ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu VAMZ

COOKBOOK+

vypracoval: Adriána Gemeľová

študijná skupina: 5ZYI21

cvičiaci: doc. Ing. Patrik Hrkút, PhD.

termín cvičenia: streda 7:00

v Žiline dňa 5. 4. 2024

Obsah

Úvod 2
1. Popis aplikácie 3
2. Spracovanie prehľadu dostupných aplikácií podobného zamerania 5
2.1 Paprika Recipe Manager 5
2.2 Mealime 5
2.3 Yummly 5
2.4 BigOven 5
3. Analýza navrhovanej aplikácie 6
3.1 Use case diagram 6
3.2 Funkcionality 6
3.3 Ukážka návrhu obrazoviek aplikácie 8
4. Implementácia 9
4.1 Viewmodel 9
4.2 Navigation 11
4.3 Room 11
4.4 Notifikácie 12
4.5 Aplikácia fotoaparátu 12
5. Výsledok vytvorenej aplikácie 12
5.1 Rozloženie aplikácie 12
5.2 Domovská obrazovka 13
5.3 Zoznam receptov 13
5.4 Rozkliknutý konkrétny recept 13
5.5 Nákupný zoznam a zoznam úloh 13
5.6 Výsledný dizajn aplikácie 14
Záver 17

Zoznam obrázkov

Obrázok 1 - BigOven 5
Obrázok 2 - Mealime 5
Obrázok 3 - Yummly 5
Obrázok 4 - Paprika Recipe Manager 5
Obrázok 5 - use case diagram aplikácie 6
Obrázok 6 - zobrazenie zoznamov 8
Obrázok 7 - úvodná obrazovka 8
Obrázok 8 - zobrazenie receptu 8
Obrázok 9 - zoznam receptov 8
Obrázok 10 - Hlavná obrazovka 14
Obrázok 11 - Obrazovka poznámok 14
Obrázok 12 - Obrazovka detailov receptu 15
Obrázok 13 - Obrazovka zoznamu receptov 15
Obrázok 14 - Obrazovka formulára 16

Úvod

Ako semestrálnu prácu z predmetu Vývoj aplikácií pre mobilné zariadenia som si vybrala vytvoriť prehľadnú a pre používateľa jednoduchú aplikáciu, ako pomôcku do kuchyne pre zariadenia Android. Mnohí (vrátane mňa samej) majú problém s pravidelným zdravým stravovaním, napríklad kvôli času, alebo poruchám pozornosti. Aplikácia COOKBOOK+ poskytuje používateľom možnosť organizácie jedál, rýchlemu uchovaniu receptov a postupov a prispieva k lepšiemu.

Aplikácia bude vytvorená v Android Studio v jazyku Kotlin. Návrhy jednotlivých obrazoviek sú vytvorené v nástroji Figma.

1. Popis aplikácie

Aplikácia COOKBOOK+ je mobilná aplikácia pre zariadenia Android navrhnutá na ukladanie receptov, plánovanie jedál a správu nákupných zoznamov a úloh. Používatelia môžu jednoducho pridávať, upravovať a mazať recepty, vytvárať a spravovať plány jedál. Okrem toho môžu používatelia spravovať úlohy spojené s prípravou jedál a vyhľadávať recepty. Aplikácia je navrhnutá tak, aby poskytovala používateľom jednoduché a efektívne riešenie pre organizáciu ich stravovacích návykov a zlepšenie riadenia ich kuchynskej rutiny.

2. Spracovanie prehľadu dostupných aplikácií podobného zamerania

2.1 Paprika Recipe Manager

Paprika je populárna aplikácia na správu receptov, ktorá umožňuje používateľom ukladať recepty z rôznych webových stránok, plánovať jedlá a vytvárať nákupné zoznamy.

2.2 Mealime

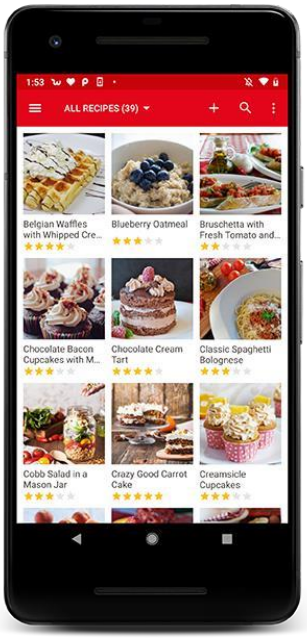
Aplikácia na plánovanie jedál, ktorá poskytuje používateľom prispôsobené jedálničky s receptami a nákupným zoznamom na základe ich preferencií a diétnych obmedzení.

