Министерство образования Новосибирской области
ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «Электронный гардероб»

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения

для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.202000.010.000ПЗ

Разработал:

Некрасов Я. А.

2024

Содержание

Введение

Данный курсовой проект является актуальным, так как по анализу самых популярных маркетплейсов России было выяснено, что такие компании как «Wildberries», «Ozon» делают покупки в интернете более доступными, а также то, что обе эти компании делают большой упор на продажу одежды. Благодаря этим сервисам у россиян становится больше одежды и приложение «Электронный гардероб» призвано служить решением для ее систематизации.

Наименование программного продукта «Электронный гардероб». Продукт представляет собой мобильное, полностью бесплатное решение для систематизации своей одежды и образов.

Разрабатываемое мобильное приложение позволит пользователям хранить, сортировать и составлять в образы их одежду для более эффективного ее использования. Предполагаемая потребность обусловлена тем, что при анализе приложений с похожей тематикой не найдено конкурентов, которые бы полностью устраивали пользователей.

Целью курсовой работы является разработка веб-приложения с адаптацией под мобильное устройство.

Для достижения цели необходимо выполнить следующие задачи:

- проанализировать приложения-конкуренты;
- спроектировать дизайн мобильного приложения;
- изучить такие языки программирования и разметки, как: Kotlin, xml;
- изучить библиотеки Room и Navigation;
- разработать функционал мобильного приложения.

Реализуемый проект – мобильное приложение «Электронный гардероб»

Исследовательский раздел

1.1 Описание предметной области

Тема электронного гардероба является актуальной и востребованной в современном мире. С развитием технологий люди все больше времени проводят в Интернете, и это касается не только работы и развлечений, но и моды. Такие сервисы как: «Wildberries», «Ozon», «Lamoda», и другие делают покупку одежды более доступной и простой. Электронный гардероб позволяет людям систематизировать свою одежду, получать новые идеи для образов и быть в курсе последних трендов.

1.2 Образ клиента

Для разработки электронного гардероба необходимо определить образ клиента, то есть целевую аудиторию приложения. Целевая аудитория электронного гардероба может быть представлена следующими группами пользователей:

люди, которые следят за модой и хотят выглядеть стильно. Эти пользователи будут заинтересованы в использовании электронного гардероба для хранения информации о своей одежде, составления образов;

люди, которые хотят сэкономить время на сборе гардероба. Эти пользователи будут использовать электронный гардероб для быстрой сортировки одежды и составления образов;

люди, которым нужна помощь в ведении собственного гардероба. Эти пользователи будут использовать электронный гардероб для поиска новых идей для образов и получения систематизированной информации об их одежде;

люди, нуждающиеся в помощи в организации гардероба семьи (многодетные родители);

1.3 Сценарии использования

У Светланы в скором времени корпоратив, на котором она должна быть одета определенным образом. Чтобы не перемерять все вещи в поисках того, что лучше сидит и с чем сочетается она открывает приложение «Электронный гардероб» и может выбирать из уже заготовленных вариантов образа, или создать новый.

Станислава – многодетная мать и хочет оптимизировать время, которое она тратит на сбор своих детей. Она может воспользоваться приложением «Электронный гардероб» и в описании ко всем вещам расписать, чья это вещь и кто из ее детей не хочет такое носить.

Григорий – любитель моды и хочет систематизировать свой подход к одежде. Она может воспользоваться приложением «Электронный гардероб» и не забывать о всех возможностях его гардероба.

1.4 Сбор и анализ прототипов

В GooglePlay и AppStore существует два приложения, выполняющих функции электронного гардероба:

– Getwardrobe;

– Acloset;

Оба этих приложения имеют главный экран, на котором собраны видеоролики от разных авторов (рисунок 1).

