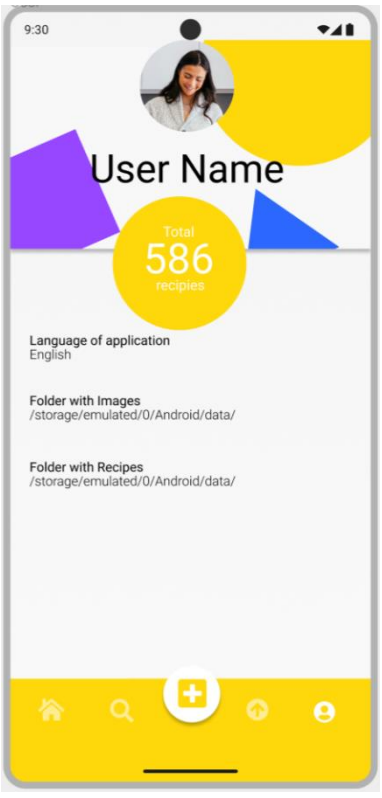
4. Ukážka návrhu obrazoviek aplikácie




Detail
Ingredients
Instructions
1. Lorem ipsum dolor sit amet consectetur. Accumsan hendrerit quisque tempus purus vel. Mattis aliquam diam libero pellentesque amet elit odio neque blandit. Suscipit at egestas blandit id urna rhoncus gravida nunc enim. Congue eros integer consequat mattis ipsum gravida nunc id sagittis.
0 min
180°C
Add image...
2. Lorem ipsum dolor sit amet consectetur. Accumsan hendrerit quisque tempus purus vel. Mattis aliquam diam libero pellentesque amet elit odio neque blandit. Suscipit at egestas blandit id


Detail
Ingredients
Instructions
30
min
4
dishes
Potatoes
30 kg
Potatoes
30 kg
Potatoes
30 kg
Potatoes
30 kg
Potatoes
30 kg
Potatoes
30 kg
Potatoes
30 kg
Add ingredient...