2.3 Yummly

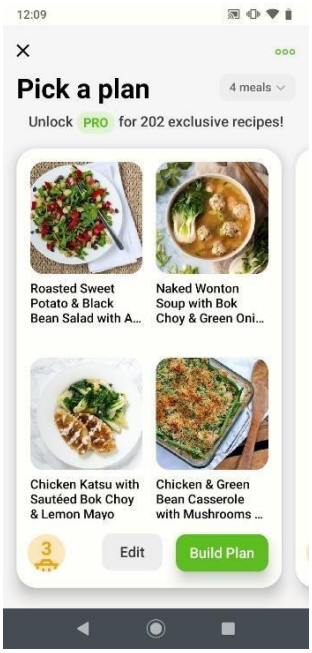
Yummly je platforma pre recepty, ktorá ponúka rozsiahlu databázu receptov z rôznych zdrojov. Používatelia môžu vyhľadávať recepty podľa rôznych kritérií a pridávať ich do vlastných zbierok.

2.4 BigOven

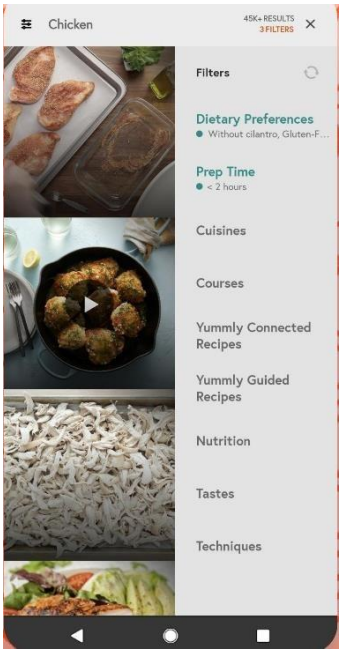
BigOven je aplikácia na ukladanie receptov, ktorá ponúka viac než 350 000 receptov, plánovanie jedál a možnosť vytvárania nákupných zoznamov na základe receptov.



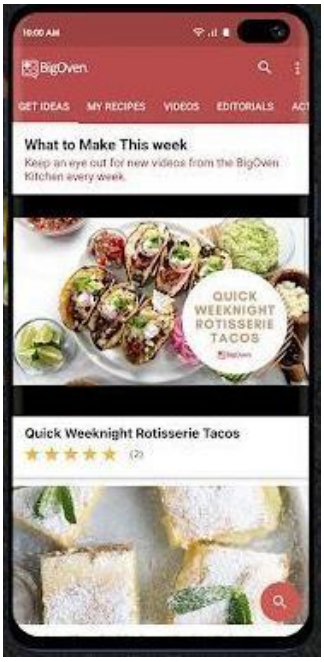
Obrázok 4 - Paprika Recipe Manager



Obrázok 2 - Mealime



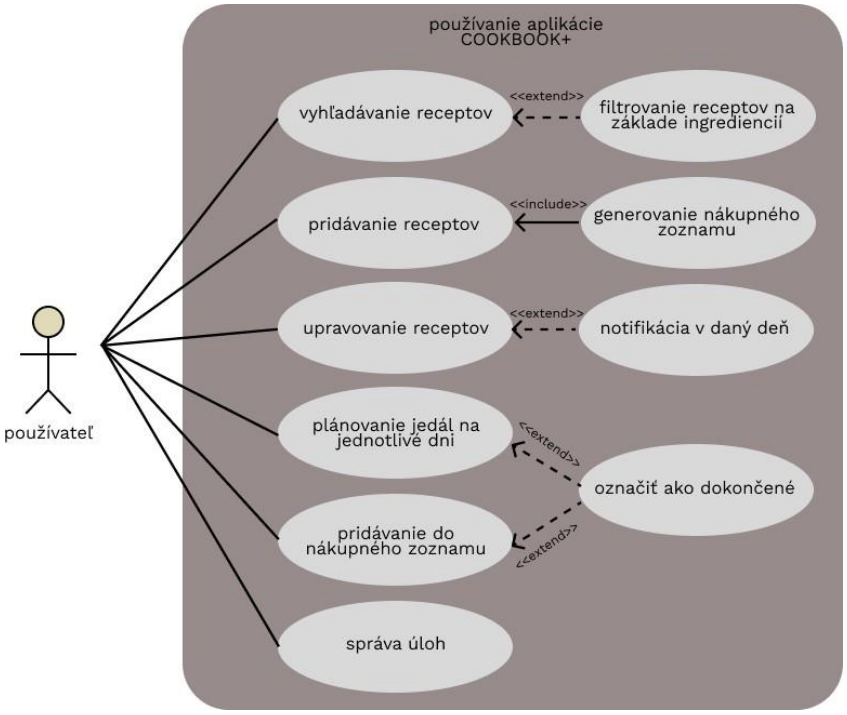
Obrázok 3 - Yummly



Obrázok 1 - BigOven

3. Analýza navrhovanej aplikácie

3.1 Use case diagram



Obrázok 5 - use case diagram aplikácie

3.2 Funkcionality

Správa receptov:

- Možnosť pridávať nové recepty do aplikácie.
- Zobrazenie detailov receptov, vrátane názvu, ingrediencií, postupu prípravy, dĺžky prípravy, počtu porcií a obrázkov.
- Možnosť upravovať a mazať existujúce recepty.

