|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №10**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  студент группыИКБО-28-22 | Некрасов Г.А. |
| **Проверил:**  к.э.н доцент | Степанов П.В. |

Москва 2024 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКОЕ ЗАДАНИЕ 3](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915647)

[Код практического задания 4](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915648)

[Тестирование практического задания 6](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915649)

[Вывод 6](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915650)

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Реализовать приложение из практической работы №7 средствами Jetpack Compose, сохранив изначальное расположение элементов разметки, применив:

1. Изученные контейнеры компоновки (Column, Row, Box, Card).

2. Динамический список (LazyColumn или LazyRow на выбор).

# Код практического задания

В ходе выполнения работы, мы реализовали приложение из практической работы №7, использовав Jetpack Compose (см. Листинги 1-2)

Листинг 1. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya1011\_test  import android.annotation.SuppressLint import android.os.Bundle import android.widget.Toast import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.Arrangement import androidx.compose.foundation.layout.Box import androidx.compose.foundation.layout.Column import androidx.compose.foundation.layout.Spacer import androidx.compose.foundation.layout.fillMaxSize import androidx.compose.foundation.layout.fillMaxWidth import androidx.compose.foundation.layout.height import androidx.compose.foundation.layout.padding import androidx.compose.foundation.lazy.LazyColumn import androidx.compose.foundation.text.BasicTextField import androidx.compose.material3.Button import androidx.compose.material3.ExperimentalMaterial3Api import androidx.compose.material3.MaterialTheme import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.material3.TopAppBar import androidx.compose.runtime.Composable import androidx.compose.runtime.getValue import androidx.compose.runtime.mutableStateOf import androidx.compose.runtime.remember import androidx.compose.runtime.rememberCoroutineScope import androidx.compose.runtime.setValue import androidx.compose.ui.Alignment import androidx.compose.ui.Modifier import androidx.compose.ui.graphics.asImageBitmap import androidx.compose.ui.platform.LocalContext import androidx.compose.ui.text.input.TextFieldValue import androidx.compose.ui.unit.dp import kotlinx.coroutines.launch import androidx.compose.ui.res.painterResource import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.\* import androidx.compose.foundation.lazy.LazyColumn import androidx.compose.material3.Button import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.material3.TopAppBar import androidx.compose.runtime.\* import androidx.compose.ui.graphics.asImageBitmap import androidx.compose.ui.platform.LocalContext import androidx.compose.ui.unit.dp import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch import kotlinx.coroutines.withContext import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.\* import androidx.compose.foundation.lazy.LazyColumn import androidx.compose.material.icons.Icons import androidx.compose.material.icons.filled.Home import androidx.compose.material.icons.filled.List import androidx.compose.material.icons.filled.Menu import androidx.compose.material3.\* import androidx.compose.runtime.\* import androidx.compose.ui.graphics.asImageBitmap import androidx.compose.ui.platform.LocalContext import androidx.compose.ui.unit.dp import androidx.work.OneTimeWorkRequestBuilder import androidx.work.WorkManager import androidx.work.Worker import androidx.work.WorkerParameters import kotlinx.coroutines.launch import java.util.concurrent.TimeUnit  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContent {  MyApp()  }  } }  @SuppressLint("UnusedMaterial3ScaffoldPaddingParameter") @OptIn(ExperimentalMaterial3Api::class) @Composable fun MyApp() {  val context = LocalContext.current  val imageRepository = remember { ImageRepository(context) }  var imageUrl by remember { mutableStateOf(TextFieldValue("")) }  val coroutineScope = rememberCoroutineScope()   // Список изображений  var imageList by remember { mutableStateOf<List<android.graphics.Bitmap>>(emptyList()) }   // Состояние текущего выбранного экрана  var selectedScreen by remember { mutableStateOf("Home") }   // Состояние для открытия/закрытия Drawer  val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)  val coroutineScopeDrawer = rememberCoroutineScope()   // Основной Scaffold  ModalNavigationDrawer(  drawerState = drawerState,  drawerContent = {  ModalDrawerSheet {  Text(  text = "Navigation",  modifier = Modifier.padding(16.dp),  style = MaterialTheme.typography.titleMedium  )  Spacer(modifier = Modifier.height(8.dp))  NavigationDrawerItem(  label = { Text("Home") },  selected = selectedScreen == "Home",  onClick = {  selectedScreen = "Home"  coroutineScopeDrawer.launch { drawerState.close() }  }  )  NavigationDrawerItem(  label = { Text("Gallery") },  selected = selectedScreen == "Gallery",  onClick = {  selectedScreen = "Gallery"  coroutineScopeDrawer.launch { drawerState.close() }  }  )  }  }  ) {  Scaffold(  topBar = {  TopAppBar(  title = { Text("Dynamic Image List") },  navigationIcon = {  IconButton(onClick = { coroutineScopeDrawer.launch { drawerState.open() } }) {  Icon(Icons.Default.Menu, contentDescription = "Menu")  }  }  )  },  bottomBar = {  BottomAppBar {  NavigationBar {  NavigationBarItem(  icon = { Icon(Icons.Default.Home, contentDescription = "Home") },  label = { Text("Home") },  selected = selectedScreen == "Home",  onClick = { selectedScreen = "Home" }  )  NavigationBarItem(  icon = { Icon(Icons.Default.List, contentDescription = "Home") },  label = { Text("Gallery") },  selected = selectedScreen == "Gallery",  onClick = { selectedScreen = "Gallery" }  )  }  }  },  content = {  when (selectedScreen) {  "Home" -> HomeScreen(  imageUrl = imageUrl,  onImageUrlChange = { imageUrl = it },  imageList = imageList,  onAddImage = { bitmap ->  imageList = imageList + bitmap  imageUrl = TextFieldValue("") // Очищаем поле ввода  },  imageRepository = imageRepository  )  "Gallery" -> GalleryScreen(imageList = imageList)  }  }  )  }   // Добавление WorkManager  WorkManager.getInstance(context).enqueue(  OneTimeWorkRequestBuilder<ImageTaskWorker>()  .setInitialDelay(15, TimeUnit.MINUTES) // Выполнение задачи через 15 минут после запуска  .build()  ) }  @Composable fun HomeScreen(  imageUrl: TextFieldValue,  onImageUrlChange: (TextFieldValue) -> Unit,  imageList: List<android.graphics.Bitmap>,  onAddImage: (android.graphics.Bitmap) -> Unit,  imageRepository: ImageRepository ) {  val context = LocalContext.current  val coroutineScope = rememberCoroutineScope()   Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp),  verticalArrangement = Arrangement.spacedBy(16.dp)  ) {  BasicTextField(  value = imageUrl,  onValueChange = { onImageUrlChange(it) },  modifier = Modifier  .fillMaxWidth()  .padding(8.dp)  .height(50.dp)  .border(1.dp, MaterialTheme.colorScheme.primary)  .padding(8.dp),  decorationBox = { innerTextField ->  Box(Modifier.fillMaxSize()) {  if (imageUrl.text.isEmpty()) {  Text(  "Enter image URL",  style = MaterialTheme.typography.bodySmall,  color = MaterialTheme.colorScheme.onSurfaceVariant  )  }  innerTextField()  }  }  )   Button(  onClick = {  if (imageUrl.text.isNotEmpty()) {  coroutineScope.launch {  val loadedImage = imageRepository.downloadImage(imageUrl.text)  if (loadedImage != null) {  onAddImage(loadedImage)  Toast.makeText(context, "Image added to list", Toast.LENGTH\_SHORT).show()  } else {  Toast.makeText(context, "Error loading image", Toast.LENGTH\_SHORT).show()  }  }  }  },  modifier = Modifier  .fillMaxWidth()  .height(50.dp)  ) {  Text(text = "Add Image to List")  }  } }  @Composable fun GalleryScreen(imageList: List<android.graphics.Bitmap>) {  LazyColumn(  modifier = Modifier.fillMaxSize(),  verticalArrangement = Arrangement.spacedBy(16.dp)  ) {  items(imageList.size) { index ->  val bitmap = imageList[index]  Box(  modifier = Modifier  .fillMaxWidth()  .height(300.dp),  contentAlignment = Alignment.Center  ) {  Image(  bitmap = bitmap.asImageBitmap(),  contentDescription = null,  modifier = Modifier.fillMaxSize()  )  }  }  } }  class ImageTaskWorker(appContext: android.content.Context, workerParams: WorkerParameters) : Worker(appContext, workerParams) {  override fun doWork(): Result {  // Задача, которую будет выполнять WorkManager  return Result.success()  } }  //@Preview(showBackground = true) @Composable fun DefaultPreview() {  MyApp() } |

