OSNOVE RAZVOJA WEB I MOBILNIH APLIKACIJA

LV 7: Navigacija unutar Jetpack Compose

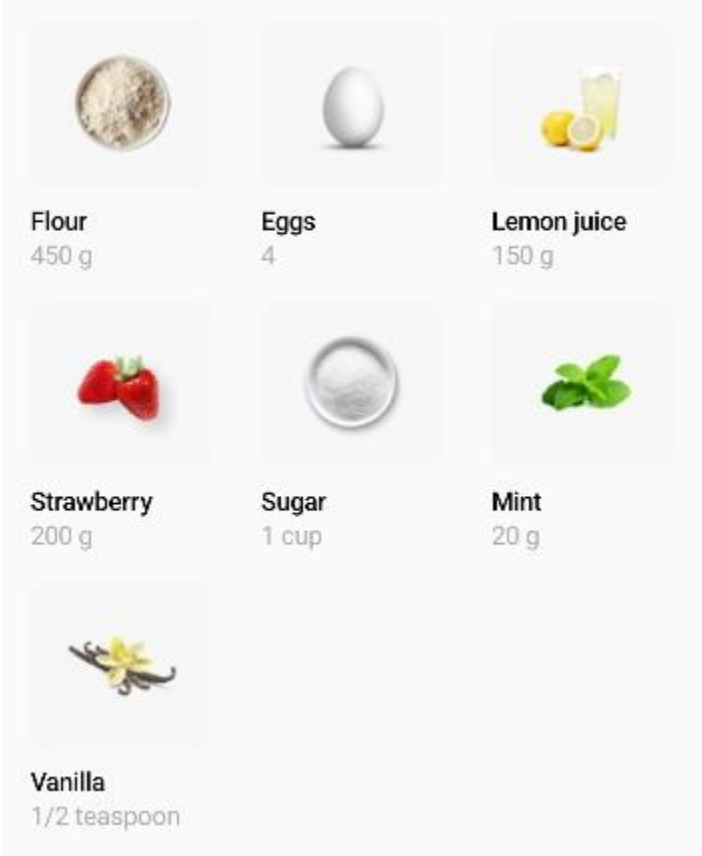
Osijek, 2023.

1. PRIPREMA

U ovoj vježbi upoznati ćete se s navigacijom između više zaslona. Na kraju ove laboratorijske vježbe napraviti ćete drugi zaslon Android aplikacije koji će se prikazivati kada kliknete na neki recept na prvom zaslonu.

1.1. Kartica sa informacijama o sastojcima

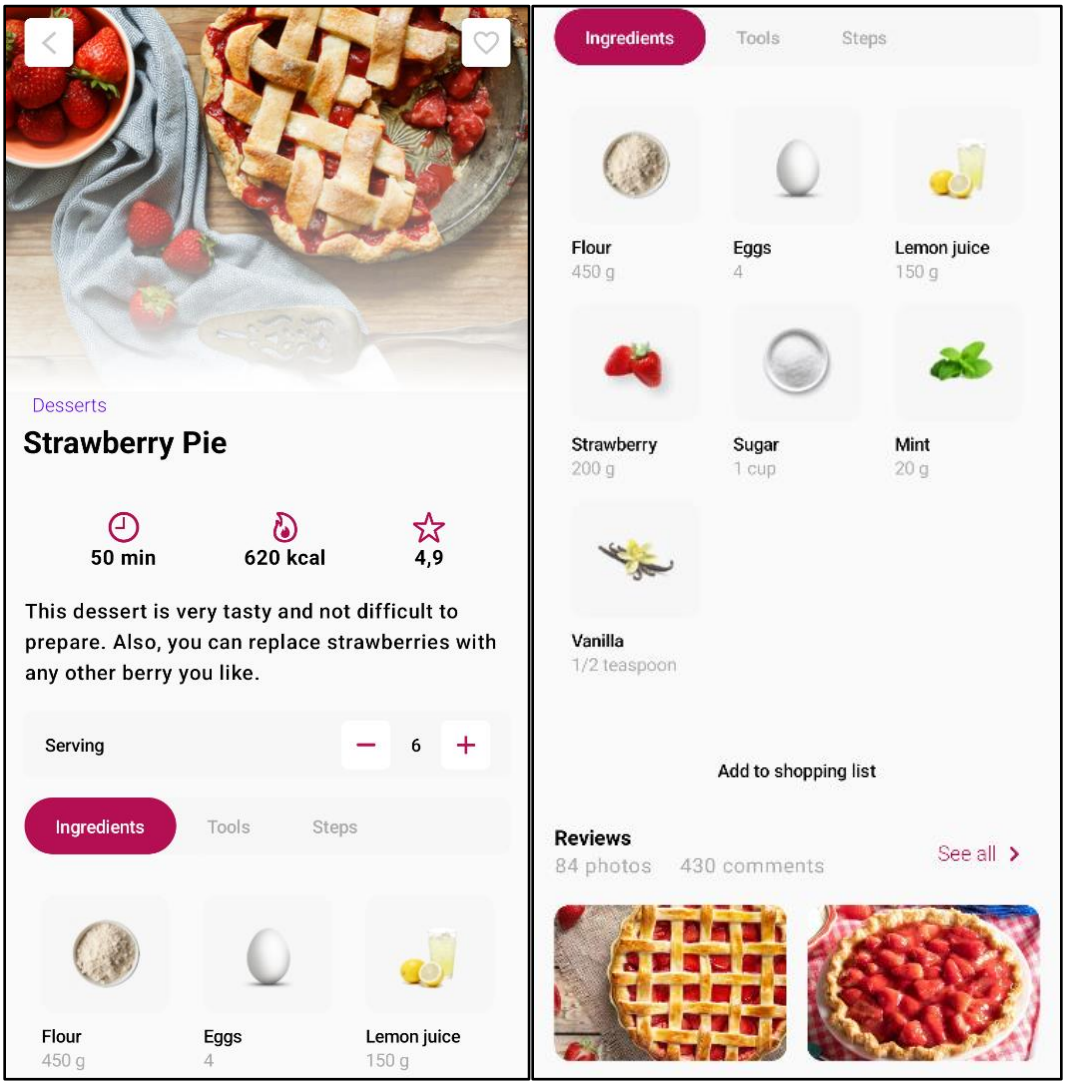
Potrebno je napraviti karticu s informacijama o sastojcima odabranog jela kada se otvori zaslon koji ćemo raditi u ovoj vježbi. Composable funkciju nazovite *IngredientCard* koja će za argumente primati: *iconResource* (@DrawableRes Int), *title* (String) i *subtitle* (String). Sam izgled kartica s informacijama o sastojcima prikazan je na slici 1.1. Slike sastojaka dostupne su na sljedećoj poveznici:



Slika 1.1. Prikaz dizajna kartica s informacijama o sastojcima jela.

2. RAD NA VJEŽBI

Preuzmite projekt s prošlih laboratorijskih vježbi i pokrenite ga unutar Android Studija i stvorite novi *RecipeDetailsScreen* datoteku unutar *ui* direktorija. Slika 2.1. prikazuje dizajn zaslona koji ćemo u ovoj vježbi napraviti.



Slika 2.1. Prikaz drugog zaslona aplikacije pod nazivom *RecipeDetailsScreen*.

2.1. Naslovna slika i gumbi

U gornjem dijelu aplikacije nalazi se slika i dva gumba (za povratak natrag i stavljanje recepta u favorite) koji umjesto teksta imaju samo ikonu. Napravite *CircularButton* Composable funkciju koja će imati sljedeće argumente:

- *iconResource* (@DrawableRes Int) – Ikona koja će se nalaziti unutar gumba
- *color* (Color) – Boja ikone unutar gumba
- *elevation* (ButtonElevation) – Dodaje sjenu ispod gumba tako da izgleda kao da je podignuto od zaslona
- *onClick* (() -> Unit) – Funkcija koja će se pozvati kada korisnik klikne na gumb

Funkcija je slična *IconButton* funkciji iz prošle laboratorijske vježbe samo nema dodatan Jetpack Compose element u sebi (*Text*). Prvi element unutar *IconButtona* je sam *Button* koji u argumentima funkcije ima upisan elevation koji je predefiniran, *contentPadding* koji će biti jednak *PaddingValues()*, *onClick* koji će pozvati *onClick()* funkciju iz argumenta Composable funkcije, *modifier* koji će odrediti širinu i dužinu samog gumba (38dp) i shape koji treba biti namješten na 5dp. Unutar *Button* elementa nalazi se *Icon* element koji će imati putem *painterResource* pozvati ikonu koja je predefinirana u *iconResoruce* argumentu. Programski kod 1 prikazuje gotov programski kod za *CircularButton* funkciju.

```
@Composable
fun CircularButton(
    @DrawableRes iconResource: Int,
    color: Color = Gray,
    elevation: ButtonElevation? =
ButtonDefaults.buttonElevation(defaultElevation = 12.dp),
    onClick: () -> Unit = {}
) {
    Button(
        contentPadding = PaddingValues(),
        elevation = elevation,
        onClick = { onClick() },
        colors = ButtonDefaults.buttonColors(containerColor = White,
contentColor = color),
        shape = RoundedCornerShape(5.dp),
        modifier = Modifier
            .width(38.dp)
            .height(38.dp)
    ) {
        Icon(
            painter = painterResource(id = iconResource),
            contentDescription = null,
        )
    }
}
```

Programski kod 1. Prikaz *CircularButton* Composable funkcije.

