БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ Кафедра программного обеспечения информационных технологий

Факультет КСИС

Специальность ПОИТ

Индивидуальная практическая работа по модулю 1

по дисциплине «Разработка программного обеспечения для мобильных платформ»

Выполнил студент: Гулькевич Д.Ю.

группа 051052

Зачетная книжка № 85100042

Минск 2024

Вариант:

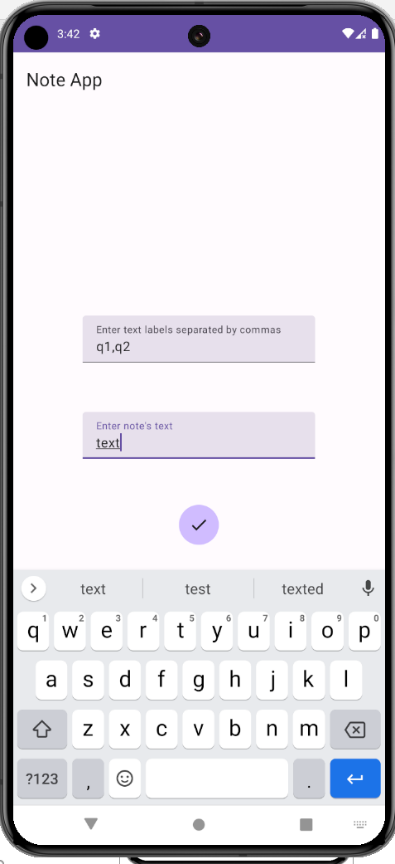
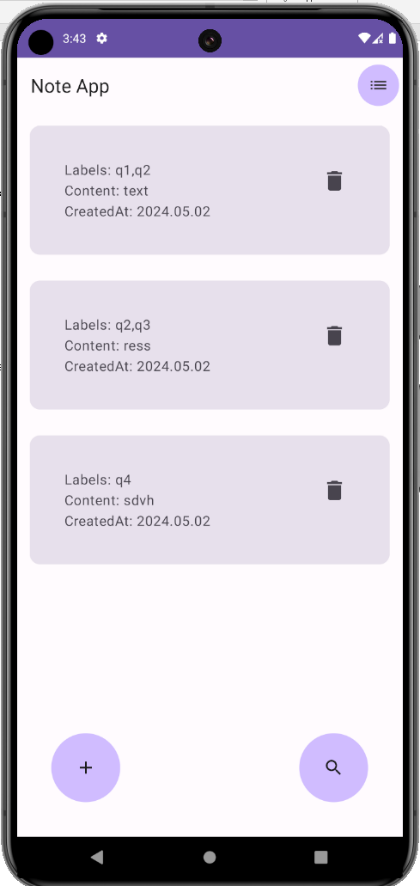
Номер варианта индивидуального задания студента определяется по правилу: 1 плюс остаток от целочисленного деления номера зачетной книжки студента (его двух последних цифр) на тридцать.

1 + 42 mod 30 = 13

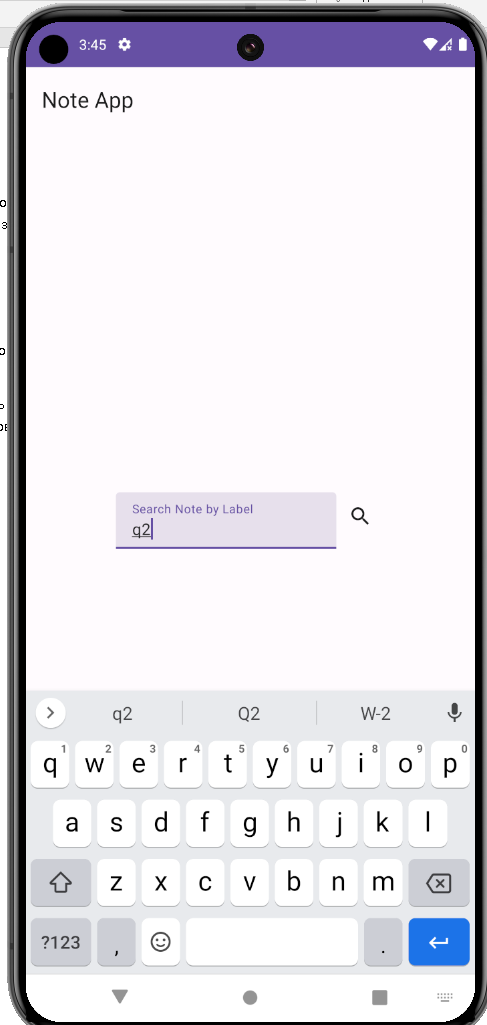
Общее задание:

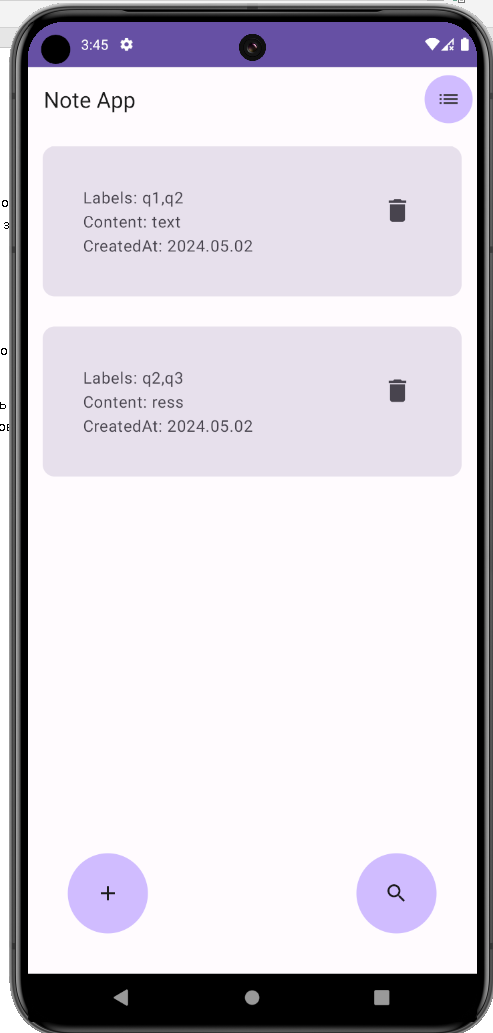
Тема: создание мобильного приложения для ведения заметок на языке Kotlin.

Вариант 13 добавить возможность создания текстовых меток и соответствующую сортировку. Также я добавил кнопку удаления и возврата ко списку со всеми записями после сортировки по метке для удобства



Стартовый экран и добавление новых заметок





Поиск по меткам

**Исходный код:**

**Модель статьи:**

Весь экран:

@OptIn(ExperimentalMaterial3Api::class)  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 val viewModel = ViewModelProvider(this).get(NoteViewModel::class.*java*)  
  
 *window*.*statusBarColor* = ContextCompat.getColor(this, R.color.*purple\_200*)  
 *setContent* **{** RoomDatabaseModuleTheme **{** ListOfNotes(viewModel)  
 **}  
 }** }  
}  
  
  
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")  
@OptIn(ExperimentalMaterial3Api::class, ExperimentalComposeUiApi::class)  
@Composable  
fun ListOfNotes(viewModel: NoteViewModel) {  
 var searchVisible by remember **{** *mutableStateOf*(false) **}** var createVisible by remember **{** *mutableStateOf*(false) **}** var listNotesVisible by remember **{** *mutableStateOf*(true) **}** val focusManager = *LocalFocusManager*.current  
 viewModel.getAllNotesSortedByDate()  
 Scaffold(topBar = **{** TopAppBar(  
 title = **{** Text("Note App", fontSize = 22.*sp*) **}**,  
 modifier = Modifier.*background*(*Purple80*),  
 actions = **{** if(!(searchVisible or createVisible)){  
 IconButton(  
 modifier = Modifier  
 .*background*(*Purple80*, *CircleShape*),  
 onClick = **{** listNotesVisible = true  
 viewModel.getAllNotesSortedByDate()  
 **}** ) **{** Icon(Icons.Default.*List*, contentDescription = "All Notes") **}** }  
 **}** )  
 **}**,  
 bottomBar = **{** Row(  
 horizontalArrangement = Arrangement.SpaceBetween,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(40.*dp*)  
 ) **{** //ADD  
 IconButton(  
 modifier = Modifier  
 .*size*(80.*dp*)  
 .*background*(*Purple80*, *CircleShape*),  
 onClick = **{** searchVisible = false  
 listNotesVisible = !listNotesVisible  
 createVisible = !createVisible  
 **}** ) **{** Icon(Icons.Filled.*Add*, contentDescription = "") **}** //SEARCH  
 IconButton(  
 modifier = Modifier  
 .*size*(80.*dp*)  
 .*background*(*Purple80*, *CircleShape*),  
 onClick = **{** searchVisible = !searchVisible  
 listNotesVisible = !listNotesVisible  
 createVisible = false  
 **}** ) **{** Icon(Icons.Filled.*Search*, contentDescription = "") **}  
 }  
 }**,  
  