Листинг 2. ImageRepository.kt

|  |
| --- |
| package com.example.practi4eskaya1011\_test  import android.content.Context import android.graphics.Bitmap import android.graphics.BitmapFactory import android.util.Log import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.withContext import java.io.File import java.io.FileOutputStream import java.io.InputStream import java.net.HttpURLConnection import java.net.URL  class ImageRepository(private val context: Context) {   suspend fun downloadImage(imageUrl: String): Bitmap? {  return withContext(Dispatchers.IO) {  try {  val url = URL(imageUrl)  val connection: HttpURLConnection = url.openConnection() as HttpURLConnection  connection.doInput = true  connection.connect()  val inputStream: InputStream = connection.inputStream  BitmapFactory.decodeStream(inputStream)  } catch (e: Exception) {  Log.e("ImageRepository", "Error downloading image", e)  null  }  }  }   suspend fun saveImageToInternalStorage(bitmap: Bitmap, filename: String): Boolean {  return withContext(Dispatchers.IO) {  try {  val file = File(context.filesDir, "$filename.jpg")  val fos = FileOutputStream(file)  bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos)  fos.flush()  fos.close()  true  } catch (e: Exception) {  Log.e("ImageRepository", "Error saving image", e)  false  }  }  } } |

# Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 1-3)



Рисунок 1. Запуск приложения

# Вывод

В процессе выполнения работы мы научились оформлять наше приложение