ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu *Vývoj Aplikácií pre Mobilné Zariadenia*

FRI ROZVRH

vypracoval: Denis Vašek

študijná skupina: 5ZYI23

cvičiaci: doc. Ing. Patrik Hrkút PhD.

termín cvičenia: Pondelok 8:00 – 10:00

v Žiline dňa 10.6.2024

1. Popis a analýza riešeného problému

a. Špecifikácia zadania a definovanie problému

Ako moju semestrálnu prácu som sa rozhodol vytvoriť aplikáciu, ktorá bude slúžiť študentom ako rozvrh na FRIčke. Ako inšpiráciu som mal to, že na vzdelávaní nemám všetky voliteľné predmety a ani predmety v ktorých som si robil zmeny.

Vytvorenie tejto aplikácie by podľa mňa pomohlo študentom, lebo určite sa stretli s tým, že na vzdelávaní nemajú vypísané všetky predmety. A realizácia tejto aplikácie je pre mňa príležitosť získať skúsenosti v oblasti návrhu a vytvorenia softvérovej aplikácie.

Síce takáto aplikácia už existuje, aj tak by som si chcel naprogramovať svoju, lebo by som v budúcnosti chcel použiť podobné funkcionality v inej aplikácii.

b. Prehľad dostupných aplikácií

Po prieskume podobných dostupných aplikácií v službe Google Play som našiel:

- UNIZA Rozvrh
  - Umožňuje vytvoriť si rozvrh pridať učiteľa a učebňu
  - Vie posielať notifikácie
  - Vie vytvoriť widgety
  - Aplikácia je bezplatná

- My Classes – Timetable and Study
  - Obsahuje funkcie podobné aplikácii UNIZA Rozvrh
  - Aplikácia je bezplatná, ale obsahuje reklamy
  - Poskytuje pro verziu, tá reklamy neobsahuje

- Školní plánovač: rozvrh hodin
  - Funkcionality má veľmi podobné už spomenutých aplikácii
  - Má funkciu pre zapísanie si domácich úloh

Podobných aplikácií sa v službe Google Play nachádza veľké množstvo, ale vybral som tieto pre ukážku rôznych funkcionalít.

2. Návrh riešenia problému

a. Návrh aplikácie

Popis jednotlivých tried a funkcií:

MainActivity – Hlavná aktivita aplikácie, v ktorej sa inicializuje obsah hlavnej obrazovky, vykreslia sa hodiny, rozvrh a pridá sa tlačítko na uloženie obsahu. Taktiež vytvorí notifikačný kanál pre notifikácie rozvrhu.

Trieda Clock – Získava aktuálny čas a zobrazí ho vo forme digitálnych hodín. Používa **LaunchedEffect** aby sa hodiny aktualizovali každú sekundu a zobrazuje ich pomocou troch boxov.

Trieda TimeTable – Zobrazuje rozvrh s možnosťou kliknutia na jednotlivé bloky. Obsahuje metódy na ukladania a načítavanie rozvrhu pomocou **SharedPreferences**.

Triedy Subjects, SubjectType a LectureType – Reprezentujú enumy údajov, na základe ktorých sa vytvárajú bloky hodín do rozvrhu.

Trieda EditorViewModel – Spravuje stav formulára pre úpravu bloku rozvrhu. Umožňuje aktualizáciu jednotlivých vlastností bloku.

Funkcia Editor – Zobrazuje formulár pre úpravu jednotlivých vlastností bloku rozvrhu. Obsahuje interaktívne prvky pre výber predmetu, učiteľa, typu prednášky a miestnosti. Poskytuje tlačidlo pre uloženie zmien.

Funkca EditorScreen – Podporuje ukladanie zmien pomocou **ViewModel**. Ukladá zmeny pomocou **SavedStateHandle**.