Plánovanie jedál:

- Funkcia pre naplánovanie ďalšieho jedla
- Po každom naplánovanom jedle príde používateľovi notifikácia

Nákupný zoznam:

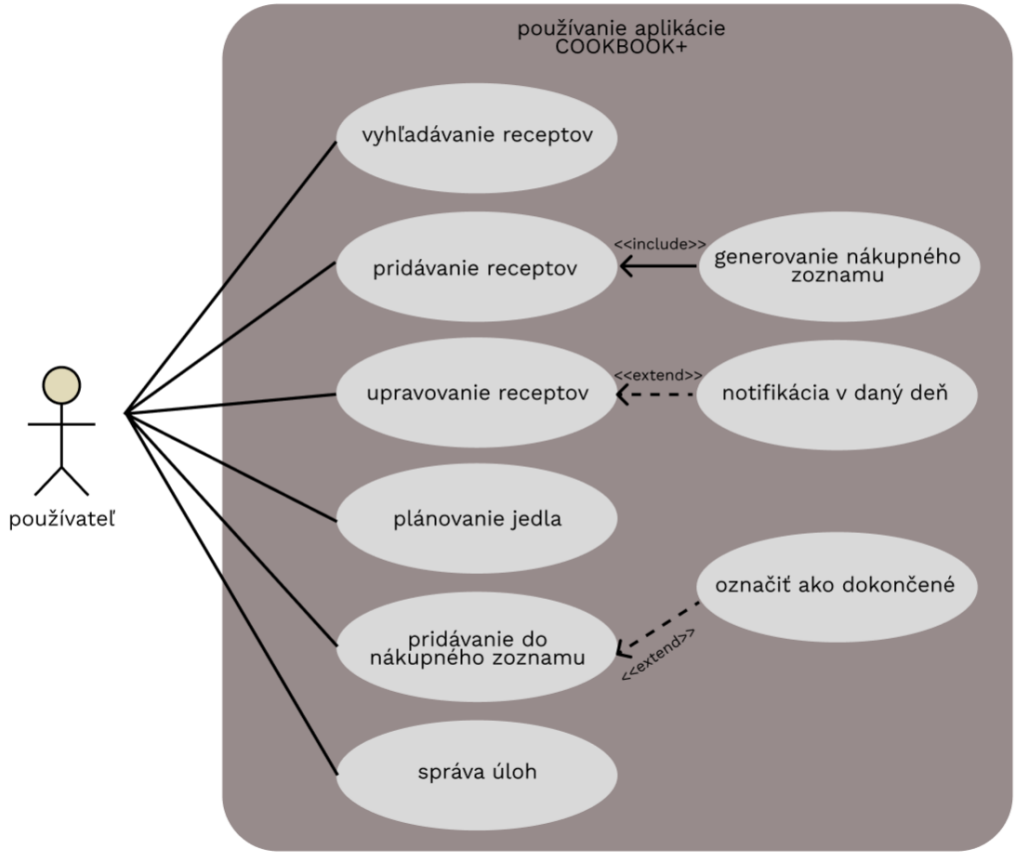
- Manuálna možnosť pridávať a odstraňovať položky z nákupného zoznamu.

Cook-do zoznam:

- Možnosť vytvárať a spravovať úlohy spojené s prípravou jedál.
- Zobrazenie aktuálnych úloh a ich stavu (napr. dokončené, nevykonané).

Vyhľadávanie a filtrovanie:

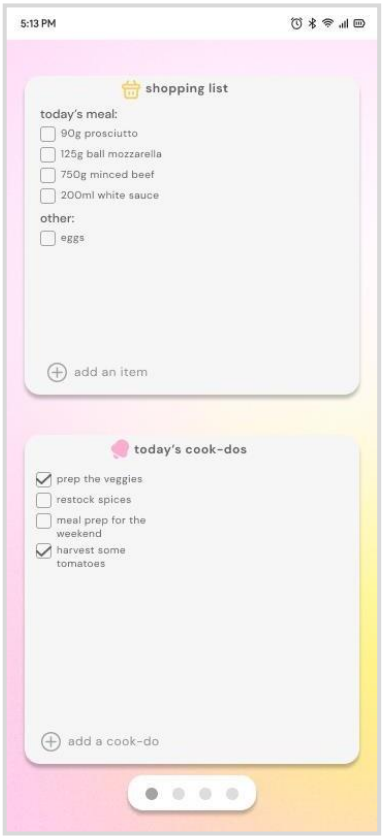
- Funkcia pre vyhľadávanie receptov podľa názvu



