БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ Кафедра программного обеспечения информационных технологий

Факультет КСИС

Специальность ПОИТ

Индивидуальная практическая работа по модулю 2

по дисциплине «Разработка программного обеспечения для мобильных платформ»

Выполнил студент: Гулькевич Д.Ю.

группа 051052

Зачетная книжка № 85100042

Минск 2024

Вариант:

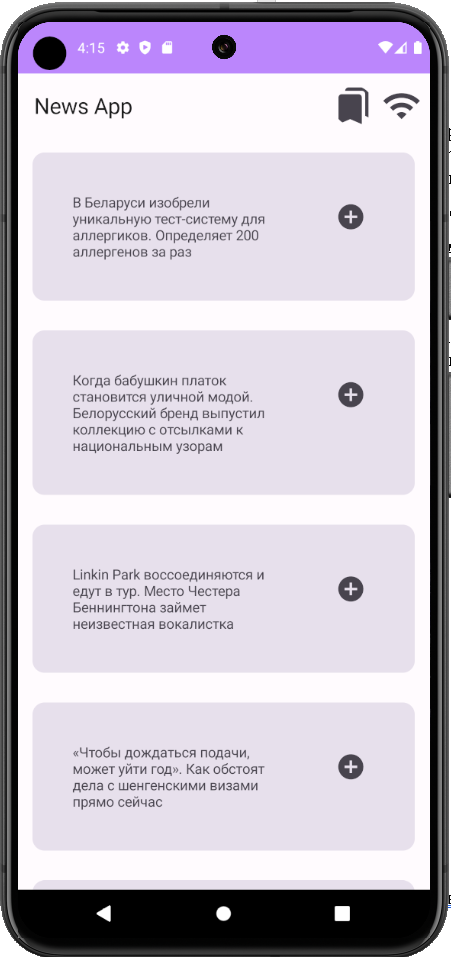
Номер варианта индивидуального задания студента определяется по правилу: 1 плюс остаток от целочисленного деления номера зачетной книжки студента (его двух последних цифр) на тридцать.

1 + 42 mod 30 = 13

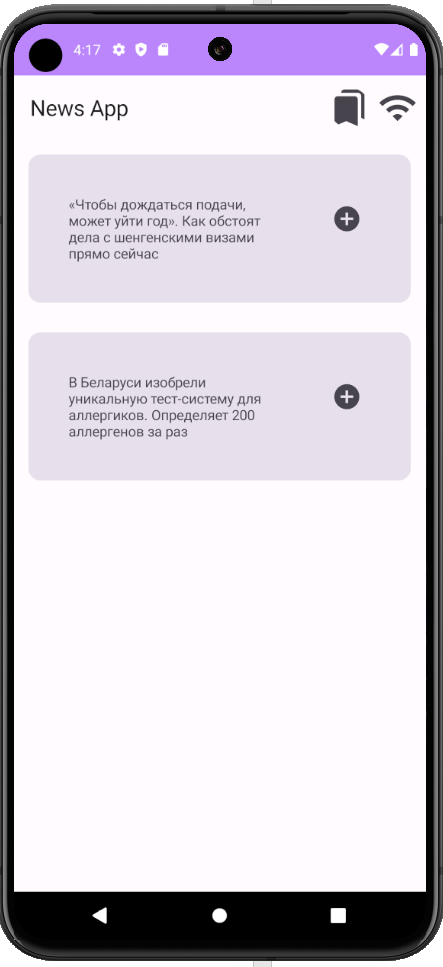
Общее задание:

Тема: создание мобильного приложения для прочтения RSS.

Вариант 13 добавить возможность создания списка статей для прочтения позже. Также я добавил кнопку перехода к списку статей, для прочтения позже, и кнопку возврата ко всем статьям. При нажатии на статью открывается полный текст статьи.