Temeljni (engl. *root*) element *TopImageAndBar* Composable funkcije je *Box* element koji će zauzeti cijelu širinu zaslona, ali samo 300dp visine. Jedini argument *TopImageAndBar* funkcije biti će slika *coverImage* (@DrawableRes Int) koja će služiti za dinamičan prikaz naslovne kliknutog recepta. Prvi element unutar *Box* elementa je *Image* koji će prikazati naslovnu sliku s *contentScale* argumentom stavljenim na *ContentScale.Crop* i modifikatorom za ispunjavanje cijelog elementa roditelja. Na samu naslovnu sliku ćemo prikazati tri stvari: dva gumba i fade out. Pošto se radi o vertikalnom pozicioniranju elemenata potrebno je postaviti Column element kojem moramo postaviti *verticalArrangement* na *Arrangement.SpaceBetween* i koji će zauzeti cijelu visinu elementa roditelja. Unutar *Column* elementa je potrebno postaviti *Row* element unutar kojeg stavljamo *CircularButton* elemente. *Row* element treba imati *verticalAlignment* na *Alignment.CenterVertically*, *horizontalArrangement* na *Arrangement.SpaceBetween* i s modifikatorima trebamo namjestiti da element ispuni cijelu širinu zaslona, da ima *statusBarPadding*, visinu od 56dp i horizontalni padding od 16dp. Na samom kraju funkcije pišemo programski kod za fade-out naslovne slike. Fade-out je običan vertikalni gradient od bijele

boje do transparetne boje. Unutar *Box* elementa definirati ćemo dva modifikatora gdje će prvi reći da će *Box* zauzeti maksimalno dozvoljenu veličinu zaslona (koju ograničava roditeljski element) i *background* će biti *Brush.verticalGradient()* gdje definiramo listu dvije boje ta naš gradient (ako želite više boja u gradientu onda definirate više boja u listi), također startY u *background* modifikatoru treba biti namješten na 100f. Programski kod 2 prikazuje završen kod za *TopImageAndBar* Composable funkciju.

```
@Composable
fun TopImageAndBar(
    @DrawableRes coverImage: Int
) {
    Box(
        modifier = Modifier
            .height(300.dp)
            .fillMaxWidth()
    ) {
        Image(
            painter = painterResource(id = coverImage),
            contentDescription = null,
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .fillMaxSize()
        )

        Column(
            verticalArrangement = Arrangement.SpaceBetween,
            modifier = Modifier
                .fillMaxHeight()
        ) {
            Row(
                verticalAlignment = Alignment.CenterVertically,
                horizontalArrangement = Arrangement.SpaceBetween,
                modifier = Modifier
                    .fillMaxWidth()
                    .statusBarsPadding()
                    .height(56.dp)
                    .padding(horizontal = 16.dp)
            ) {
                CircularButton(R.drawable.ic_arrow_back)
                CircularButton(R.drawable.ic_favorite)
            }
```

```
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .background(
                        Brush.verticalGradient(
                            colors = listOf(
                                Color.Transparent,
                                Color.White
                            ),
                            startY = 100f
                        )
                    )
            )
        }
    }
}
```

Programski kod 2. Prikaz *TopImageAndBar* Composable funkcije.

2.2. Naslov i osnovne informacije o jelu

Nakon naslovne slike i gumbova za povratak natrag i dodavanje odabranog jela u favorite potrebno je dodati i osnovne informacije o samom jelu. U Composable funkciji *ScreenInfo* definirati će se kategorija i naziv odabranog jela, a argumenti funkcije su: *title* (String) i *category* (String). Kako tekstovi trebaju biti jedan ispod drugog koristiti će se *Column* element unutar kojeg treba staviti dva *Text* elementa. *Text* element koji prikazuje kategoriju odabranog jela treba biti ljubičaste boje (Purple500), 15sp veličine i tankog fonta uz modifikator za horizontalni padding od 16dp. Text element koji prikazuje naziv odabranog jela treba biti crn boje (*Color.Black*), 24sp veličine i boldan font uz horizontalni modifikator od 16dp. Programski kod 3 prikazuje *ScreenInfo* funkciju.

```
@Composable
fun ScreenInfo(
    title: String,
    category: String
) {
    Column {
        Text(
            text = category,
            style = TextStyle(color = Purple500, fontSize = 15.sp,
fontWeight = FontWeight.Light),
            modifier = Modifier
                .padding(horizontal = 16.dp)
        )

        Text(
            text = title,
            style = TextStyle(color = Color.Black, fontSize = 24.sp,
fontWeight = FontWeight.Bold),
            modifier = Modifier
                .padding(horizontal = 16.dp)
        )
    }
}
```

Programski kod 3. Prikaz *ScreenInfo* Composable funkcije.