Obrázok 6 - Use case diagram výslednej aplikácie

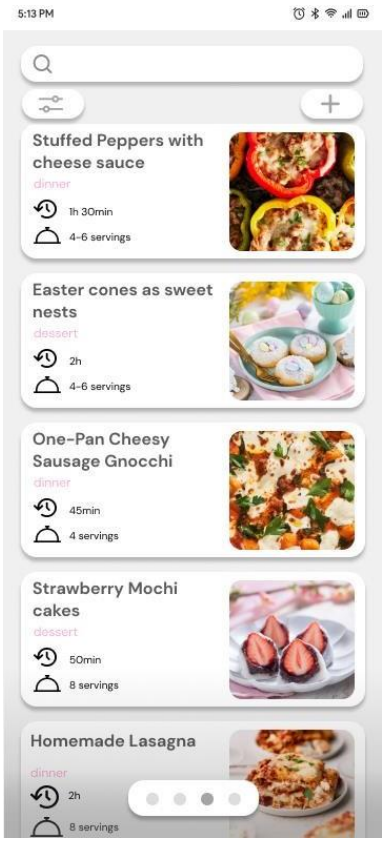
3.3 Ukážka pôvodného návrhu obrazoviek aplikácie

Obrázok 8 - úvodná obrazovka



Obrázok 7 - zobrazenie zoznamov

Obrázok 9 - zobrazenie receptu



Obrázok 10 - zoznam receptov

4. Implementácia

4.1 Viewmodel

V aplikácii sa celkovo nachádza 5 ViewModelov (MVWM). Takisto sa v projekte nacháza ViewModelFactory na ich vytváranie.

```
object ViewModelFactory : ViewModelProvider.Factory {

    private lateinit var application: Application
    lateinit var recipeRepository: RecipeRepository
    lateinit var noteRepository: NoteItemRepository

    AdrianaGemelova *
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return when {
            modelClass.isAssignableFrom(AddRecipeViewModel::class.java) -> {
                AddRecipeViewModel(recipeRepository) as T
            }
            modelClass.isAssignableFrom(RecipeListViewModel::class.java) -> {
                RecipeListViewModel(recipeRepository) as T
            }
            modelClass.isAssignableFrom(SharedViewModelMealCard::class.java) -> {
                SharedViewModelMealCard(application, recipeRepository) as T
            }
            modelClass.isAssignableFrom(SharedViewModel::class.java) -> {
                SharedViewModel(recipeRepository) as T
            }modelClass.isAssignableFrom(ShoppingListViewModel::class.java) -> {
                ShoppingListViewModel(noteRepository) as T
            }
            else -> throw IllegalArgumentException("Unknown ViewModel class")
        }
    }
```

ShoppingListViewModel:

```
AdrianaGemelova *
class ShoppingListViewModel(private val repository: NoteItemRepository) : ViewModel() {

    private val _shoppingItems = MutableStateFlow<List<NoteItem>>(emptyList())
    val shoppingItems: StateFlow<List<NoteItem>> = _shoppingItems

    private val _cookDoItems = MutableStateFlow<List<NoteItem>>(emptyList())
    val cookDoItems: StateFlow<List<NoteItem>> = _cookDoItems

    private var itemCount = 0
```

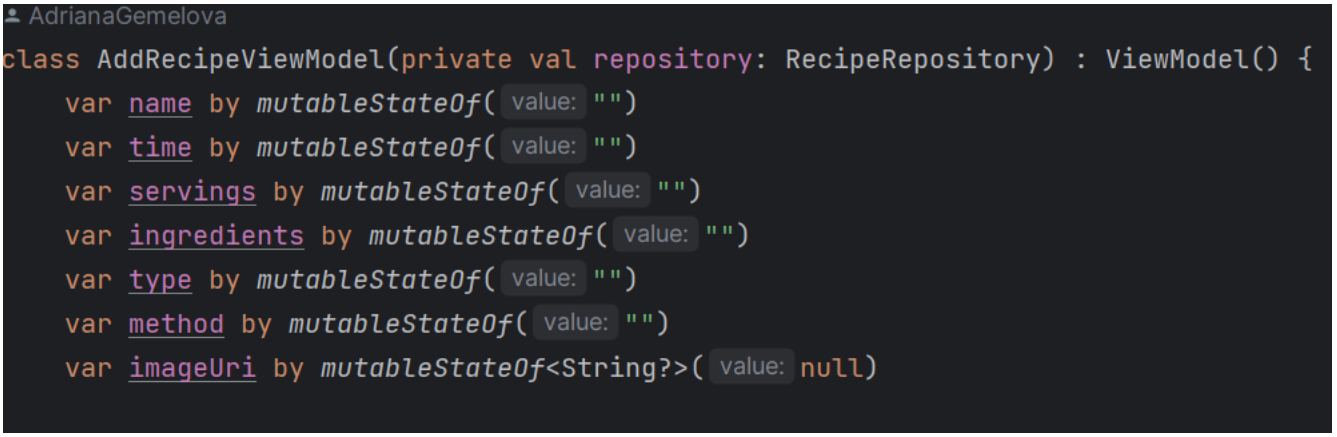
RecipeListViewModel:

```
class RecipeListViewModel(private val repository: RecipeRepository) : ViewModel() {
    private val _searchText = MutableStateFlow( value: "")
    val searchText: StateFlow<String> = _searchText

    val recipes: StateFlow<List<Recipe>> = _searchText
        .debounce( timeoutMillis: 300) // optimalizacia
        .flatMapLatest { searchText ->
            if (searchText.isEmpty()) {
                repository.getAllRecipes() ^flatMapLatest
            } else {
                repository.searchRecipesByName(searchText) ^flatMapLatest
            }
        }
        .stateIn(viewModelScope, SharingStarted.Lazily, emptyList())

    AdrianaGemelova
    fun onSearchTextChanged(newText: String) {
        _searchText.value = newText
    }
}
```