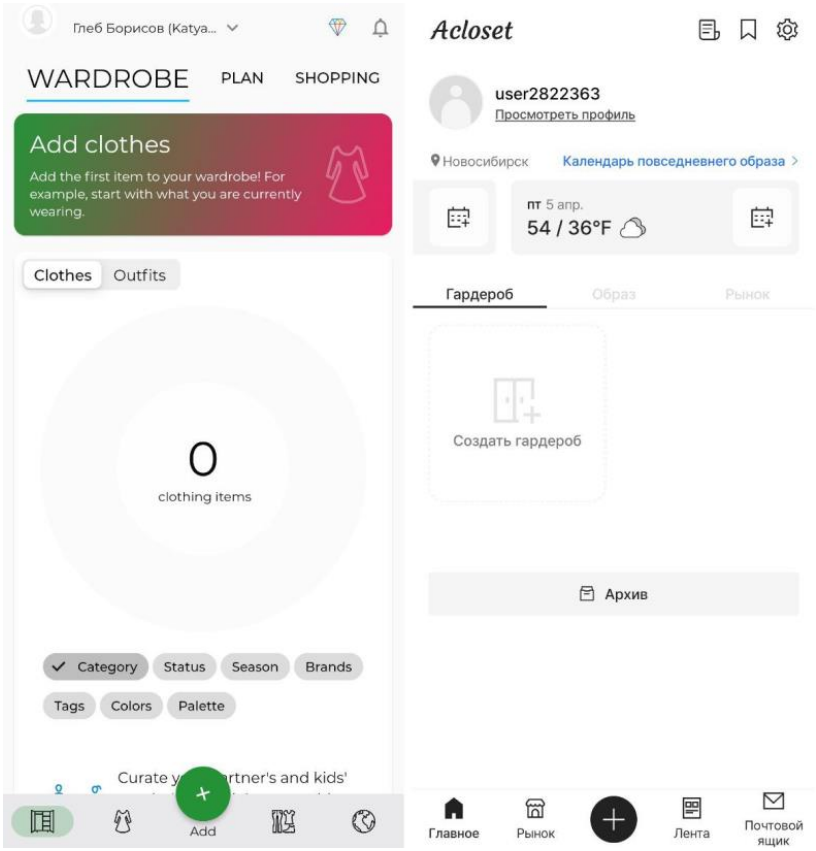


Рисунок 1 – Главный экран приложений-конкурентов

В приложениях на главном экране можно увидеть нижнюю и верхнюю панель, отвечающие за навигацию по приложению. На верхней панели в обоих приложениях почти одинаковые кнопки:

- профиль;
- уведомления.

Нижняя панель отличается только элементами, к которым она перенаправляет, но общими являются:

- добавить;
- главный экран;
- лента с новостями от других пользователей.

Основные критерии приложений, вынесенные из использования представлены в таблице 1.

Таблица 1 – Сравнение приложений

Параметры	Getwardrobe	Acloset
Стоимость	Бесплатно, но с возможностью оформления подписки	Бесплатно, но с покупками внутри приложения
Онлайн функционал	Есть	Есть
Необходимость постоянного подключения к интернету	Нет, но функционал в офлайн режиме ограничен	Есть
Добавление тегов (меток) к одежде	Есть	Есть
Добавление тегов, не предусмотренных разработчиками	Отсутствует	Частично отсутствует

1.4.1 Основные негативные отзывы от пользователей приложений

Часто повторяющиеся жалобы пользователей приложений-конкурентов:

- возможность добавить только 100 предметов одежды, образов (расширяется с подпиской);
- цена подписки;
- количество рекламы;
- скорость работы приложения;
- качество работы нейросетей.

Проектирование приложения

1.5 UI/UX дизайн приложения

Дизайн проекта разработан в программе Figma.

Для проекта были определены основные экраны:

- просмотр одежды;
- просмотр образов;
- добавление одежды;
- добавление образов;
- редактирование одежды;
- редактирование образов;
- поиск одежды;
- поиск образов;
- добавление одежды в образ;
- просмотр одежды, добавленной в образ.

Для тем приложения определены следующие две цветовые схемы. Первая тема реализуется в тёмных тонах, поэтому в ней основными цветами являются: серый, черный, белый (Рисунок 2).

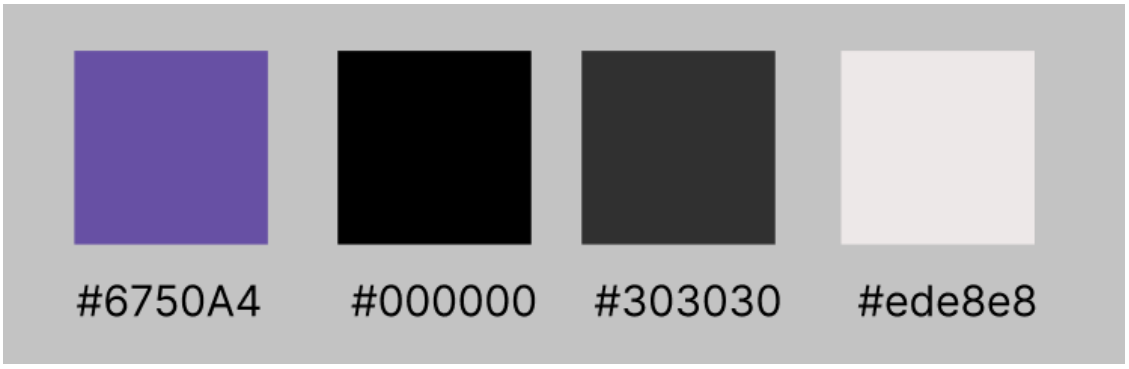


Рисунок 2 – Цветовая схема

Но стоит заметить, что данные цвета, за исключением фиолетового являются нейтральными и на их изображение одежды будет выделяться, но не будет выделяться из цветовой схемы. Фиолетовый цвет сообщает пользователю об основных интерактивных элементах, таких как кнопки подтверждения, добавления.

Ниже на рисунке 3 представлен логотип приложения.