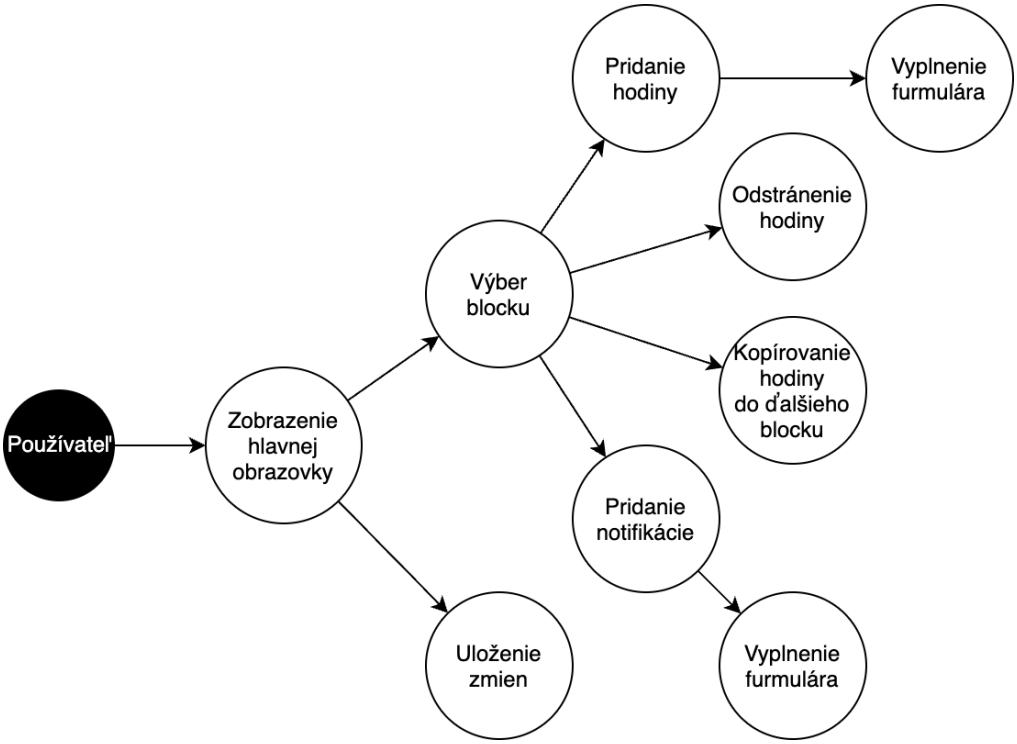
Trieda TimeTableNotificatioReciever – Receiver pre prijímanie broadcast správ pre upozornenia na rozvrh. V momente prijatia správy vytvorí inštanciu *TimeTableNotifications* a zobrazí upozornenie.

Trieda TimeTableNotifications – Slúži na zobrazovanie upozornení týkajúcich sa rozvrhu. Obsahuje metódu *showNotification*, ktorá vytvorí a zobrazí notifikáciu s danou správou a časom.

Funkcia NotificationScreen – Zobrazuje obrazovku pre pridanie nového upozornenia na rozvrh. Obsahuje formulár pre zadaní správy, času a dňa týždňa pre upozornenie.

b. Krátka analýza

UseCase diagram zobrauje hlavné funkcionality.



UseCase diagram

Popis diagramu:

Zobrazenie hlavnej obrazovky – Používateľovi sa zobrazí hlavná obrazovka s hodinami a rozvrhom.

Výber blocku – Používateľ si vyberie block z rozvrhu.

Pridanie hodiny – Používateľovi sa otvorí obrazovka s formulárom pre pridanie hodiny do rozvrhu.

Vyplnenie formulára – Používateľ vyplní formulár a následne pridá novú hodinu do blocku.

Odstránenie hodiny – Používateľ odstráni blok hodiny z rozvrhu.

Kopírovanie hodiny do ďalšieho bloku – Používateľ zduplikuje blok hodiny do vedlajšieho.

Pridanie notifikácie – Používateľovi sa otvorí obrazovka s formulárom pre naplánovanie notifikácie.

Vyplnenie formulára – Používateľ vyplní formulár a následne vyhodí notifikáciu.