Prema dizajnu nakon kategorije i naziva odabranog jela idu osnovne informacije vezane uz samo jelo poput: vrijeme trajanja pripreme jela, kalorije koje odabrano jelo ima i ocijenu ostalih korisnika. Za potrebe prikaza tih informacija morati ćemo napraviti dvije Composable funkcije: *InfoColumn* i *BasicInfo*. *InfoColumn* sastoji se od *Column* elementa koji ima *horizontalAlignment* namješten na *Alignment.CenterHorizontally*, a u sebi sadrži *Icon* element koji prikazuje *iconResource* argument i *Text* koji prikazuje *text* argument funkcije. *Icon* element treba biti ružičaste boje (*Pink*) i s visinom od 24dp, a *Text* element mora biti boldan. *BasicInfo* Composable funkcija će imati jedan argument *recipe* (*Recipe*) koji će prihvaćati, a unutar nje moramo definirati *Row* element s *horizontalArrangement* postavljenim na *Arrangement.SpaceEvenly* i modifikatorima reći da želimo ispuniti cijelu širinu zaslona i dodati padding samo prema gore od 16dp. Unutar *Row* elementa postavite tri *InfoColumn* elementa za vrijeme trajanja kuhanja, broj kalorija koliko jelo ima i ocijenu recepta. Programski kod 4. prikazuje *InfoColumn* i *BasicInfo* Composable funkcija.

```
@Composable
fun InfoColumn(
    @DrawableRes iconResource: Int,
    text: String
) {
    Column(
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Icon(
            painter = painterResource(id = iconResource),
            contentDescription = null,
            tint = Pink,
            modifier = Modifier.height(24.dp)
        )
        Text(text = text, fontWeight = FontWeight.Bold)
    }
}

@Composable
fun BasicInfo(recipe: Recipe) {
    Row(
        horizontalArrangement = Arrangement.SpaceEvenly,
        modifier = Modifier
            .fillMaxWidth()
            .padding(top = 16.dp)
    ) {
        InfoColumn(R.drawable.ic_clock, recipe.cookingTime)
        InfoColumn(R.drawable.ic_flame, recipe.energy)
        InfoColumn(R.drawable.ic_star, recipe.rating)
    }
}
```

Programski kod 4. Prikaz *BasicInfo i InfoColumn* Composable funkcija.

U *Description* Composable funkcija će imati *recipe* (Recipe) argument koji će se iskoristiti za prikaz opisa recepta na zaslon. *Text* element treba imati srednju debljinu fonta (*FontWeight.Medium*) i vertikalni i horizontalni padding od 16dp i 20dp.

```
@Composable
fun Description(
    recipe: Recipe
) {
    Text(
        text = recipe.description,
        fontWeight = FontWeight.Medium,
        modifier = Modifier
            .padding(horizontal = 16.dp, vertical = 20.dp)
    )
}
```

Programski kod 5. Prikaz *Description* Composable funkcije.

Servings Composable funkcija prikazivati će jednostavan brojač gdje će se klikom na jedan od dva CircularButtona vrijednost brojača povećati ili smanjiti. Text element u između dva CircularButtona prikazuju trenutnu vrijednost brojača. Prvi Text element koji prikazuje tekst „Serving“ ima Modifier.weight(1f) modifikator koji će zauzeti cijeli prostor koji je slobodan.

```
@Composable
fun Servings() {
    var value by remember {
        mutableStateOf(6)
    }

    Row(
        verticalAlignment = Alignment.CenterVertically,
        modifier = Modifier
            .padding(horizontal = 16.dp, vertical = 0.dp)
            .clip(RoundedCornerShape(6.dp))
            .background(LightGray)
            .padding(horizontal = 16.dp)
    ) {
        Text(
            text = "Serving",
            modifier = Modifier.weight(1f),
            style = TextStyle(fontWeight = FontWeight.Medium)
        )
        CircularButton(iconResource = R.drawable.ic_minus, elevation =
null, color = Pink) {
            value--
        }
        Text(
            text = "$value",
            modifier = Modifier
                .padding(16.dp),
            style = TextStyle(fontWeight = FontWeight.Medium)
        )
        CircularButton(iconResource = R.drawable.ic_plus, elevation =
null, color = Pink) {
            value ++
        }
    }
}
```

Programski kod 6. Prikaz *Description* Composable funkcije.