Рисунок 3 – Логотип приложения

Цветовая схема логотипа состоит из: основного цвета приложения - фиолетового, и контрастного для него белого цвета это отсылает нас сразу к нескольким вещам:

– логотип «Wildberries» (рисунок 4) как основного маркетплейса России и основного продавца одежды для потенциальных клиентов

– вешалки «bagis» (рисунок 5), как символ домашнего и доступного дизайна

Рисунок 4 – Логотип Wildberries

Рисунок 5 – вешалки «bagis»

Определившись с цветовой схемой приложения и создав его логотип, был разработан дизайн следующих экранов:

- одежда (ListFragment)
- добавление одежды (AddFragment)
- изменение одежды (UpdateFragment)
- поиск одежды (SearchFragment)
- наряды (OutfitListFragment)
- добавление одежды (AddOutfitFragment)
- изменение одежды (UpdateOutfitFragment)
- добавление одежды в наряд (AddToOutfitFragment)
- просмотр одежды содержащейся в наряде (MoreAboutOutfitFragment)

Ниже на рисунке 6 представлен дизайн приложения.

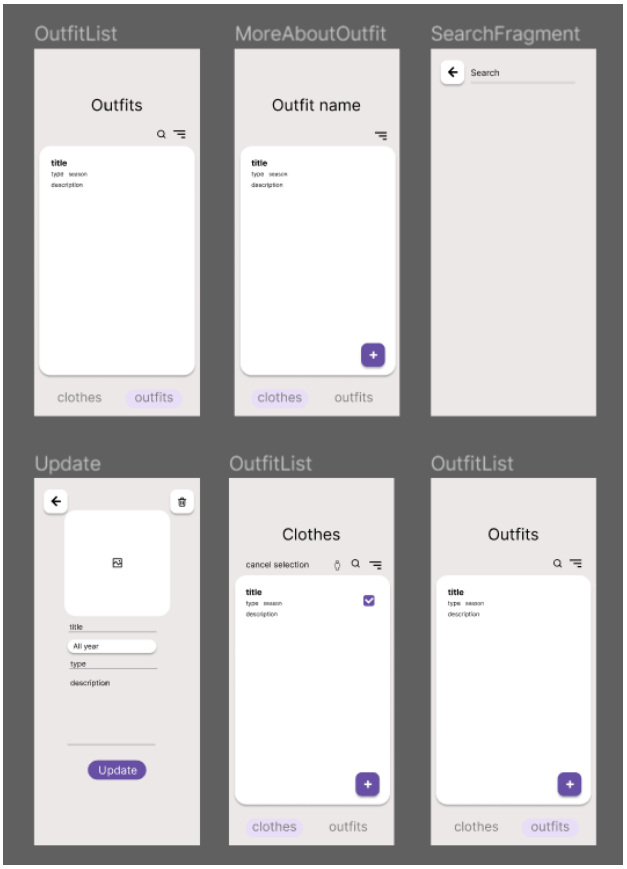


Рисунок 6 – Дизайн приложения

На рисунке 7 показано перемещение пользователя в приложении.

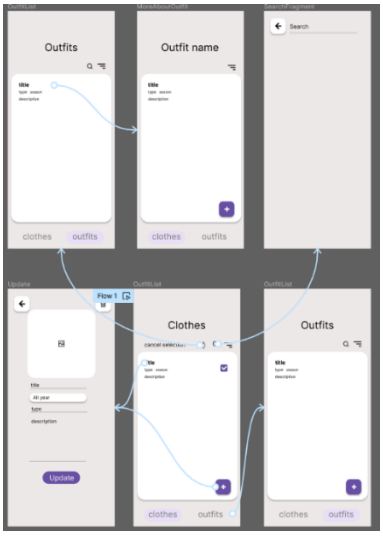


Рисунок 7 – Перемещение пользователя в приложении

При первом запуске приложения пользователь попадает на экран просмотра Элементов одежды. При первом входе пользователь еще не добавил ничего в базу данных, и приложение дает ему подсказку к дальнейшим действиям – надпись «Одежда» меняется на «Нажмите на кнопку “Плюс”». это показано на рисунке 8. Так же эта надпись, как и все приложение оно переводится на английский если язык устройства пользователя – не русский. После добавления надпись изменится обратно. После у пользователя есть несколько путей взаимодействия с приложением на выбор: добавить еще одежды, добавить образ и привязать к нему одежду, изменить уже добавленную одежду и.т.д.