Uloženie zmien – Zmeny sa uložia.

Aktéri:

Používateľ – Hlavný aktér, ktorý interaguje s prostredím.
Notifikačný systém – Systém, ktorý posiela do zariadenie notifikácie.

3. Popis implementácie

a. Reakcia na otočenie displeja:

Aplikácia je navhnutá tak, aby korektne reagovala na otáčanie displeja.

Použitie **ViewModel** tried zabezpečí aby sa pri zmene otočenia obrazovky zachovali údaje vyplnené vo formulári.

Použitie **rememberSaveable** premenných taktiež aby sa vyplnené polia nezmazali pri otočení obrazovky.

Použitie **LazyColumn** aby sa pri otočení obrazovky nepokazil layout a aby sa vedel používateľ dostať ku všetkým tlačidlám.

```
DonisPan *
class EditorViewModel(savedStateHandle: SavedStateHandle) : ViewModel() {
    DonisPan
    var subject: Subjects by mutableStateOf( value: savedStateHandle["subject"] ?: Subjects.entries[0])
        private set
    DonisPan
    var teacherName: String by mutableStateOf( value: savedStateHandle["teacherName"] ?: subject.teachers.first())
        private set
    DonisPan
    var type: LectureType? by mutableStateOf( value: savedStateHandle["type"] ?: LectureType.entries.first())
```

Použitie ViewModel

```
DonisPan
@Composable
fun NotificationScreen(navController: NavController, service: TimeTableNotifications) {
    var message by rememberSaveable { mutableStateOf( value: "") }
    var time by rememberSaveable { mutableStateOf( value: "") }
    var expandDays by rememberSaveable { mutableStateOf( value: false) }
    var selectedDay by rememberSaveable { mutableStateOf( value: "") }
    val daysOfWeek = listOf("Pondelok", "Utorok", "Streda", "Štvrtok", "Piatok", "Sobota", "Nedeľa")
    selectedDay = daysOfWeek[0]
```

Použitie rememberSaveable

```
    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: LazyListScope
        item { this: LazyItemScope
            Text( text: "Pridať upozornenie", fontSize = 24.sp, fontWeight = FontWeight.Bold)
            Spacer(modifier = Modifier.height(16.dp))
        }
```