2.3. Popis sastojaka odabranog jela

Korištenjem *EasyGrid* Composable funkcije koja je priložena u programskom kodu 7 i korištenjem *IngredientCard* Composable funkcije koje ste morali napraviti u pripremi napraviti ćemo popis sastojaka. *IngredientsList* Composable funkcija kao argument primati će *recipe*

(Recipe) iz kojeg će dohvaćati *ingredients* svojstvo podatkovne klase koji će se koristiti pri prikazu *IngredientCard* kartica unutar *EasyGrid* elementa. *EasyGrid* prihvaća dva argumenta *nColumns* koji govori *EasyGrid* elementu koliko će stupaca biti, a *items* argument govori *EasyGrid* elementu koji su podatci koje mora prikazati na zaslonu aplikacije. Prije samog popisa sastojaka potrebno je napraviti *IngredientsHeader* Composable funkciju koja će biti zapravo kopija *RecipeCategories* funkcije iz prošle vježbe.

```
@Composable
fun <T> EasyGrid(nColumns: Int, items: List<T>, content: @Composable (T) -
> Unit) {
    Column(Modifier.padding(16.dp)) {
        for (i in items.indices step nColumns) {
            Row {
                for (j in 0 until nColumns) {
                    if (i + j < items.size) {
                        Box(
                            contentAlignment = Alignment.TopCenter,
                            modifier = Modifier.weight(1f)
                        ) {
                            content(items[i + j])
                        }
                    } else {
                        Spacer(Modifier.weight(1f, fill = true))
                    }
                }
            }
        }
    }
}
```

Programski kod 7. Prikaz *IngredientsList* Composable funkcije.

Unutar kopiranog programskog koda iz prošle vježbe promjenite naziv funkcije i tekstove *TabButton* elemenata unutra (iz „All“, „Breakfast“, „Lunch“ u „Ingredients“, „Tools“ i „Steps“) . Programski kod 8 prikazuje gotovu *IngredientsList* i *IngredientsHeader* Composable funkcije.

```
@Composable
fun IngredientsHeader() {
    var currentActiveButton by remember { mutableStateOf(0) }

    Row(
        verticalAlignment = Alignment.CenterVertically,
        modifier = Modifier
            .padding(horizontal = 16.dp, vertical = 16.dp)
            .clip(RoundedCornerShape(8.dp))
            .background(LightGray)
            .fillMaxWidth()
            .height(44.dp)
    ) {
        TabButton("Ingredients", currentActiveButton == 0) {
            currentActiveButton = 0
        }
        TabButton("Tools", currentActiveButton == 1){
            currentActiveButton = 1
        }
        TabButton("Steps", currentActiveButton == 2) {
            currentActiveButton = 2
        }
    }
}


@Composable
fun IngredientsList(
    recipe: Recipe
) {
    EasyGrid(nColumns = 3, items = recipe.ingredients) {
        IngredientCard(it.image, it.title, it.subtitle)
    }
}
```

Programski kod 8. Prikaz *IngredientsList* Composable funkcije.

2.4. Donji dio zaslona

Prema dizajnu potrebno je dodati gumb koji će biti prikazan na zaslonu između popisa sastojaka za odabrano jelo i ocijena. *ShoppingListButton* Composable funkcija biti će samo jedan *Button* element koji će imati prozirnu pozadinu, a boja teksta će biti postavljena na crnu, a modifikatorima ćemo zadati da *Button* element zauzme maksimalnu širinu zaslona sa paddingom od 16dp. Tekst unutar samog gumba ćemo prikazati s *Text* elementom koji će imati padding od 8dp.

```
@Composable
fun ShoppingListButton() {
    Button(
        onClick = { /*TODO*/ },
        elevation = null,
        colors = ButtonDefaults.buttonColors(
            containerColor = Color.Transparent,
            contentColor = Color.Black
        ),
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp)
    ) {
        Text(text = "Add to shopping list", modifier =
Modifier.padding(8.dp))
    }
}
```

Programski kod 9. Prikaz *ShoppingListButton* Composable funkcije.