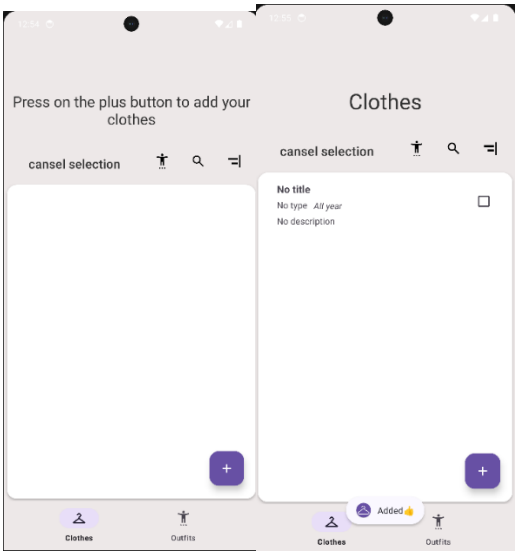


Рисунок 8– Изменение главного экрана от действий пользователя

Так же реализованы разные элементы списков для отображения записей базы данных в разных условиях. Все их вариации показаны на рисунке 9

Рисунок 9 – Все версии элементов списка

Для отображения этой разметки в проект были добавлены «Адаптеры». Они необходимы в приложении для связи фрагмента и разметки для элементов списка recyclerView.

2.2 Выбор технологии, языка и среды программирования

Средой программирования выбрана программа Android Studio, так как она довольно удобна для создания приложения и является довольно популярной, поэтому в случае возникновения ошибки, легче найти способ её решения.

Языком программирования является Kotlin, так как это новый и развивающийся язык программирования, к которому очень часто добавляются новые удобные функции, которые пригодятся во время разработки. Так же Kotlin поддерживает корутины из асинхронного программирования, что очень важно для приложения, основой которого является база данных.

Для реализации базы данных была выбрана библиотека Room. Она незначительно увеличивает размер установочного файла по сравнению с другими решениями. Так же Room позволяет писать более сложные запросы с использованием SQL.

Также для реализации переходов между экранами была добавлена библиотека Navigation. Что позволяет работать не с активностями, а с фрагментами, что в незначительной степени изменяет написание кода, но позволяет создавать переходы между экранами с использованием визуального интерфейса, что изображено на рисунке 10. В дополнении к Navigation был установлен плагин kotlin-parcelize, который позволяет передавать между экранами объекты классов.

2.3 Разработка базы данных

Для того, чтобы пользователь мог добавить несколько единиц одежды к одному образу и один элемент одежды в несколько образов была реализована связь: «Многие ко многим» между сущностями: ClothingItem и Outfit. Далее представлен фрагмент кода – сущность базы данных «ClothingItemt»

```
@Parcelize
@Entity(tableName = "ClothingItem")
data class ClothingItem (
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    val image: String?,
    val title: String,
    val type: String,
    val season: String,
    val description: String,
    val dateUpdated: Long
```

Ниже представлен фрагмент кода – сущность базы данных «Outfit»

```
@Parcelize
@Entity(tableName = "Outfit")
data class Outfit(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val image: String?,
    val title: String,
    val season: String,
    val style: String?,
    val description: String,
    val dateUpdated: Long
```

Для реализации базы данных с помощью Room также необходимы:

– DAO. Это интерфейсы, создаваемые для каждой сущности, в них прописываются все запросы и методы для взаимодействия с таблицей базы данных;

– класс базы данных. К нему подключаются DAO и сущности, так же через него будет разорвана связь между DAO и репозиторием;

– репозиторий. Это необязательный компонент, но он является «правилом хорошего тона». В нем прописывается асинхронность методам из DAO;

– view model. Это тот компонент, к которому будут обращаться фрагменты для получения информации для отображения пользователю.

Схематично взаимодействие всех этих компонентов с пользователем изображено на рисунке 10.

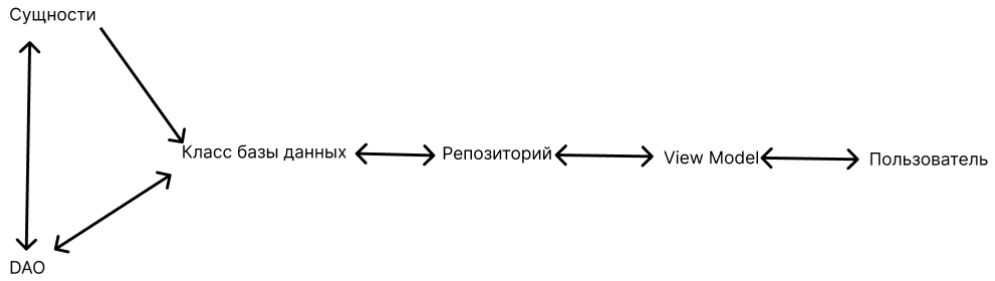


Рисунок 10 – Взаимодействие элементов в Room

2.4 Разработка мультимедийного контента

Весь мультимедийный контент разрабатывался с помощью языка разметки XML. Вёрстка выполнялась по дизайну, разработанному ранее в приложении Figma. Однако, по мере разработки, в дизайн были внесены изменения.

Иконки, и другие элементы приложения, были импортированы в проект в виде XML-файлов, это показано на рисунке 11. Такой способ хранения уменьшает вес приложения, а также, избавляет от проблем с потерей качества мультимедийного контента. Все иконки хранятся в папке «drawable», все элементы списков в «layout».