  
 ) **{** val state = viewModel.state  
  
 AnimatedVisibility(  
 visible = listNotesVisible,  
 enter = *scaleIn*(animationSpec = *tween*(durationMillis = 650)),  
 exit = *scaleOut*(animationSpec = *tween*(durationMillis = 650))  
 )**{** LazyColumn(  
 horizontalAlignment = Alignment.CenterHorizontally,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(top = **it**.calculateTopPadding())  
 ) **{** *items*(state.value) **{** Card(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(15.*dp*)  
 .*clickable* **{ }**,  
 ) **{** Row(horizontalArrangement = Arrangement.SpaceBetween,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(40.*dp*))**{** Text("Labels: ${**it**.labels}\nContent: ${**it**.content}\nCreatedAt: ${SimpleDateFormat("yyyy.MM.dd").format(Date(**it**.date))}")  
 IconButton(  
 onClick = **{** viewModel.deleteNote(**it**)  
  
 **}** ) **{** Icon(  
 imageVector = Icons.Default.*Delete*,  
 contentDescription = "",  
 modifier = Modifier.*size*(30.*dp*)  
 )  
 **}  
 }  
 }  
  
  
 }  
 }  
 }** AnimatedVisibility(  
 visible = createVisible,  
 enter = *fadeIn*(initialAlpha = 0.4f),  
 exit = *fadeOut*(animationSpec = *tween*(durationMillis = 250))  
 ) **{** Box(  
 contentAlignment = Alignment.Center,  
 modifier = Modifier.*fillMaxSize*()  
 ) **{** val focusRequester = remember **{** FocusRequester() **}** var labels by remember **{** *mutableStateOf*("") **}** var noteContent by remember **{** *mutableStateOf*("") **}** var date by remember **{** *mutableLongStateOf*(0) **}** Column(  
 modifier = Modifier.*fillMaxHeight*(0.3f),  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.SpaceBetween  
 ) **{** TextField(value = labels,  
 onValueChange = **{** labels = **it  
 }**,  
 label = **{** Text("Enter text labels separated by commas") **}**,  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(imeAction = ImeAction.Done),  
 keyboardActions = KeyboardActions(  
 onDone = **{** focusRequester.requestFocus() **}** ),  
 modifier = Modifier.*onKeyEvent* **{** if (**it**.nativeKeyEvent.*keyCode* == KeyEvent.*KEYCODE\_ENTER*) {  
 focusRequester.requestFocus()  
 true  
 }  
 false  
 **}** )  
  
 TextField(  
 value = noteContent,  
 label = **{** Text("Enter note's text") **}**,  
 onValueChange = **{** noteContent = **it  
 }**,  
 modifier = Modifier.*focusRequester*(focusRequester),  
 )  
 IconButton(  
 modifier = Modifier  
 .*size*(40.*dp*)  
 .*background*(*Purple80*, *CircleShape*),  
 onClick = **{** focusManager.clearFocus()  
 listNotesVisible = !listNotesVisible  
 createVisible = !createVisible  
 date = java.util.Date().*time* val newNote = Note(  
 labels = labels,  
 content = noteContent,  
 date = date  
 )  
 viewModel.insertNote(newNote)  
 **}** ) **{** Icon(Icons.Filled.*Done*, contentDescription = "") **}  
  