SharedViewModel – slúži na vymazanie a otvorenie konkrétneho receptu – na predávanie si zvoleného receptu

SharedViewModelMealCard – slúži na uchovávanie zvoleného najbližšieho jedlo, ktoré je zobrazené na hlavnej obrazovke. Takisto v sebe uchováva funkciu na posielanie notifikácií

AddRecipeViewModel – uchováva informácie o pridávanom/editovanom recepte



4.2 Navigation

Na navigáciu je použitý NavController.

```
@Composable
fun Navigation(navController: NavHostController, modifier: Modifier = Modifier) {
    val sharedViewModel: SharedViewModel = viewModel(factory = ViewModelFactory)
    val shoppingListViewModel: ShoppingListViewModel = viewModel(factory = ViewModelFactory)
    val sharedViewModelMealCard: SharedViewModelMealCard = viewModel(factory = ViewModelFactory)

    NavHost(navController = navController,
        startDestination = MainDestination.route,
        modifier = modifier
    ) { this: NavGraphBuilder
        composable(route = MainDestination.route) { this: AnimatedContentScope  it: NavBackStackEntry
            MainScreen(
```

4.3 Room

Vo svojej aplikácii používam ROOM databázu. Nachádzajú sa v nej dve tabuľky – Recipe a NoteItem.

```
@Entity(tableName = "note_items")
data class NoteItem(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val name: String,
    val isChecked: Boolean = false,
    val type: NoteType
)
```

```
@Entity(tableName = "recipe")
data class Recipe(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val name: String,
    val time: String,
    val servings: String,
    val ingredients: String,
    val type: String,
    val method: String,
    var isSelected: Boolean = false,
    val imageUri: String?
)
```

Databáza sa na začiatku naplní (populate) desiatimi ukážkovými receptami.
Trieda RecipeDatabase vytvorí databázu. Túto triedu som čiastočne prebrala z CodeLab-u, ktorý sme robili spoločne aj na cvičení.

4.4 Notifikácie

Aplikácia posiela používateľovi notifkáciu, keď si zvolí najbližšie jedlo – slúži to ako pripomienka.

```
1 AdrianaGemelova *
private fun sendNotification(context: Context, title: String, message: String) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        if (ActivityCompat.checkSelfPermission(context, Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {

            return
        }
    }

    val intent = Intent(context, MainActivity::class.java).apply { this: Intent
        flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    }
    val pendingIntent: PendingIntent = PendingIntent.getActivity(context, requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)

    val builder = NotificationCompat.Builder(context, channelId: "CHANNEL_ID")
        .setSmallIcon(R.drawable.ic_meal)
        .setContentTitle(title)
        .setContentText(message)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true)

    with(NotificationManagerCompat.from(context)) { this: NotificationManagerCompat
        notify( id: 1, builder.build())
    }

}
```

4.5 Zvolenie fotografie z interného úložiska zariadenia

Používateľ si môže k receptu zvoliť jednu fotografiu z interného úložiska

```
var imageUri by remember { mutableStateOf<Uri?>( value: null) }

val launcher = rememberLauncherForActivityResult(ActivityResultContracts.OpenDocument()) { uri: Uri? ->
    uri?.let { it: Uri
        context.contentResolver.takePersistableUriPermission(it, Intent.FLAG_GRANT_READ_URI_PERMISSION)
        imageUri = it
        viewModel.imageUri = it.toString()
    }
}
```

5. Výsledok vytvorenej aplikácie

5.1 Rozloženie aplikácie

Používateľské rozhranie sa skladá z 3 hlavných častí:

- Domovská obrazovka
- Zoznam receptov
- Nákupný zoznam + zoznam úloh
- Obrazovka detailov o recepte
- Formulár pre pridanie/úpravu receptu

Zároveň, kliknutím na akýkoľvek recept sa používateľ dostane k detailnému popisu receptu.