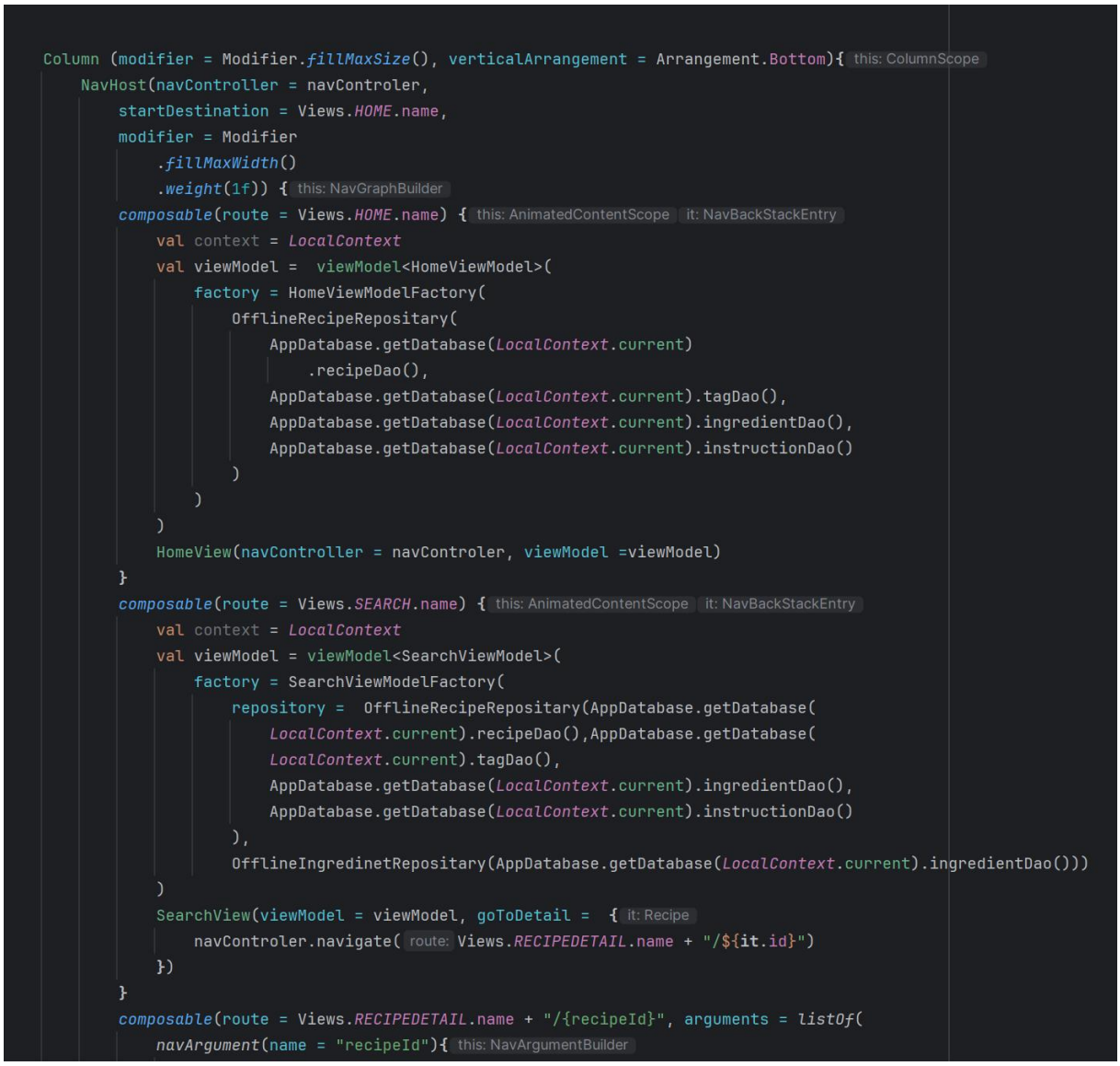

9:30
Import from web
Paste url from suported websites...
IMPORT


9:30
User Name
Total
586
recipies
Language of application
English
Folder with Images
/storage/emulated/0/Android/data/
Folder with Recipes
/storage/emulated/0/Android/data/

5. Popis implementácie

5.1. Navigation

Komponent Navigation používam na navigáciu medzi jednotlivými obrazovkami. Taktiež ho využívam pri zobrazovaní detailu receptu kde pomocou neho sa prepínam medzi jednotlivým pohľadmi na recept (Opis, Ingrendiencie, Postup)



```
Column (modifier = Modifier.fillMaxSize(), verticalArrangement = Arrangement.Bottom){ this: ColumnScope
    NavHost(navController = navControler,
        startDestination = Views.HOME.name,
        modifier = Modifier
            .fillMaxWidth()
            .weight(1f)) { this: NavGraphBuilder
        composable(route = Views.HOME.name) { this: AnimatedContentScope  it: NavBackStackEntry
            val context = LocalContext
            val viewModel =  viewModel<HomeViewModel>(
                factory = HomeViewModelFactory(
                    OfflineRecipeRepositary(
                        AppDatabase.getDatabase(LocalContext.current)
                            .recipeDao(),
                        AppDatabase.getDatabase(LocalContext.current).tagDao(),
                        AppDatabase.getDatabase(LocalContext.current).ingredientDao(),
                        AppDatabase.getDatabase(LocalContext.current).instructionDao()
                    )
                )
            )
            HomeView(navController = navControler, viewModel =viewModel)
        }
        composable(route = Views.SEARCH.name) { this: AnimatedContentScope  it: NavBackStackEntry
            val context = LocalContext
            val viewModel = viewModel<SearchViewModel>(
                factory = SearchViewModelFactory(
                    repository =  OfflineRecipeRepositary(AppDatabase.getDatabase(
                        LocalContext.current).recipeDao(),AppDatabase.getDatabase(
                        LocalContext.current).tagDao(),
                        AppDatabase.getDatabase(LocalContext.current).ingredientDao(),
                        AppDatabase.getDatabase(LocalContext.current).instructionDao()
                    ),
                    OfflineIngredinetRepositary(AppDatabase.getDatabase(LocalContext.current).ingredientDao()))
            )
            SearchView(viewModel = viewModel, goToDetail =  { it: Recipe
                navControler.navigate( route: Views.RECIPEDETAIL.name + "/${it.id}")
            })
        }
        composable(route = Views.RECIPEDETAIL.name + "/{recipeId}", arguments = listOf(
            navArgument(name = "recipeId"){ this: NavArgumentBuilder
```