Стартовый экран и добавление новости в лист прочтения позже(клик по кнопке +)



Переход к списку прочесть позже (кнопка с закладками). Кнопка Wifi для возврата ко всем статьям

**Исходный код:**

@OptIn(ExperimentalMaterial3Api::class)  
class MainActivity : ComponentActivity() {  
 private lateinit var newsDao: NewDao  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 val viewModel = NewsViewModel()  
 *window*.*statusBarColor* = ContextCompat.getColor(this, R.color.*purple\_200*)  
 *setContent* **{** ListOfNews(viewModel)  
 **}** }  
}  
  
  
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")  
@OptIn(ExperimentalMaterial3Api::class, ExperimentalComposeUiApi::class)  
@Composable  
fun ListOfNews(viewModel: NewsViewModel) {  
  
 Scaffold(topBar = **{** TopAppBar(  
 title = **{** Text("News App", fontSize = 22.*sp*) **}**,  
 modifier = Modifier.*background*(*Purple80*),  
 actions = **{** IconButton(  
 onClick = **{** viewModel.getAllReadLaterNews()  
 **}**,  
  
 ) **{** Icon(  
 imageVector = Icons.Filled.*Bookmarks*,  
 contentDescription = "Read later",  
 modifier = Modifier.*size*(60.*dp*)  
 )  
 **}** IconButton(  
 onClick = **{** viewModel.getAllNews()  
 **}**,  
  
 ) **{** Icon(  
 imageVector = Icons.Filled.*Wifi*,  
 contentDescription = "Read later",  
 modifier = Modifier.*size*(60.*dp*)  
 )  
 **}  
 }** )  
 **}** ) **{** val state = viewModel.state  
 LazyColumn(  
 horizontalAlignment = Alignment.CenterHorizontally,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(top = **it**.calculateTopPadding())  
 ) **{** *items*(state.value) **{** ExpandableCard(**it**!!, viewModel)  
 **}  
 }  
  
 }**}  
  
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")  
@OptIn(ExperimentalMaterial3Api::class)  
@Preview(showSystemUi = true)  
@Composable  
fun check(){  
 Scaffold(topBar = **{** TopAppBar(  
 title = **{** Text("News App", fontSize = 22.*sp*) **}**,  
 modifier = Modifier.*background*(*Purple80*),  
 actions = **{** IconButton(  
 onClick = **{  
  
 }** ) **{** Icon(  
 imageVector = Icons.Filled.*AddCircle*,  
 contentDescription = "Read later",  
 modifier = Modifier.*size*(60.*dp*)  
 )  
 **}** IconButton(  
 onClick = **{  
 }**,  
  
 ) **{** Icon(  
 imageVector = Icons.Filled.*Wifi*,  
 contentDescription = "Read later",  
 modifier = Modifier.*size*(60.*dp*)  
 )  
 **}  
 }** )  
 **}** ) **{** Text(text="qwe")  
  
 **}**}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun ExpandableCard(rssObj: New, viewModel: NewsViewModel) {  
  
 var expanded by remember **{** *mutableStateOf*(false) **}** Card(  
  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(15.*dp*),  
 onClick = **{** expanded = !expanded **}** ) **{** Row(  
 modifier = Modifier.*fillMaxWidth*().*padding*(40.*dp*),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Column **{** Text(text = rssObj?.title!!, modifier = Modifier.*width*(200.*dp*))  
 if (expanded) {  
 Text(  
 text = Html.fromHtml(rssObj.description, 1).toString(),  
 modifier = Modifier  
 .*padding*(8.*dp*)  
 )  
 Text(  
 text = rssObj.pubDate!!.toString(),  
 modifier = Modifier.*padding*(2.*dp*)  
 )  
 }  
 **}** IconButton(  
 onClick = **{** viewModel.insert(New(title = rssObj?.title!!, description = rssObj.description, pubDate = rssObj.pubDate!!.toString()))  
 **}** ) **{** Icon(  
 imageVector = Icons.Filled.*AddCircle*,  
 contentDescription = "Read later",  
 modifier = Modifier.*size*(30.*dp*)  
 )  
 **}  
 }  
  