Po kliknutí tlačidla *edit,* alebo po kliknutí tlačidla + pre pridanie nového receptu sa

používateľovi zobrazí formulár pre nahranie informácií nového receptu, ako aj možnosť pridať fotku z galérie

5.2 Domovská obrazovka

Zobrazuje viacero informácií:

- Pozdrav pre používateľa
- Najbližšie naplánované jedlo
- skratku pre nákupný zoznam a zoznam úloh, kam vie používateľ rýchlo a jednoducho pridávať

5.3 Zoznam receptov

Posunutím doľava sa vie používateľ presunúť na zoznam receptov, ktoré môže filtrovať na základe názvu

5.4 Rozkliknutý konkrétny recept

Používateľ vidí detailné informácie o jedle, ako aj ingrediencie v bodoch, ktoré si dokáže jedným kliknutím pridať do nákupného zoznamu

5.5 Nákupný zoznam a zoznam úloh

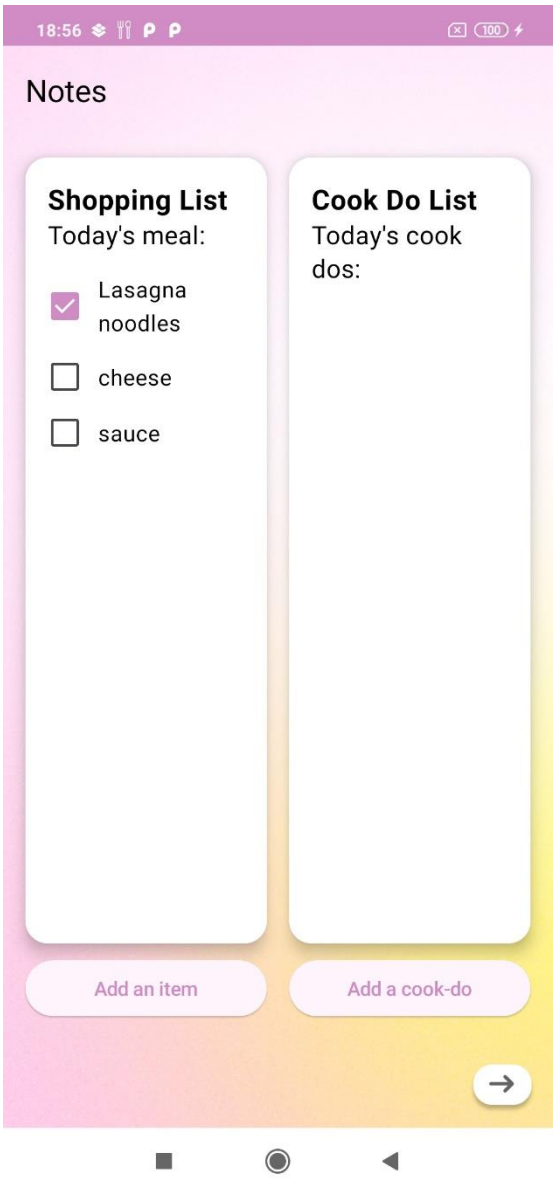
Posunutím doprava z domovskej obrazovky sa používateľ dostane k týmto zoznamom. Jednotlivé úlohy a položky nákupného zoznamu sa dajú pridávať, upravovať, mazať a označovať ako hotové.