Donji dio zaslona aplikacije sadrži i ocijene za odabrani recept. Ocijene će biti sadržane unutar *Reviews* Composable funkcije koja kao argument će prihvaćati *recipe* (*Recipe*) objekt. Prema dizajnu elementi unutar *Reviews* elementa su poredani u redak kojem je širina zapravo maksimalna širina zaslona s paddingom na početku od 16dp, a *horizontalArrangement* je postavljen na *Arrangement.SpaceBetween*. Kako bi prikazali tekstove „Reviews“ i brojčani prikaz fotografija i komentara jedan ispod drugog koristiti ćemo *Column* element gdje ćemo *fontSize* prvog *Text* elementa staviti na 16sp i podebljani font, a boja drugog *Text* elementa će biti *DarkGray*. Na desnoj strani *Row* elementa nalazi se gumb s ikonom, a za to ćemo iskoristiti već napisanu *IconButton* Composable funkciju samo će side argument postavljen na 1. Programski kod 9 prikazuje *Reviews* Composable funkciju. Nakon što implementirate *Reviews* Composable funkciju potrebno je u *RecipesScreen* datoteci izmjeniti *IconButton* i dodati colors i side argumente. Programski kod 10 prikazuje izmjenjenu Composable funkciju unutar RecipesScreen datoteke.

```
@Composable
fun IconButton(
    @DrawableRes iconResource: Int,
    text: String,
    colors: ButtonColors = ButtonDefaults.buttonColors(containerColor =
Pink),
    side: Int = 0
) {
    Button(
        onClick = { /*TODO*/ },
        colors = colors,
    ) {
        Row {
            if (side == 0) {
                Icon(
                    painter = painterResource(id = iconResource),
                    contentDescription = text
                )
                Spacer(Modifier.width(2.dp))
                Text(
                    text = text,
                    style = TextStyle(
                        fontSize = 16.sp,
                        fontWeight = FontWeight.Light
                    )
                )
            }
            else {
                Text(
                    text = text,
                    style = TextStyle(
                        fontSize = 16.sp,
                        fontWeight = FontWeight.Light
                    )
                )
                Spacer(Modifier.width(2.dp))
                Icon(
                    painter = painterResource(id = iconResource),
                    contentDescription = text
                )
            }
        }
    }
}
```

Programski kod 10. Prikaz *IconButton* Composable funkcije (*RecipesScreen*).

Naposlijetku potrebno je napraviti *OtherRecipes* Composable funkciju koja će prikazati dvije slike jednu pored druge unutar *Row* elementa. *Row* element mora zauzimati maksimalnu širinu zaslona s paddingom od 16dp, a *Image* elementi imati će *weight(1f)* kako bi zauzeli

maksimalan prostor unutar *Row* elementa i *clip(RoundedCornerShape(12dp))* kako bi imale oble rubove.

```
@Composable
fun Reviews(recipe: Recipe) {
    Row(
        horizontalArrangement = Arrangement.SpaceBetween,
        modifier = Modifier
            .fillMaxWidth()
            .padding(PaddingValues(start = 16.dp))
    ) {
        Column {
            Text(text = "Reviews", style = TextStyle(fontSize = 16.sp,
fontWeight = FontWeight.Bold))
            Text(text = recipe.reviews, color = DarkGray)
        }

        IconButton(
            iconResource = R.drawable.ic_arrow_right,
            text = "See all",
            colors = ButtonDefaults.buttonColors(containerColor =
Transparent, contentColor = Pink),
            side = 1
        )
    }
}
```

Programski kod 11. Prikaz *Reviews* Composable funkcije.

```
@Composable
fun OtherRecipes() {
    Row(
        horizontalArrangement = Arrangement.SpaceBetween,
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp)
    ) {
        Image(
            painter = painterResource(id = R.drawable.strawberry_pie_2),
            contentDescription = "Strawberry Pie",
            modifier = Modifier
                .weight(1f)
                .clip(RoundedCornerShape(12.dp))
        )
        Spacer(modifier = Modifier.width(16.dp))
        Image(
            painter = painterResource(id = R.drawable.strawberry_pie_3),
            contentDescription = "Strawberry Pie",
            modifier = Modifier
                .weight(1f)
                .clip(RoundedCornerShape(12.dp))
        )
    }
}
```

Programski kod 12. Prikaz *OtherRecipes* Composable funkcije.

Nakon što su se sve Composable funkcije definirale potrebno je iste funkcije redom pozvati unutar *LazyColumn* elementa gdje će *verticalArrangement* biti postavljen na *Arrangement.Top*, *horizontalAlignment* biti postavljen na *Alignment.Start*, te će zauzeti maksimalno dozvoljenu veličinu zaslon. Argumenti unutra RecipeDetailsScreen Composable funkcije su *navigation* (NavController) i *recipeId* (Int) koji će služiti za navigaciju kroz samu aplikaciju i za prikaz odabranog recepta. Programski kod 13 prikazuje *RecipeDetailsScreen* Composable funkciju.

```
@Composable
fun RecipeDetailsScreen(
    navigation: NavController,
    recipeId: Int
) {
    val recipe = recipes[recipeId]
    val scrollState = rememberLazyListState()
    LazyColumn(
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.Start,
        state = scrollState,
        modifier = Modifier
            .fillMaxSize()
    ) {
        item {
            TopImageAndBar(
                coverImage = recipe.image,
                navigation = navigation
            )
            ScreenInfo(recipe.title, recipe.category)
            BasicInfo(recipe)
            Description(recipe)
            Servings()
            IngredientsHeader()
            IngredientsList(recipe)
            ShoppingListButton()
            Reviews(recipe)
            OtherRecipes()
        }
    }
}
```

Programski kod 13. Prikaz *RecipeDetailsScreen* Composable funkcije.

