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

Выполнил:  
студент гр. ПИбд-32  
Алейкин А.М.  
Проверил:  
доцент кафедры  
Филиппов А.А.

Ульяновск, 2023 г

**Задание**

Архитектура приложения (MVVM).

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.

**Выполнение**

1. Были созданы интерфейсы-репозитории и их реализации для пользователя, истории и почты:

interface UserRepository {  
 fun getAllUsers(): Flow<List<User>>  
  
 suspend fun getUserById(id: Int): Flow<User?>  
  
 suspend fun getUserByLogin(login: String): User?  
  
 suspend fun insertUser(user: User)  
  
 suspend fun updateUser(user: User)  
  
 suspend fun deleteUser(user: User)  
  
 suspend fun getUserByLoginAndPassword(login: String, password: String): User?  
}

interface CardRepository {  
 fun getAllCards(): Flow<PagingData<Card>>  
  
 fun getCardsByUserId(userId: Int): Flow<PagingData<Card>>  
  
 fun getCardById(id: Int): Flow<Card?>  
  
 suspend fun insertCard(card: Card)  
  
 suspend fun updateCard(card: Card)  
  
 suspend fun deleteCard(card: Card)  
}

class OfflineUserRepository(private val userDao: UserDao): UserRepository {  
 override fun getAllUsers(): Flow<List<User>> = userDao.getAll()  
  
 override suspend fun getUserById(id: Int): Flow<User?> = userDao.getById(id)  
  
 override suspend fun getUserByLogin(login: String): User? = userDao.getByLogin(login)  
  
 override suspend fun insertUser(user: User) = userDao.insert(user)  
  
 override suspend fun updateUser(user: User) = userDao.update(user)  
  
 override suspend fun deleteUser(user: User) = userDao.delete(user)  
  
 override suspend fun getUserByLoginAndPassword(login: String, password: String): User? = userDao.getUserByLoginAndPassword(login, password)  
}

class OfflineCardRepository(private val cardDao: CardDao): CardRepository {  
 override fun getAllCards(): Flow<PagingData<Card>> {  
 return Pager(  
 config = PagingConfig(  
 pageSize = 5,  
 prefetchDistance = 1,  
 enablePlaceholders = true,  
 initialLoadSize = 10,  
 maxSize = 15  
 ),  
 pagingSourceFactory = **{** cardDao.getAll()  
 **}** ).flow  
 }  
  
 override fun getCardsByUserId(UserId: Int):Flow<PagingData<Card>> {  
 return Pager(  
 config = PagingConfig(  
 pageSize = 5,  
 prefetchDistance = 1,  
 enablePlaceholders = true,  
 initialLoadSize = 10,  
 maxSize = 15  
 ),  
 pagingSourceFactory = **{** cardDao.getByUserId(UserId)  
 **}** ).flow  
 }  
  
 override fun getCardById(id: Int): Flow<Card?> = cardDao.getById(id)  
  
 override suspend fun insertCard(card: Card) = cardDao.insert(card)  
  
 override suspend fun updateCard(card: Card) = cardDao.update(card)  
  
 override suspend fun deleteCard(card: Card) = cardDao.delete(card)

2. Были разработаны классы ViewModel для реализации бизнес-логики, в них была перенесена и доработана логика из представлений, слой UI теперь обращается к этим классам:

class UserViewModel(private val userRepository: UserRepository): ViewModel() {  
 val getAllUsers = userRepository.getAllUsers()  
  
 suspend fun getUser(id: Int): Flow<User?> = userRepository.getUserById(id)  
  
 fun updateUser(user: User) = *viewModelScope*.*launch* **{** userRepository.updateUser(user)  
 **}** fun deleteUser(user: User) = *viewModelScope*.*launch* **{** userRepository.deleteUser(user)  
 **}** suspend fun getUserByLogin(login: String): User? = userRepository.getUserByLogin(login)  
  
 fun regUser(user: User) = *viewModelScope*.*launch* **{** val globalUser = userRepository.getUserByLogin(user.login)  
 globalUser?.*let* **{** return@launch  
 **}** ?: *run* **{** if(user.password.*isEmpty*()){  
 return@launch  
 }  
  
 userRepository.insertUser(user)  
 GlobalUser.getInstance().setUser(userRepository.getUserByLogin(user.login))  
 **}  
 }** fun authUser(user: User) = *viewModelScope*.*launch* **{** val globalUser = userRepository.getUserByLogin(user.login)  
 if (user.password.*isNotEmpty*() && user.password == globalUser?.password){  
 GlobalUser.getInstance().setUser(globalUser)  
 }  
 **}** private fun isValidEmail(email: String): Boolean {  
 return android.util.Patterns.*EMAIL\_ADDRESS*.matcher(email).matches()  
 }  
}

class CardViewModel(private val cardRepository : CardRepository):ViewModel() {  
 val getAllCards: Flow<PagingData<Card>> = cardRepository.getAllCards().*cachedIn*(*viewModelScope*)  
  
 fun getCardById(id: Int): Flow<Card?> = cardRepository.getCardById(id)  
  
 fun getCardByUserId(userId: Int): Flow<PagingData<Card>> = cardRepository.getCardsByUserId(userId).*cachedIn*(*viewModelScope*)  
  
 fun insertCard(card: Card) = *viewModelScope*.*launch* **{** cardRepository.insertCard(card) **}** fun updateCard(card: Card) = *viewModelScope*.*launch* **{** cardRepository.updateCard(card) **}** fun deleteCard(card: Card) = *viewModelScope*.*launch* **{** cardRepository.deleteCard(card) **}**}

3. Был внедрен механизм пагинации для списков автомобилей, это было сделано для того, чтобы данные подгружались частями незаметно для пользователя, не тормозя приложение:

В интерфейсе-репозитории

@Query("select \* from cards")  
fun getAll(): PagingSource<Int, Card>

В реализации интерфейса

override fun getAllCards(): Flow<PagingData<Card>> {  
 return Pager(  
 config = PagingConfig(  
 pageSize = 5,  
 prefetchDistance = 1,  
 enablePlaceholders = true,  
 initialLoadSize = 10,  
 maxSize = 15  
 ),  
 pagingSourceFactory = **{** cardDao.getAll()  
 **}** ).flow  
}

В ViewModel-классе

fun getAllCards(): Flow<PagingData<Card>>

4. Демонстрация работы приложения. Внешний вид программы не изменился с прошлой лабораторной.