 }  
 }  
  
 }** AnimatedVisibility(  
 visible = searchVisible,  
 enter = *fadeIn*(  
 initialAlpha = 0.4f  
 ),  
 exit = *fadeOut*(  
 animationSpec = *tween*(durationMillis = 250)  
 )  
 ) **{** var searchText by remember **{** *mutableStateOf*("") **}** Box(  
 contentAlignment = Alignment.Center,  
 modifier = Modifier.*fillMaxSize*()  
 ) **{** Row(horizontalArrangement = Arrangement.Center) **{** TextField(  
 value = searchText,  
 label = **{** Text("Search Note by Label") **}**,  
 modifier = Modifier.*width*(220.*dp*),  
 onValueChange = **{** searchText = **it  
 }**)  
 IconButton(onClick = **{** listNotesVisible = !listNotesVisible  
 searchVisible = !searchVisible  
 viewModel.searchNoteByLabel(searchText)  
 **}**) **{** Icon(Icons.Filled.*Search*, contentDescription = "") **}  
 }  
  
 }  
  
 }  
 }**}

Dao заметки в БД:

@Dao  
interface NoteDao {  
 @Query("SELECT \* FROM notes ORDER BY date")  
 fun getAllNotesSortedByDate(): List<Note>  
  
 @Query("SELECT \* FROM notes ORDER BY labels")  
 fun getAllNotesSortedByLabels(): List<Note>  
  
 @Query("SELECT \* FROM notes WHERE instr(labels, :searchingLabel) > 0")  
 fun getAllNotesFoundedByLabel(searchingLabel: String): List<Note>  
  
 @Upsert  
 suspend fun insetNote(note: Note)  
  
 @Delete  
 suspend fun delete(note: Note)  
  
  
}

Модель заметки в БД:

@Entity(tableName = "notes")  
data class Note(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val labels: String?,  
 val content: String?,  
 val date: Long,  
)

Note View Model:

class NoteViewModel(): ViewModel() {  
 private val database = NoteDatabase.getInstance(app.context!!)  
 private val noteDao = database.noteDao()  
  
 private val \_state: MutableState<List<Note>> = *mutableStateOf*(*emptyList*())  
 val state: State<List<Note>> = \_state  
  
 private fun getAll(){  
  
 *viewModelScope*.*launch*(Dispatchers.IO) **{** \_state.value = noteDao.getAllNotesSortedByDate()  
 **}** }  
  
 fun getAllNotesSortedByDate() = *viewModelScope*.*launch*(Dispatchers.IO) **{** \_state.value = noteDao.getAllNotesSortedByDate()  
 **}** fun searchNoteByLabel(label: String) = *viewModelScope*.*launch*(Dispatchers.IO) **{** \_state.value = noteDao.getAllNotesFoundedByLabel(label)  
 **}** fun insertNote(note: Note) = *viewModelScope*.*launch*(Dispatchers.IO) **{** noteDao.insetNote(note)  
 getAll()  
 **}** fun deleteNote(note: Note) = *viewModelScope*.*launch*(Dispatchers.IO) **{** noteDao.delete(note)  
 getAll()  
 **}**}

}

Бд:

@Database(entities = *arrayOf*(Note::class), version = 1)  
abstract class NoteDatabase: RoomDatabase() {  
 companion object {  
 @Volatile  
 private var INSTANCE: NoteDatabase? = null  
  
 fun getInstance(context: Context): NoteDatabase {  
 return INSTANCE ?: *synchronized*(this) **{** val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 NoteDatabase::class.*java*,  
 "NotesDatabase"  
 ).build()  
 INSTANCE = instance  
 instance  
 **}** }  
 }  
 abstract fun noteDao(): NoteDao  
}

**Сторонние библиотеки**

*implementation*("androidx.room:room-common:2.6.1")  
*implementation*("androidx.room:room-ktx:2.6.1")  
*kapt*("androidx.room:room-compiler:2.6.1")  
*implementation*("androidx.core:core-ktx:1.10.1")  
*implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.6.1")  
*implementation*("androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1")  
*implementation*("androidx.activity:activity-compose:1.7.0")  
*implementation*(platform("androidx.compose:compose-bom:2023.08.00"))  
*implementation*("androidx.compose.ui:ui")  
*implementation*("androidx.compose.ui:ui-graphics")  
*implementation*("androidx.compose.ui:ui-tooling-preview")  
*implementation*("androidx.appcompat:appcompat:1.6.1")  
*implementation*("com.google.android.material:material:1.11.0")  
*implementation*("androidx.activity:activity:1.8.0")  
*implementation*("androidx.constraintlayout:constraintlayout:2.1.4")  
*implementation*("androidx.compose.material3:material3")

**Разрешения**