2.5. Navigacija unutar aplikacije

Navigacijska komponenta pruža jednostavan način navigacije unutar Android aplikacija. Ovo sučelje podržava niz konteksta i UI okvira, od Jetpack Composea i aktivnosti pa sve do Fragmenata i prilagođenih UI okvira (engl. *frameworks*). *NavController* je središnji navigacijski API. Prati koja je odredišta korisnik posjetio, a njegove metode omogućuju korisniku kretanje između odredišta. Kako biste koristili *NavController* morate dodati sljedeću liniju u *build.gradle* datoteku:

```
implementation("androidx.navigation:navigation-compose:2.7.4")
```

Možete imati više *NavControllera* s različitim funkcijama i za svaki *NavController* se izrađuje interni *NavHost* graf. *NavController* se stvara pomoću *rememberNavController()* funkcije koja vraća instancu *NavHostController* objekta. U direktoriju gdje se nalazi *MainActivity.kt* stvorite još jednu Kotlin datoteku pod nazivom *Navigation.kt* gdje ćemo stvoriti *NavigationController* Composable funkciju. Unutar *NavigationController* funkcije ćemo u varijablu pod nazivom *navController* spremiti instancu *NavHostController*. Zatim ćemo pomoću *NavHost* funkcije stvoriti navigacijski graf. Unutar nje ćemo definirati dvije *composable()* funkcije koje kao argument uzimaju putanju na koji će se pokrenuti programski kod unutar *composable()* funkcije i/ili argumenti koje želite prenijeti na drugi zaslon. *NavHost* povezuje *NavController* s navigacijskim grafom koji specificira zaslone (prilagođene Composable funkcije) između kojih bi korisnik aplikacije trebao moći navigirati. Dok se korisnik kreće kroz navigacijski graf sav sadržaj se dinamički rekomponira. Odredištu unutar navigacijskog grafa pristupa se putem *rute (route)*. *Ruta* je niz znakova koji definira putanju do vaše Composable funkcije. Možete to zamisliti kao implicitnu dubinsku vezu koja vodi do određenog odredišta. Svako odredište **mora** imati jedinstvenu rutu.

Prvi zaslon (*RecipesScreen*) pokretati će se kada aplikacija od navigacijske komponente zatraži „recipeList“ rutu. Također zadano početno odredište aplikacije će biti spomenuti zaslon. Drugi zaslon (*RecipeDetailsScreen*) pokretati će se kada aplikacija od navigacijske komponente zatraži „recipeDetails/{recipeId}“ rutu. Umjesto *{recipeId}* dinamično će se postavljati cjelobrojna vrijednost indeksa recepta unutar liste objekata. Za razliku od prvog zaslona moramo definirati i listu argumenata zajedno s njihovim nazivom i tipom. Argument *recipeId* će biti *NavType.IntType* jer će se zahtjevati cjelobrojna vrijednost (engl. *integer*). Za potrebe ovih laboratorijskih vježbi koristi se „primitivan“ pristup prosljeđivanja indeksa recepta jer se sam objekt klase ne može proslijediti na drugi zaslon tako lako. Composable funkcijama svih zaslona treba dodati argument pod nazivom navigation koji će čuvati instancu *NavHostController* objekta. Zbog lakšeg održavanja programskog koda potrebno je rute pohraniti kao varijable unutar *Routes* objekta. Osim varijabli do ruta potrebno je deklarirati i definirati *getRecipeDetailsPath()* funkciju koja će

Kako bi sve radilo potrebno je dodatno izmijeniti programski kod gdje će svi zasloni imati *navigation* argument i gdje će se unutar zaslona pozivati *NavControllerHost* funkcije za navigaciju na odredište ili povratak natrag u navigacijskom grafu. Unutar *RecipesScreen* Composable funkcije potrebno je *navigation* argument proslijediti *RecipeList* Composable funkciji. Kada korisnik klikne na *RecipeCard* okida se *onClick* event unutar kojeg je potrebno pozvati *navigation.navigate()* funkciju koja će uz *Routes.getRecipeDetailsPath()* funkciju prikazati drugi zaslon s detaljima o odabranom jelu. *Routes.getRecipeDetailsPath()* funkcija kao argument prihvaća indeks kliknutog recepta (jer izlaz *items()* funkcije je zapravo indeks objekta kojeg prikazuje na zaslonu), a kao rezultat vraća rutu koja je potrebna *navigation.navigate()* funkciji. Programski kod 14 prikazuje izmijenjenu *RecipeList* Composable funkciju.

```
object Routes {
    const val SCREEN_ALL_RECIPES = "recipeList"
    const val SCREEN_RECIPE_DETAILS = "recipeDetails/{recipeId}"

    fun getRecipeDetailsPath(recipeId: Int?) : String {
        if (recipeId != null && recipeId != -1) {
            return "recipeDetails/$recipeId"
        }
        return "recipeDetails/0"
    }
}

@Composable
fun NavigationController() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination =
Routes.SCREEN_ALL_RECIPES) {
        composable(Routes.SCREEN_ALL_RECIPES) {
            RecipesScreen(navigation = navController)
        }
        composable(
            Routes.SCREEN_RECIPE_DETAILS,
            arguments = listOf(
                navArgument("recipeId") {
                    type = NavType.IntType
                }
            )
        ) {backStackEntry ->
            backStackEntry.arguments?.getInt("recipeId")?.let {
                RecipeDetailsScreen(
                    navigation = navController,
                    recipeId = it
                )
            }
        }
    }
}
```