5.6 Formulár pre pridanie/úpravu receptu

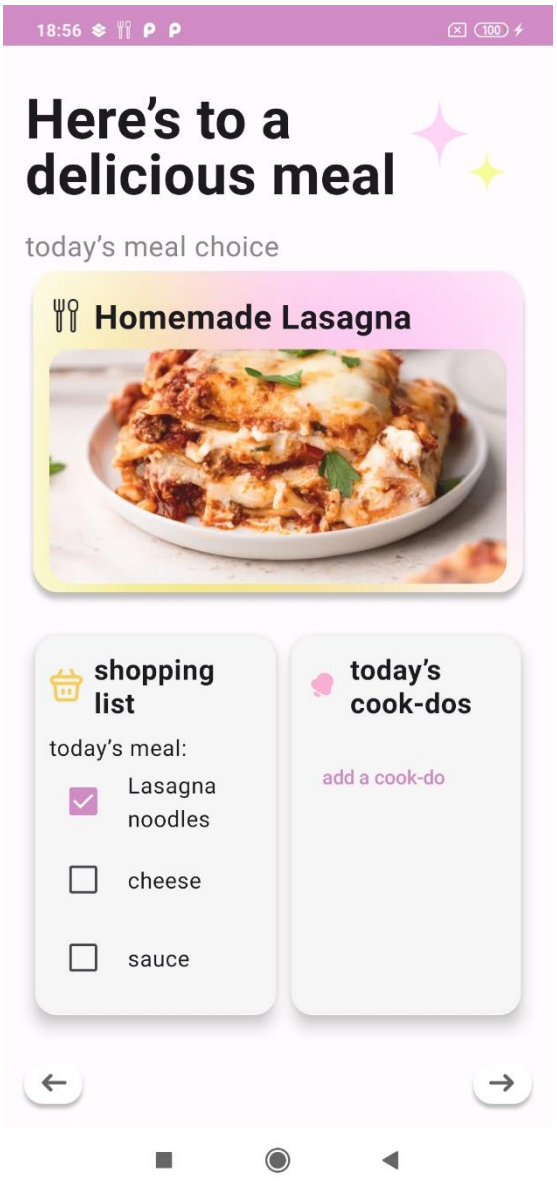
Používateľ dokáže pridať nový recept – vyplní potrebné informácie a zvolí obrázok. Ak žiadny nezvolí, k receptu sa priradí default obrázok. V postupe stačí pridať vety oddelené bodkou, tie sa automaticky pretvoria na kroky. Používateľ podobne dokáže upravovať existujúci recept.

.