5.2. Room

Databázu Room využívam na ukladanie všetky informácií ohľadom receptov (Okrem obrázkov ktoré su moc veľké na uloženie do databázy). Návrh databázy zahrnuje aj vytvorenie interfacov repozitárov ktoré je možné implementovať podľa potreby takže lokálnu offline databázu je možné veľmi jednoducho nahradiť iným typom lokálnej databázy alebo online databázov.

```
@Database(entities = [Recipe::class, Ingredient::class, Instruction::class, Tag::class],
    version = 1,
    exportSchema = false
)
@TypeConverters(Converters::class)
abstract class AppDatabase : RoomDatabase(){
    GoodBoySK
    abstract fun recipeDao(): RecipeDao
    GoodBoySK
    abstract fun ingredientDao(): IngredientDao
    GoodBoySK
    abstract fun instructionDao(): InstructionDao
    GoodBoySK
    abstract fun tagDao(): TagDao
    GoodBoySK
    companion object {
        @Volatile
        private var Instance: AppDatabase? = null

        GoodBoySK
        fun getDatabase(context: Context): AppDatabase {
            // if the Instance is not null, return it, otherwise create a new database instance.
            //synchroniczed = can be run only by 1 thread at a time
            return Instance ?: synchronized( lock: this) {
                Room.databaseBuilder(context, AppDatabase::class.java,  name: "ChefsRecipeDatabase")
                    .build() AppDatabase
                    .also { Instance = it }
            }
        }
    }


}
```

5.3. ViewModel

V celom projekte je dodržaná koncepcia viewmodel kde v týchto triedach je uchovávaný stav a taktiež zaobalená funkčnosť. Každá obrazovka má svoj vlastný viewmodel v ktorom uchováva stav na obrazovke. Kvôli tomu otočenie displeja sa bude chovať korektne (Nezmiznú žiadne informácie po rekompozícií)

```
class HomeViewModel(val recipeRepositary: RecipeRepositary) : ViewModel() {
    private val _recipesFilter:StateFlow<FilteredRecipes> = MutableStateFlow(FilteredRecipes())
    private val _maindishFlowRecipes: StateFlow<List<Recipe>>
    private val _dezerFlowRecipes: StateFlow<List<Recipe>>
    private val _soupFlowRecipes: StateFlow<List<Recipe>>
    private val _breakfastFlowRecipes: StateFlow<List<Recipe>>

    GoodBoySK
    init {
        _maindishFlowRecipes = recipeRepositary.getRecipeOfType(DishType.MainDish).stateIn(viewModelScope,
            SharingStarted.WhileSubscribed( stopTimeoutMillis: 100), emptyList())
        _dezerFlowRecipes = recipeRepositary.getRecipeOfType(DishType.Dezert).stateIn(viewModelScope,
            SharingStarted.WhileSubscribed( stopTimeoutMillis: 100), emptyList())
        _soupFlowRecipes = recipeRepositary.getRecipeOfType(DishType.Soup).stateIn(viewModelScope,
            SharingStarted.WhileSubscribed( stopTimeoutMillis: 100), emptyList())
        _breakfastFlowRecipes = recipeRepositary.getRecipeOfType(DishType.Breakfast).stateIn(viewModelScope,
            SharingStarted.WhileSubscribed( stopTimeoutMillis: 100), emptyList())
```

5.4. Notifikácie

Notifikácia je posielaná ako súčasť služby časovača kde v notifikácií je možné vidieť zostávajúci čas časovača. Pre notifikáciu je na začiatku aplikácie vytvorený notifikačný kanál s názvom „stops_chanel“

```
    private fun start(text: String, last:Boolean) {

        var notification = NotificationCompat.Builder( context: this, channelId: "stops_chanel").
        setContentTitle("Timer")
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentText(text)
            .setSilent(!last)

        startForeground( id: 1 ,notification.build())

    }
```

