МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Программирование мобильных устройств»

Лабораторная работа №4

Архитектура приложения (MVVM).

Выполнил:

студент гр. ПИбд-31

Распаев Н.И.

Проверил:

доценткафедры ИС

Филиппов А.А.

Ульяновск, 2023 г

**Необходимо:**

1. Создать интерфейсы-репозитории для абстрагирования от реализации механизма хранения данных и их реализацию для работы с DAO, которые были разработаны в ЛР №3.
2. Создать ViewModel для реализации бизнес-логики.
3. Вынести всю бизнес-логику из представлений (UI) во ViewModel-классы.
4. Организовать взаимодействие между UI и слоем бизнес-логики.
5. Все списки элементов в UI должны поддерживать пагинацию.
6. Отчет и изменения проекта загрузить в репозиторий по адресу http://student.git.athene.techВ ходе выполнения были реализованы классы ViewModel для каждой страницы приложения. Для инициализации данных классов испоьзуется viewModelFactory. Например:

package com.nodj.hardwareStore.ui

import androidx.lifecycle.ViewModelProvider

import androidx.lifecycle.createSavedStateHandle

import androidx.lifecycle.viewmodel.CreationExtras

import androidx.lifecycle.viewmodel.initializer

import androidx.lifecycle.viewmodel.viewModelFactory

import com.nodj.hardwareStore.db.database.ShopApplication

import com.nodj.hardwareStore.db.service.OrderService

import com.nodj.hardwareStore.ui.category.CategorizedProductsViewModel

import com.nodj.hardwareStore.ui.category.CategoryListViewModel

import com.nodj.hardwareStore.ui.category.edit.CategoryEditViewModel

import com.nodj.hardwareStore.ui.page.cart.ProductListInCartViewModel

import com.nodj.hardwareStore.ui.page.orders.OrdersViewModel

import com.nodj.hardwareStore.ui.page.orders.order.OrderViewModel

import com.nodj.hardwareStore.ui.product.edit.CategoryDropDownViewModel

import com.nodj.hardwareStore.ui.product.edit.ProductEditViewModel

import com.nodj.hardwareStore.ui.product.list.ProductListViewModel

object AppViewModelProvider {

val Factory = viewModelFactory {

initializer {

ProductListViewModel(

shopApplication().container.productRepository,

shopApplication().container.userWithProductsRepository

)

}

initializer {

ProductListInCartViewModel(

shopApplication().container.productRepository,

shopApplication().container.userWithProductsRepository,

OrderService(

shopApplication().container.orderRepository,

shopApplication().container.productRepository,

shopApplication().container.orderWithProductsRepository,

shopApplication().container.userWithProductsRepository,

))

}

initializer {

ProductEditViewModel(

this.createSavedStateHandle(),

shopApplication().container.productRepository

)

}

initializer {

CategoryDropDownViewModel(shopApplication().container.categoryRepository)

}

initializer {

OrdersViewModel(

shopApplication().container.orderRepository

)

}

initializer {

OrderViewModel(

this.createSavedStateHandle(),

shopApplication().container.productRepository,

shopApplication().container.orderRepository,

shopApplication().container.userWithProductsRepository,

)

}

initializer {

CategoryListViewModel(

shopApplication().container.categoryRepository

)

}

initializer {

CategorizedProductsViewModel(

shopApplication().container.productRepository,

shopApplication().container.userWithProductsRepository,

this.createSavedStateHandle()

)

}

initializer {

CategoryEditViewModel(

this.createSavedStateHandle(),

shopApplication().container.categoryRepository

)

}

}

}

fun CreationExtras.shopApplication(): ShopApplication =

(this[ViewModelProvider.AndroidViewModelFactory.APPLICATION\_KEY] as ShopApplication)

Для создания и редактирования материалов нужно было реализовать загрузку изображений с телефона, для этого была реализована функция для загрузки:

fun convertFileInputStreamToByteArray(fileInputStream: FileInputStream): ByteArray? {

try {

val buffer = ByteArray(1024)

val byteArrayOutputStream = ByteArrayOutputStream()

var bytesRead: Int

while (fileInputStream.read(buffer).also { bytesRead = it } != -1) {

byteArrayOutputStream.write(buffer, 0, bytesRead)

}

return byteArrayOutputStream.toByteArray()

} catch (e: IOException) {

e.printStackTrace()

} finally {

try {

fileInputStream.close()

} catch (e: IOException) {

e.printStackTrace()

}

}

return null

}

@Composable

fun ImageSelectionScreen(

productUiState: ProductUiState,

onUpdate: (ProductDetails) -> Unit,

) {

val context = LocalContext.current

val launcher = rememberLauncherForActivityResult(

contract = ActivityResultContracts.GetContent(),

onResult = { uri: Uri? ->

uri?.let { saveImageToInternalStorage(context, it) }

onUpdate(productUiState.productDetails.copy(image = convertFileInputStreamToByteArray(context.openFileInput("image.jpg"))!!))

}

)

Button(onClick = {

launcher.launch("image/\*")

}) {

Text(text = "Select Image")

}

}

Код можно посмотреть по ссылкам:

StudentGit: <http://student.git.athene.tech/Nikolay-Raspaev/PMU/src/branch/master>

GitHub: <https://github.com/keyssen/mobile_app/tree/master>