Použitie LazyColumn

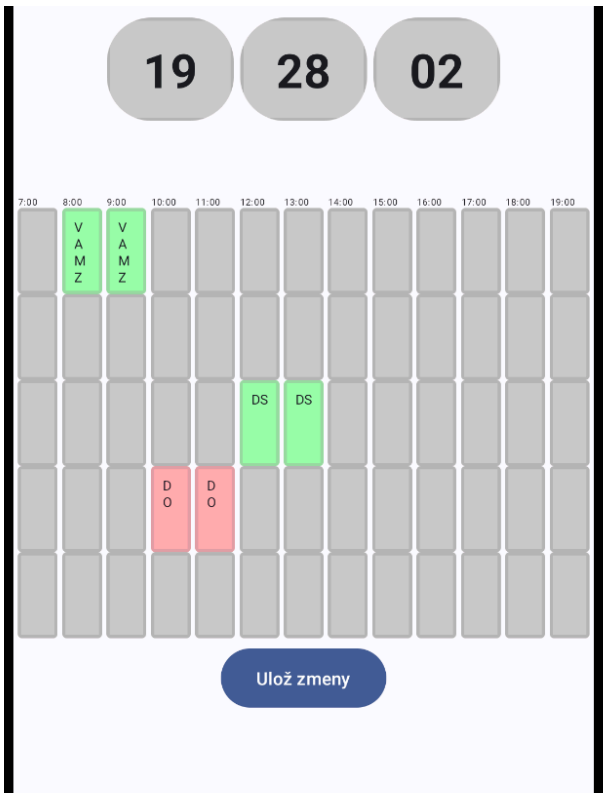
b. Obrazovky aplikácie

Hlavná obrazovka – zobrazí hodiny, rozvrh a tlačidlo na uloženie zmien. A ak si používateľ vyberie blok, tak sa mu vypíšu informácie k bloku a zobrazia tlačidlá pre ďalšie úpravy.

```
DonisPan *
@Composable
fun MainScreen(navController: NavController, clock: Clock, timeTable: TimeTable) {
    LazyColumn(
        modifier = Modifier.fillMaxSize()
    ) { this: LazyListScope
        item { this: LazyItemScope
            Box(
                modifier = Modifier
                    .fillMaxWidth()
            ) { this: BoxScope
                GetClockScreen(clock)
            }
            Spacer(modifier = Modifier.height(30.dp))
        }

        item { this: LazyItemScope
            Box(
                modifier = Modifier
                    .fillMaxWidth()
```

Príklad kódu hlavnej obrazovky



Hlavná obrazovka

Výpis o hodine a ďalšie tlačidlá

Obrazovka pridania hodiny – Zobrazí formulár s informáciami pre blok hodiny.

```
DonisPan
@Composable
fun EditorScreen(navController: NavController, index: Int, onSave: (TimeTable.ClassBlock) -> Unit) {
    val viewModel: EditorViewModel = viewModel()

    Editor(
        subject = viewModel.subject,
        onSubjectChange = { viewModel.updateSubject(it) },
        teacherName = viewModel.teacherName,
        onTeacherChange = { viewModel.updateTeacher(it) },
        type = viewModel.type,
        onTypeChange = { viewModel.updateType(it) },
        classRoom = viewModel.classRoom,
        onClassRoomChange = { viewModel.updateClassRoom(it) },
        onSave = {
            onSave(viewModel.toClassBlock())
            navController.popBackStack()
        }
    )
}
```

Obsahuje aj **DropdownMenu**, ktoré vypíše predmety a ďalšie **DropdownMenu**, ktoré vypíše relevantných učiteľov k zvolenému predmetu.

```
Text(
    text = subject.getName(),
    modifier = Modifier
        .clickable { expandSubjects = true }
        .padding(16.dp)
        .background(MaterialTheme.colorScheme.primary.copy(alpha = 0.4f))
        .fillMaxWidth()
        .padding(16.dp),
    fontSize = 16.sp
)
DropdownMenu(
    expanded = expandSubjects,
    onDismissRequest = { expandSubjects = false }
) { this: ColumnScope
    Subjects.entries.forEach { subjectItem ->
        DropdownMenuItem(
            text = { Text(subjectItem.getName()) },
            onClick = {
                onSubjectChange(subjectItem)
                onTeacherChange(subjectItem.teachers.first())
                expandSubjects = false
            }
        )
    }
}
```

DropdownMenu pre výber predmetov

Obrazovka pre pridanie novej hodiny

Obrazovka pridania notifikácie – Zobrazí formulár s informáciami pre notifikáciu. Používa **rememberSaveable** pre ukladanie údajov vo formulári.

```
DonisPan
@Composable
fun NotificationScreen(navController: NavController, service: TimeTableNotifications) {
    var message by rememberSaveable { mutableStateOf( value: "") }
    var time by rememberSaveable { mutableStateOf( value: "") }
    var expandDays by rememberSaveable { mutableStateOf( value: false) }
    var selectedDay by rememberSaveable { mutableStateOf( value: "") }
    val daysOfWeek = listOf("Pondelok", "Utorok", "Streda", "Štvrtok", "Piatok", "Sobota", "Nedeľa")
    selectedDay = daysOfWeek[0]

    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center
```

```
item { this: LazyItemScope
    Button(
        onClick = {
            service.showNotification(message, time)
            navController.popBackStack()
        },
        modifier = Modifier.fillMaxWidth()
    ) { this: RowScope
        Text( text: "Pridať upozornenie")
    }
}
```

Obrazovka pre pridanie notifikácie

c. Využitie AndroidX

ViewModel – Použitý pre správu stavu obrazoviek.