Programski kod 14. Prikaz *Navigation.kt* datoteke.

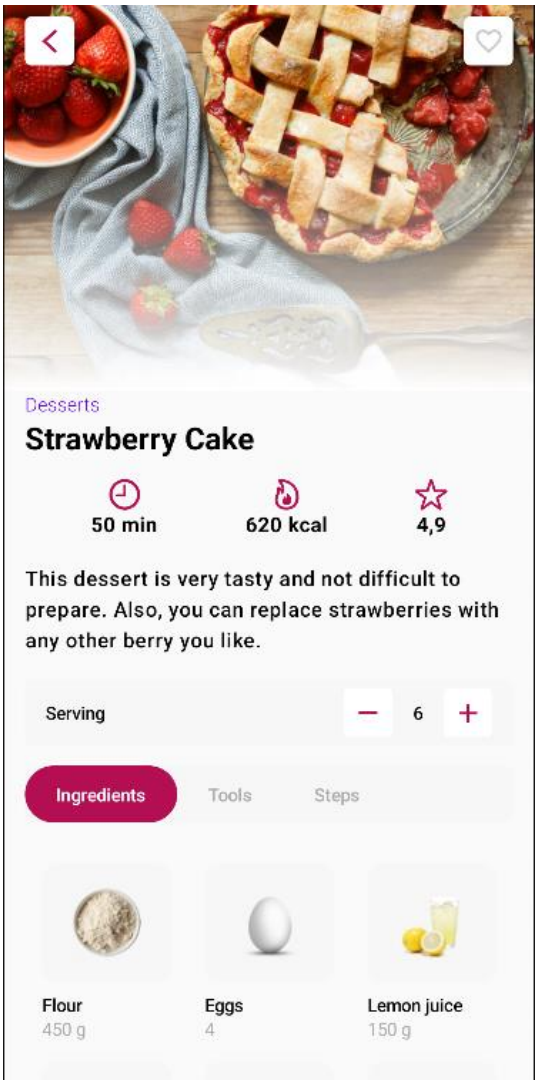
Unutar *RecipeDetailsScreen* zaslona potrebno je definirati dva argumenta: *navigation* (NavController) i *recipeId* (Int). Prije samog prikaza elemenata zaslona u *LazyColumn* spremite vrijednost odabranog recepta u *recipe* varijablu i zamijenite stare objekte klasa pod nazivom *strawberryCake* s novom varijablom.

```
@Composable
fun RecipeList(
    recipes: List<Recipe>,
    navigation: NavController
) {
    Column {
        Row(
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp, vertical = 12.dp)
        ) {
            Text(
                text = "7 recipes",
                style = TextStyle(color = Color.DarkGray, fontSize =
14.sp)
            )
            Icon(
                painter = painterResource(id = R.drawable.ic_flame),
                contentDescription = "Flame",
                tint = Color.DarkGray,
                modifier = Modifier
                    .width(18.dp)
                    .height(18.dp)
            )
        }
        LazyRow(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp)
        ) {
            items(recipes.size) {
                RecipeCard(
                    imageResource = recipes[it].image,
                    title = recipes[it].title
                ) {
                    navigation.navigate(
                        Routes.getRecipeDetailsPath(it)
                    )
                }
                Spacer(modifier = Modifier.width(8.dp))
            }
        }
    }
}
```

Programski kod 15. Prikaz izmijenjene *RecipeList* Composable funkcije.

3. ZADATAK ZA SAMOSTALAN RAD

Koristeći se funkcijama iz *NavController* klase omogućite povratak natrag na prethodni zaslon. Također potrebno je promijeniti boju gumba iz *DarkGray* u *Pink*. Slika 3.1 prikazuje kako zaslon treba izgledati.



Slika 3.1. Prikaz rezultata zadatka za samostalan rad.