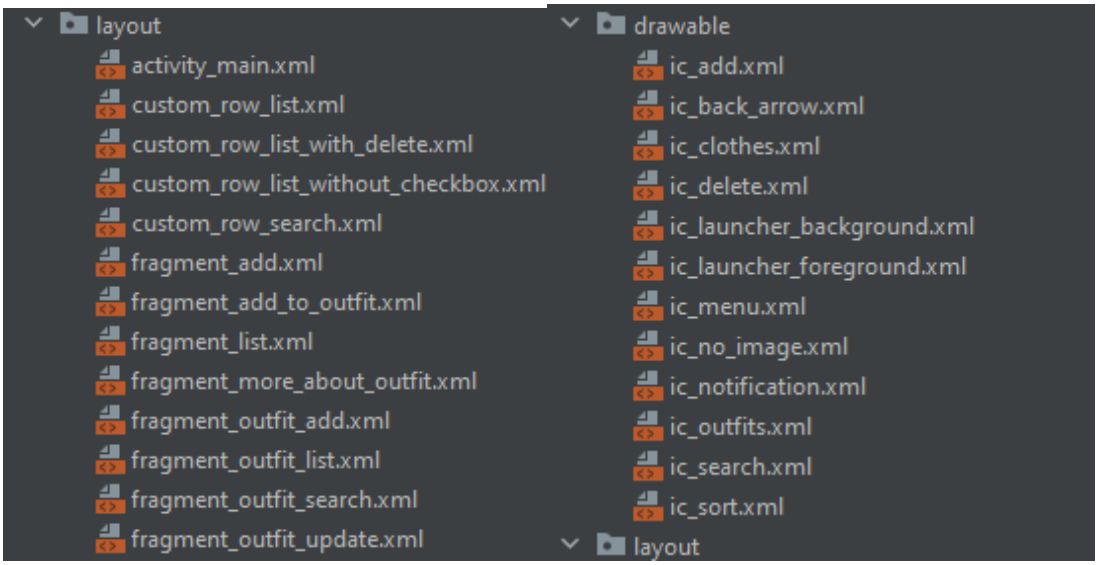


Рисунок 11 – Мультимедийный контент

2.5 Описание используемых плагинов

В проекте используются библиотеки с различными компонентами и функциями. Список всех библиотек, а также их описание представлен в таблице 2.

Таблица 2 – Библиотеки и их описание

Библиотека	Описание
1	2
com.google.firebase:firebase-bom:31.3.0	Обеспечивает согласованную версию всех Firebase-библиотек.
kotlin-kapt	Обеспечивает обработку аннотаций, этот плагин не будет использоваться напрямую, он необходим для работы Room
kotlin-parcelize	Позволяет передавать объекты классов между экранами.
androidx.navigation: navigation-fragment-ktx:2.5.3, androidx.navigation: navigation-ui-ktx:2.5.3	Библиотеки, предоставляющие навигационные компоненты для управления переходами и навигацией между фрагментами и активностями в приложении.

Таблица 2 – Окончание

1	2
androidx.lifecycle: lifecycle-livedata-ktx:2.6.1, androidx.lifecycle: lifecycle-viewmodel-ktx:2.6.1	Библиотеки, обеспечивающие жизненный цикл компонентов Android (таких как активности и фрагменты) и предоставляющие LiveData и ViewModel для удобной работы с данными в приложении.
org.jetbrains.kotlinx:kotlinx-coroutines-core	Богатая библиотека для корутин, разработанная JetBrains. Она содержит ряд высокоуровневых примитивов с поддержкой корутин, которые рассматриваются в этом руководстве, включая launch , async и другие.

Данные библиотеки являются неотъемлемой частью приложения, обеспечивая его правильное функционирование и реализацию всех задуманных возможностей. Без этих библиотек приложение не сможет работать в полной мере, так как они предоставляют необходимые инструменты и функции, которые необходимы для его работы.

2.6 Описание разработанных процедур и функций

В приложении реализованы следующие методы (таблица 3):

Таблица 3 – Методы приложения

Метод	Описание
1	2
textViewLimit()	Ограничение максимальной длины отображаемого текста, отображаемого пользователю. Если текст больше длины ограничения, то текст отображается не полностью
showToast()	Метод позволяющий вызвать объекты «Toast» которые не перекрывают друг друга
saveImageToDatabase()	Сохраняет выбранное пользователем изображение для дальнейшего хранения

Таблица 3 – Окончание

1	2
setImageToImageButton()	При нажатии на некоторые ImageButton пользователь может выбрать изображение из памяти устройства и установить его на ранее нажатый ImageButton, также выбранное изображение сохранится для дальнейшего использования
openGalleryForImage()	Выбор изображения из галереи
setData()	Передает данные в адаптер для дальнейшего показа данных пользователю
finallyDeleteItem()	Удаление записи из базы данных
replaceCurrentImage()	Замена изображения в записи базы данных
updateItem()	Обновление записи базы данных
getDatabase()	Получение экземпляра базы данных
updateAdapter()	Обновление состояния адаптера