 }**}

Dao новости в БД:

@Dao  
interface NewDao {  
 @Query("SELECT \* FROM news ORDER BY pubDate")  
 fun getAllNotesSortedByDate(): List<New>  
  
 @Upsert  
 suspend fun inset(anew: New)  
  
 @Delete  
 suspend fun delete(aNew: New)  
  
  
}

Модель новости в БД:

@Entity(tableName = "news")  
data class New(  
 @PrimaryKey() val title: String,  
 val description: String?,  
 val pubDate: String,  
)

News View Model:

class NewsViewModel() : ViewModel() {  
 private val database = NewsDatabase.getInstance(app.context!!)  
 private val newsDao = database.newsDao()  
 private val \_state: MutableState<List<New>> = *mutableStateOf*(*emptyList*())  
 private val \_allNews: MutableState<List<New>> = *mutableStateOf*(*emptyList*())  
 init {  
 val parser = RssStandardParser()  
 fun mapNews(  
 src: RssStandardItem,  
 ) = New(title = src.title!!, description = src.description, pubDate = src.pubDate!!)  
 *viewModelScope*.*launch*(Dispatchers.IO) **{** val httpClient = *HttpClient*()  
 val xml = httpClient.get("https://people.onliner.by/feed").body<String>()  
 val parsedXmlList = parser.parse(xml).items?.*toList*()  
 if (parsedXmlList != null) {  
 \_state.value = parsedXmlList.*map* **{** New(title = **it**.title!!, description = **it**.description, pubDate = **it**.pubDate!!)**}** }  
 httpClient.close()  
 **}**// if (Network.checkConnectivity(app.context!!)){  
// viewModelScope.launch(Dispatchers.IO) {  
//  
// }  
// }  
// else  
// {  
// getAll()  
// }  
 }  
  
  
 val state: State<List<New>> = \_state  
  
 fun getAllReadLaterNews(){  
  
 *viewModelScope*.*launch*(Dispatchers.IO) **{** \_allNews.value = \_state.value  
 \_state.value = newsDao.getAllNotesSortedByDate()  
 **}** }  
  
 fun getAllNews(){  
 \_state.value = \_allNews.value  
 }  
  
 fun getNewsSortedByDate() = *viewModelScope*.*launch*(Dispatchers.IO) **{** \_state.value = newsDao.getAllNotesSortedByDate()  
 **}** fun insert(new: New) = *viewModelScope*.*launch*(Dispatchers.IO) **{** newsDao.inset(new)  
 **}** fun deleteNote(aNew: New) = *viewModelScope*.*launch*(Dispatchers.IO) **{** newsDao.delete(aNew)  
 **}**}

Бд:

@Database(entities = *arrayOf*(New::class), version = 2)  
abstract class NewsDatabase: RoomDatabase() {  
  
 companion object {  
 @Volatile  
 private var INSTANCE: NewsDatabase? = null  
  
 fun getInstance(context: Context): NewsDatabase {  
 return INSTANCE ?: *synchronized*(this) **{** val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 NewsDatabase::class.*java*,  
 "NewsDatabase"  
 ).build()  
 INSTANCE = instance  
 instance  
 **}** }  
 }  
  
 abstract fun newsDao(): NewDao  
}

**Сторонние библиотеки**

*implementation*("com.github.ivanisidrowu.KtRssReader:kotlin:v2.2.1")  
*implementation*("io.ktor:ktor-client-cio:2.3.6")  
*implementation*("io.ktor:ktor-client-android:2.3.6")  
  
*implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.1")  
*implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.1")  
*implementation*("io.coil-kt:coil-compose:2.2.2")  
*implementation*("androidx.lifecycle:lifecycle-runtime-compose:2.6.2")  
  
*implementation*("androidx.compose.material:material-icons-extended:1.6.6")  
  
*implementation*("androidx.compose.material3:material3")

**Разрешения**

<uses-permission android:name="android.permission.INTERNET" />