5.5. Service (Binding service)

Táto služba simuluje časovač kde si automaticky sleduje zostávajúci čas a pomaly ho znižuje. Táto služba je typu Bound teda je možné sa na ňu napojiť a získavať z nej informácie

```
class StopwatchService : Service() {

    private val binder = LocalBinder()
    private var job: Job? = null
    private var _seconds:MutableStateFlow<Int> = MutableStateFlow<Int>( value: 0)
    val seconds = _seconds.asStateFlow()
    GoodBoySK
    companion object{
        val TIMEKEY = "time"
    }
    GoodBoySK
    override fun onBind(intent: Intent?): IBinder {
        return binder
    }
    GoodBoySK
    inner class LocalBinder : Binder()
    {
        GoodBoySK
        fun getService():StopwatchService = this@StopwatchService;
    }
```

5.6. Sensor (Camera)

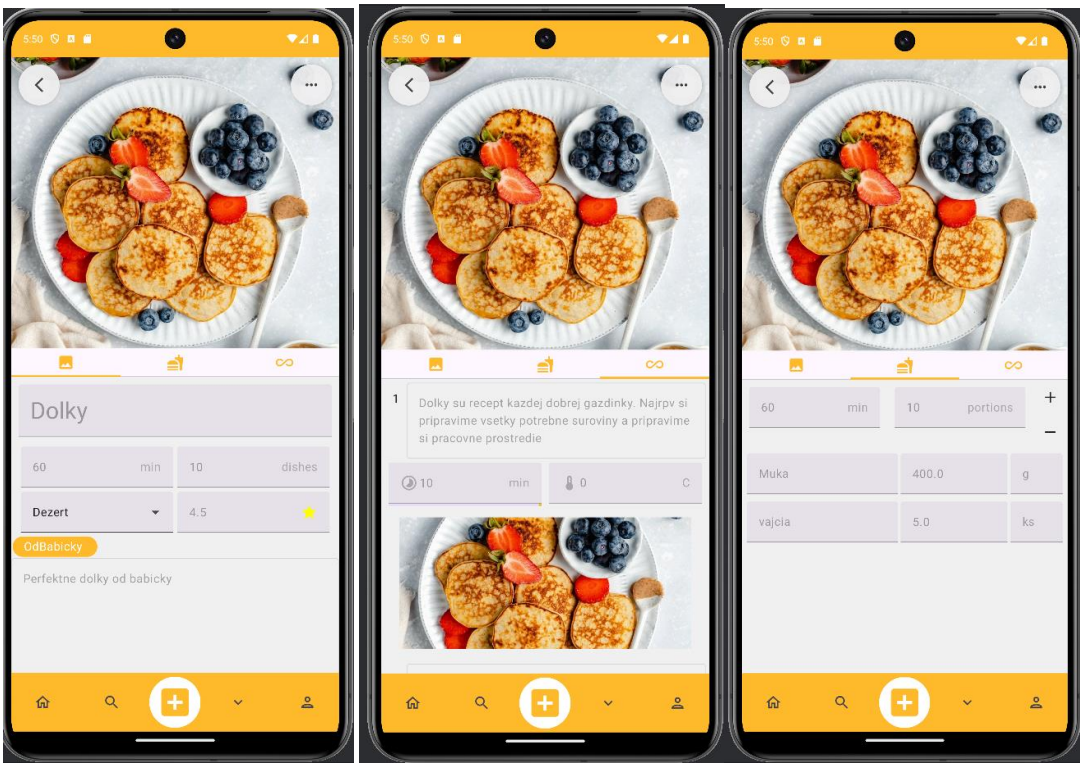
V jedno z mojich componentov využívam kameru na možnosť odfotenia a následného uloženia obrázku do interného úložiska na základe cesty. Toto odfotenie využívam na základe Intentu ktorý na mi vráti odfotený obrázok.