Добавление записей в базу данных происходит через обращение к viewModel.

```
private lateinit var mClothingItemView: ClothesViewModel
private fun insertDataToDatabase(view: View) {
            val title = titleInput.text.toString().takeIf { 
it.isNotBlank() } ?: "No title"
            val season = seasonSpinner.selectedItem.toString()
            val type = typeImput.text.toString().takeIf { 
it.isNotBlank() } ?: "No type"
            val description = 
descriptionInput.text.toString().takeIf { it.isNotBlank() } ?: 
"No description"

            val clothingItem = ClothingItem(
                id = 0,
                image = currentImagePath,
                title = title,
                type = type,
                season = season,
```

```
                description = description,
                dateUpdated = System.currentTimeMillis()
            )
            mClothingItemView.addClothingItem(clothingItem)
            showToast(getString(R.string.on_added_message))

findNavController().navigate(R.id.action_addFragment_to_listFrag
ment)
        }
```

В случае, если пользователь не записал ничего в поля «title», «type» «description», будут подставлены значения «No title», «No type» «No description» соответственно.

Для получения изображения используются следующие методы:

```
private var currentImagePath: String? = null
imageButton = view.findViewById(R.id.add_imageButton)
            imageButton.setOnClickListener {
                handleImageSelection()
            }
private fun handleImageSelection() {
        if (currentImagePath != null) {
                replaceCurrentImage(currentImagePath)
            }
            openGalleryForImage()
        }
private fun openGalleryForImage() {
        val intent = Intent(Intent.ACTION_PICK)
        intent.type = "image/*"
        resultLauncher.launch(intent)
    }

    private val resultLauncher =

registerForActivityResult(ActivityResultContracts.StartActivityF
orResult()) { result ->
```

```
            if (result.resultCode == Activity.RESULT_OK) {
                result.data?.data?.let { uri ->
                    try {
                        currentImagePath =
saveImageToDatabase(uri)
                        currentImagePath?.let { path ->
                            setImageToImageButton(path)
                        }
                    } catch (e: IOException) {
                        e.printStackTrace()
                    }
                }
            }
        }
@Throws(IOException::class)
    private fun saveImageToDatabase(uri: Uri): String {
        val inputStream =
requireActivity().contentResolver.openInputStream(uri)
        return inputStream?.use { stream ->
            val selectedImageBitmap =
BitmapFactory.decodeStream(stream)

            // Масштабирование изображения до 575x575 пикселей
            val scaledBitmap =
Bitmap.createScaledBitmap(selectedImageBitmap, 575, 575, false)

            val storageDir = File(

requireActivity().getExternalFilesDir(Environment.DIRECTORY_DOWN
LOADS),
                "Wardrobe app"
            )
            if (!storageDir.exists()) {
                storageDir.mkdirs()
            }
```

```
            val imageFile = File.createTempFile("IMG_", ".png",
storageDir)
            val outputStream = FileOutputStream(imageFile)
            scaledBitmap.compress(Bitmap.CompressFormat.PNG, 80,
outputStream)
            outputStream.close()
            imageFile.absolutePath
        } ?: throw IOException("Input stream is null")
    }
private fun setImageToImageButton(imagePath: String) {
        val imageFile = File(imagePath)
        if (imageFile.exists()) {
            val imageBitmap =
BitmapFactory.decodeFile(imagePath)
            imageButton.setImageBitmap(imageBitmap)
        } else {
            showToast(getString(R.string.missed_image_error))
        }
    }
private fun replaceCurrentImage(imagePath: String?) {
        lifecycleScope.launch(Dispatchers.IO) {
            val isImagePathUsed =
mClothingItemView.isClothesImagePathUsed(currentImagePath)
            if (!isImagePathUsed) {
                deleteImage(currentImagePath)
            }
        }
        showToast("Previous image is cleared")
    }
```

Изначально переменная «currentImagePath» имеет значение null. При нажатии на imageButton используется метод «handleImageSelection». Если currentImagePath не равно null, то используется метод «replaceCurrentImage», который проверяет, не используется ли в других записях это изображение. Если изображение используется только в этой записи, то оно удаляется. Далее открывается галерея. Если при этом не возникло ошибок, то к полученному

изображению получается ссылка. И изображение по этой ссылке сохраняется в файлы приложения в формате png 575 на 575, что позволяет хранить изображение в хорошем качестве и не использовать много памяти устройства. Файл передается как изображение для imageButton для того, что пользователь видел подтверждение того, что в процессе не произошло ошибок.

Подобные методы повторяются для нарядов и не нуждаются в повторении. После добавления одежды и наряда пользователь может добавить в наряд элементы одежды.

Для вышеупомянутого функционала на элементах списка одежды расположены comboBox. Далее при нажатии на кнопку добавления одежды в образ, которая показана на рисунке 12, приложение перенаправляет пользователя на следующий фрагмент, на котором изображены все образы, добавленные им в приложение.