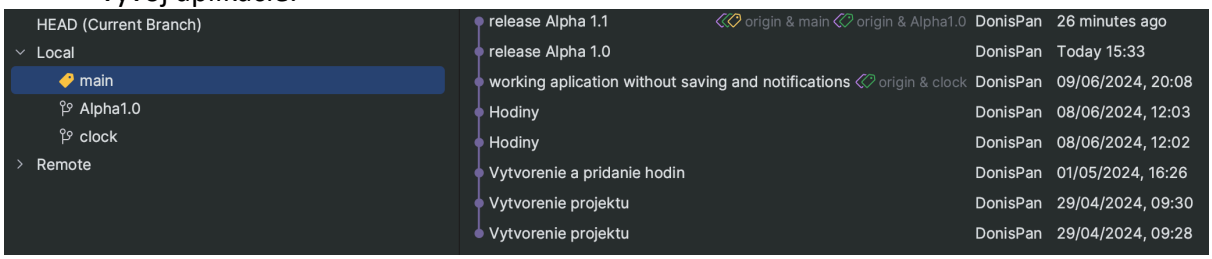
Navigation – Použitý na navigáciu medzi obrazovkami.

LaunchedEffect – Použitý pre aktualizáciu hodín každú sekundu.

RememberSaveable – Použité pre ukladanie premenných počas rotácie obrazovky.

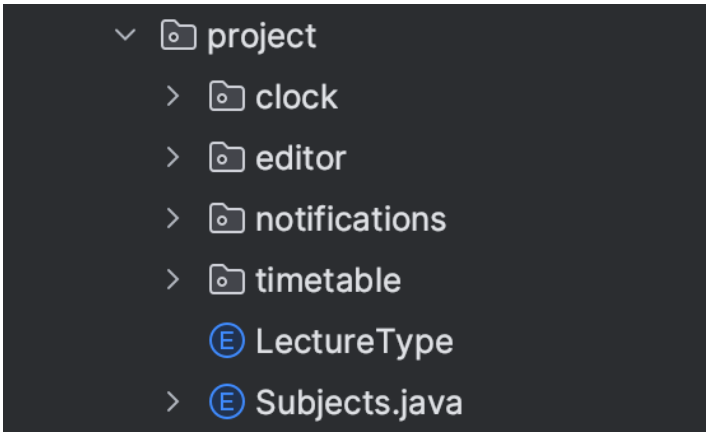
d. Práca s Gitom

Aplikácia bola vyvíjaná s použitím Gitu, s pravidelnými commitmi dokumentujúcimi vývoj aplikácie.



e. Architektúra aplikácie

Aplikácia je rozdelená do balíčkov.



Identifikátory sú pomenované logicky a intuitívne. Aplikácia obsahuje mechanizmy na ošetrenie výnimiek. Kód je optimalizovaný, bez zbytočných duplicít.

f. Dizajn aplikácie

Dizajn a ovládanie aplikácie je veľmi jednoduché a intuitívne. Obsahuje všetko potrebné a neobsahuje žiadne zbytočné rozptýlenia.

4. Zoznam použitých zdrojov

Prekonvertovanie tabuľky na json
Pomoc s riešením notifikácií + iné diskusie na StackOverflow

5. Záver

Projekt bol vytvorený s dôrazom na prehľadnosť, efektivitu a používateľskú prístupnosť.

Vývoj tejto aplikácie mi poskytol hlbšie porozumenie procesov spojených s vývojom softvéru, najmä v oblasti mobilných aplikácií. Taktiež som mal príležitosť aplikovať moderné technológie, ako sú Jetpack Compose a ViewModel.

Pracovanie s Android Studiom, Kotlinom a Jetpack Composom mi dalo zabrať. Hlavne preto, lebo som si to nechal na poslednú chvíľu. Ale za ten čas čo som na tejto aplikácií pracoval, som pre seba skonštatoval, že by som v budúcnosti určite nechcel pracovať s Kotlinom a Jetpack Composom.