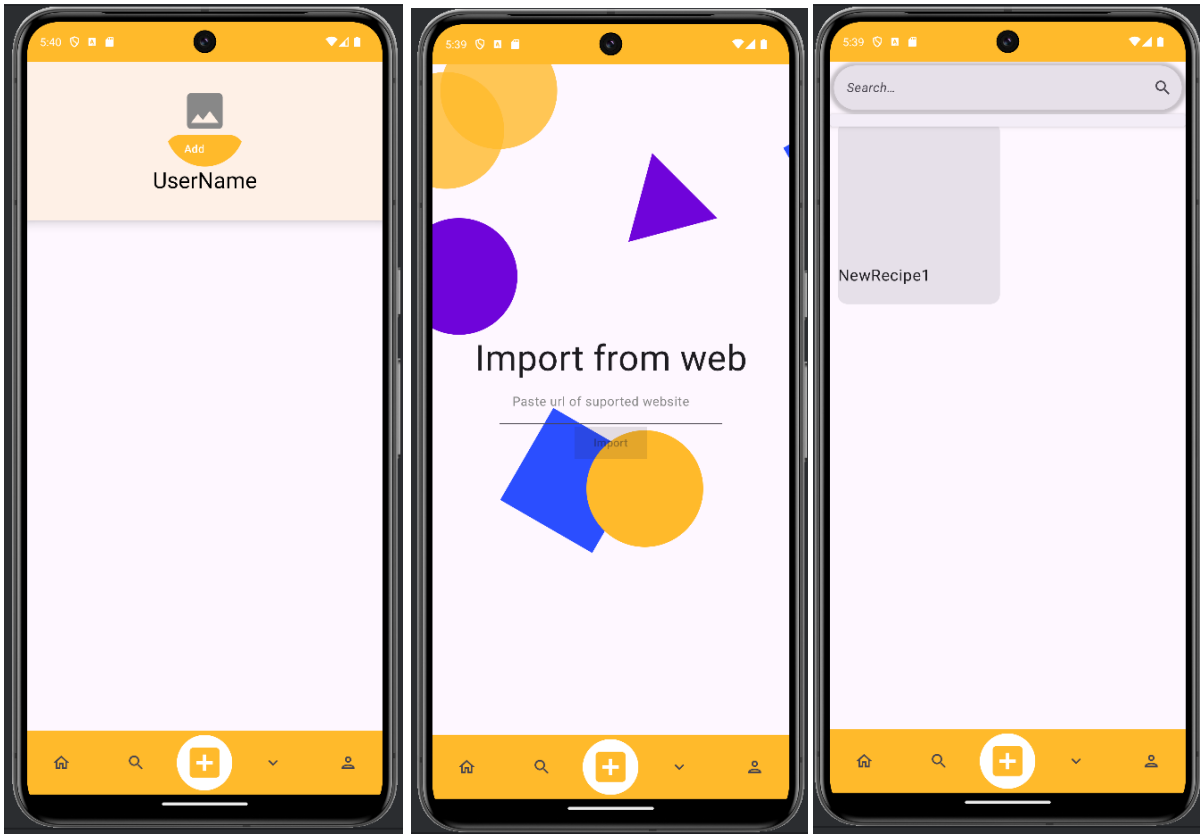
```
var cameraPickerActivity = rememberLauncherForActivityResult(
    contract = ActivityResultContracts.StartActivityForResult(),
    onResult = {result ->
        if (result.resultCode == Activity.RESULT_OK) {
            val data = tempUri?.let { it: Uri
                val source = ImageDecoder.createSource(context.contentResolver, it)
                ImageDecoder.decodeBitmap(source) ^let
            }
            if (data != null) {
                saveImage(context, path, data)
            }
            onImgChose?.invoke(data,  path: context.filesDir.path + path)
        }
        onImgChose?.invoke( newPath: null,  path: "")
        boolean = false
    }

)
```

5.7. Obrazovky

V celom mojom projekte je celkovo 5 obrazoviek ako je uvedené v návrhu. Každá z nich je navrhnutá tak že pokrýva kompletne celú obrazovku (Cez niektoré obrazovky je ukázaná navigačná lišta). Na každej obrazovke je aplikovaný návrhový model MVVM (Model view viewmodel).






5:39
Top Rated
Soup
NewRecipe1
MainDish
Dezert
BreakFast