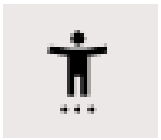


Рисунок 12 – Кнопка добавления одежды в образ

При нажатии на recyclerView, создаются записи, в сущности, базы данных «ClothingItemOutfitCrossRef», которая отвечает за связь между сущностями. По завершении создания пользователя перенаправят на экран просмотра одежды, содержащейся в образе, он показан на рисунке 13.

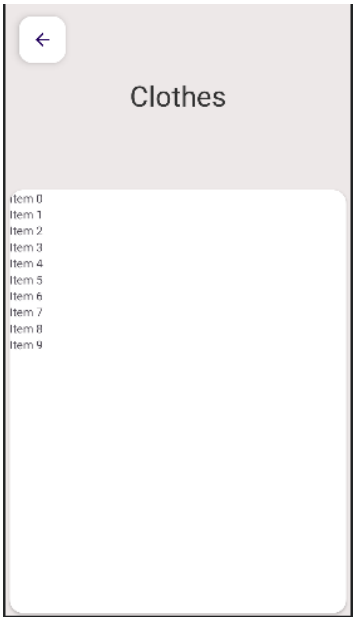


Рисунок 13 – Экран просмотра одежды, содержащейся в образе

Так же на этот экран можно попасть, через просмотр образа. В нижней части приложения реализован bottomNavigationView, который позволяет переключаться пользователю между просмотром одежды и образов. При нажатии на элементы списков для отображения пользователя направит на экран обновления и просмотра записей в базе данных. Подобные экраны почти аналогичны для clothingItem и outfit, они показаны на рисунке 14.

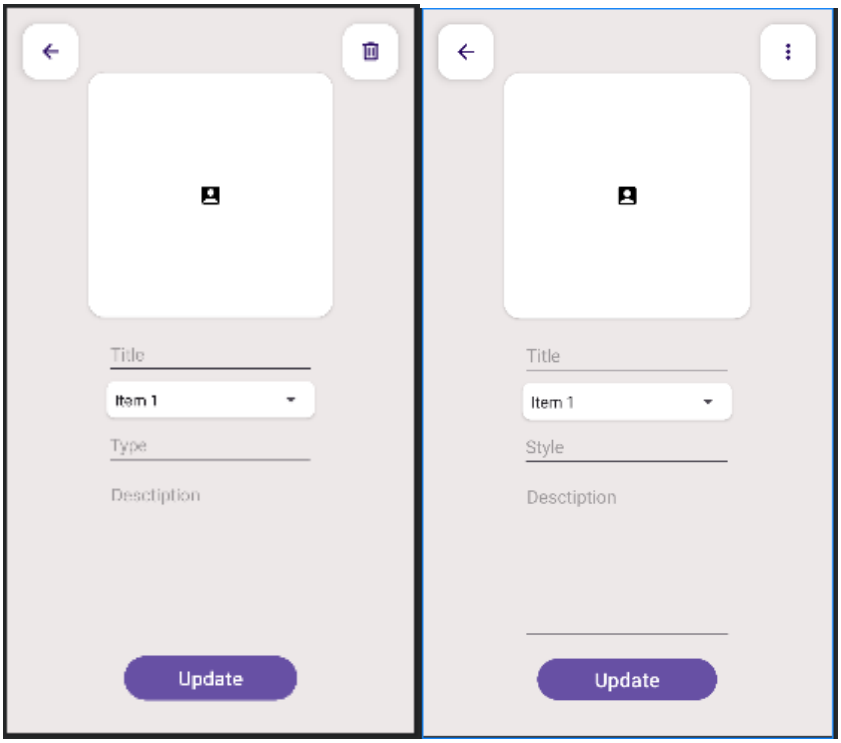


Рисунок 14 – Экран Обновления одежды (слева) и образа (справа)

Как видно, их различие состоит в кнопке в правом верхнем углу. На окне с обновлением образов нажатие открывает меню со следующими опциями:

- подробнее. При нажатии на этот пункт меню пользователю будет показана одежда, связанная с этим нарядом;
- удалить. При нажатии на этот пункт меню пользователю будет показано окно для подтверждения удаления. Это окно показано на рисунке 15.

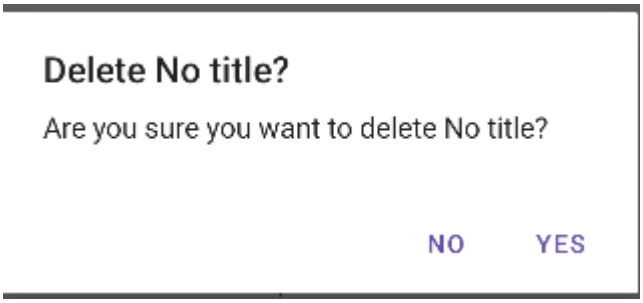


Рисунок 15 – Окно подтверждения удаления одежды или образа

Так как первый пункт меню логически не нужен на окне одежды, на нем нажатие на кнопку в правом верхнем углу выполняет те же функции, что и пункт меню «удалить».