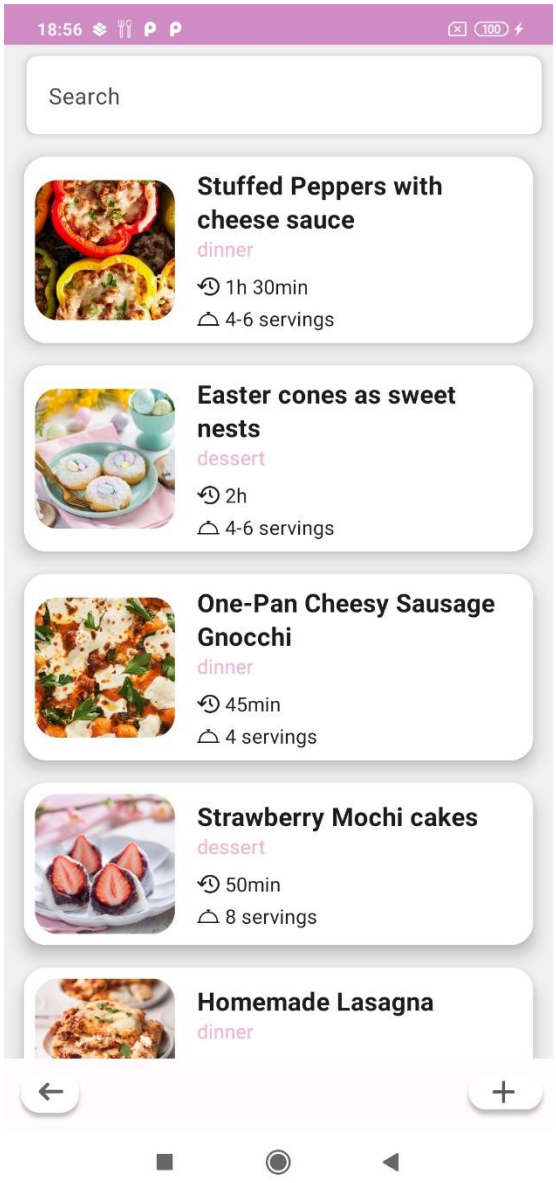
5.6 Výsledný dizajn aplikácie



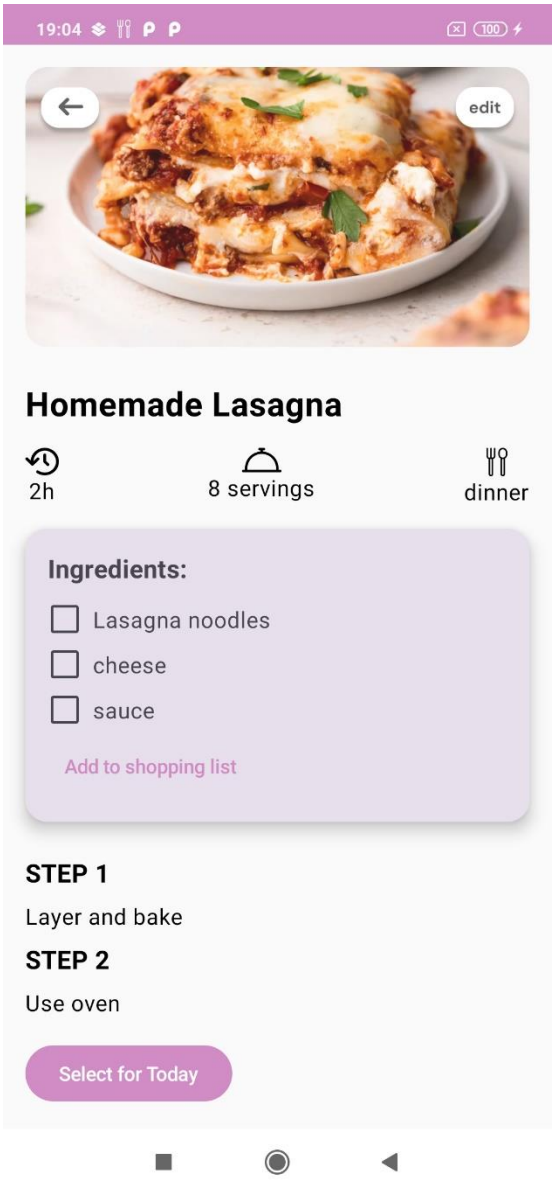
Obrázok 12 - Obrazovka poznámok



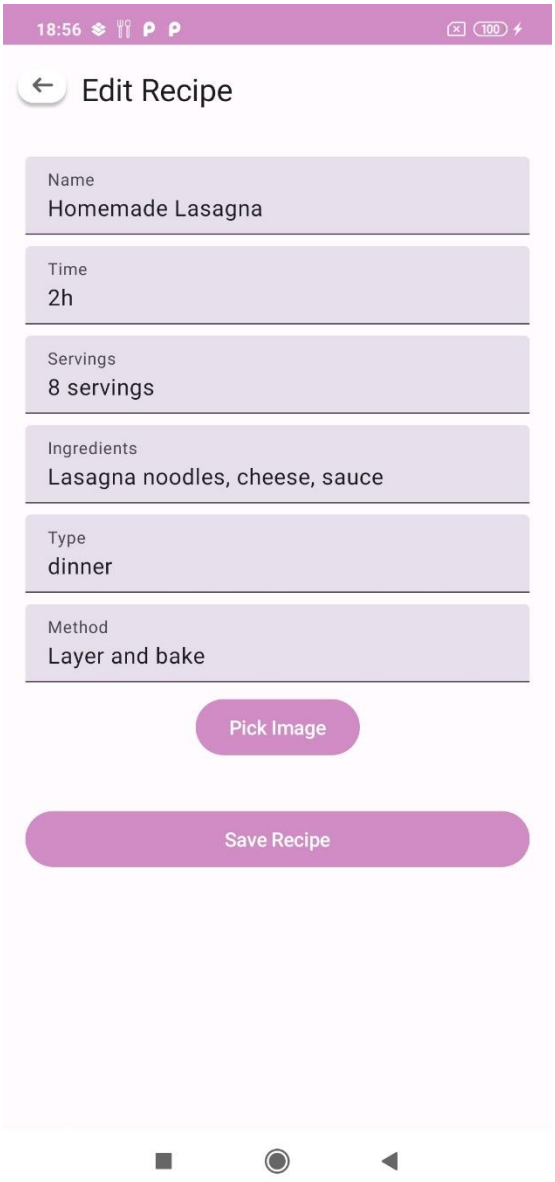
Obrázok 11 - Hlavná obrazovka



Obrázok 14 - Obrazovka zoznamu receptov



Obrázok 13 - Obrazovka detailov receptu



Obrázok 15 - Obrazovka formulára

Záver

Na záver navrhnutého konceptu aplikácie COOKBOOK+ by som chcela zdôrazniť jej potenciál ako užitočného nástroja pre organizáciu a zlepšenie stravovacích návykov používateľov.

Navrhnutá aplikácia sa snaží riešiť bežné problémy s pravidelným stravovaním a ponúka jednoduché a efektívne riešenia prostredníctvom svojich funkcionalít, ako je ukladanie receptov, plánovanie jedla a správa nákupných zoznamov a úloh. S dôrazom na jednoduchosť a intuitívne používateľské rozhranie, COOKBOOK+ môže byť skvelým spoločníkom pre všetkých, ktorí túžia po lepšej organizácii svojich stravovacích návykov a podporovaní zdravého životného štýlu.

Podarilo sa mi implementovať takmer všetko, čo som si určila v checkpointe, no zopár funkcionalít, ako filtrovanie na základe typu a času si nechám na ďalší beh.

Aplikácia teda poskytuje priestor pre ďalšie nadstavby a upravenia, ako napríklad automatický import receptov prostredníctvom URL, alebo pridaním funkcionality prihlásenia sa a zdieľania receptov medzi používateľmi, prípadne synchronizácia medzi zariadeniami.

Použité zdroje:

https://developer.android.com/develop/ui/views/notifications/build-notification

https://developer.android.com/develop/ui/compose/side-effects#launchedeffect

https://amitshekhar.me/blog/instant-search-using-kotlin-flow-operators

https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room#4