Тестирование

2.7Протокол тестирования дизайна приложения

Тестирование дизайна приложения проводится на самом минимальном (Android SDK 28) и на более позднем (Android SDK 34) с различной диагональю экранов для проверки разметки страниц и вёрстки приложения.

Примеры проверок отображения элементов на экране представлены на рисунках 16–21.

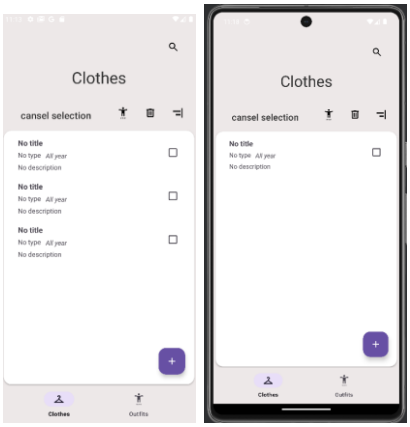


Рисунок 16 – Экраны просмотра одежды

На экране не обнаружено ошибок, все надписи и кнопки отображаются корректно

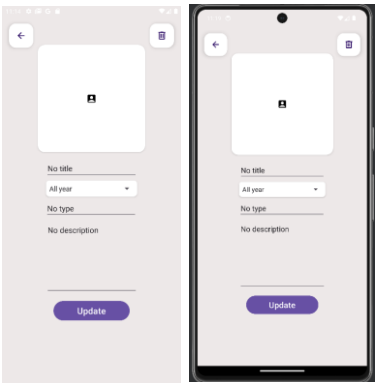


Рисунок 17 – Экраны обновления

Экран обновления так же отображается корректно на обоих устройствах, все элементы интерфейса расположены на своих местах.

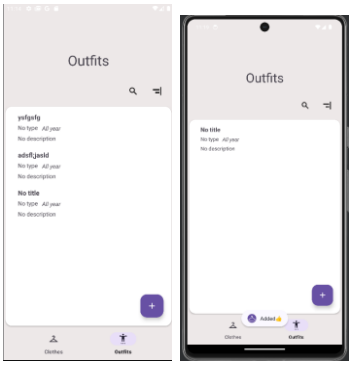


Рисунок 18 – Экраны просмотра образов

Экран просмотра образов тоже отображается корректно. Верхняя и нижняя панель нормальных размеров, элементы на них не сдвинуты. Логотип так же находится на своём месте.

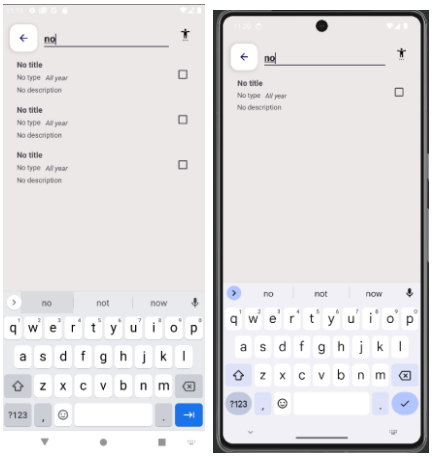


Рисунок 19 – Экраны поиска

Элементы экрана отображены корректно. Клавиатура не мешает разметке и не закрывает каких-либо элементов дизайна.

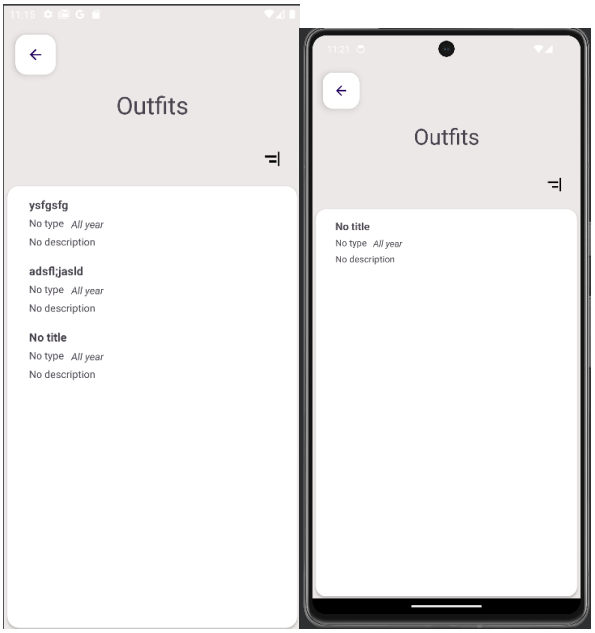


Рисунок 20 – Экраны добавления одежды в образ

Экраны отображены корректно, все элементы сохраняют свое положение в соответствии с разметкой.

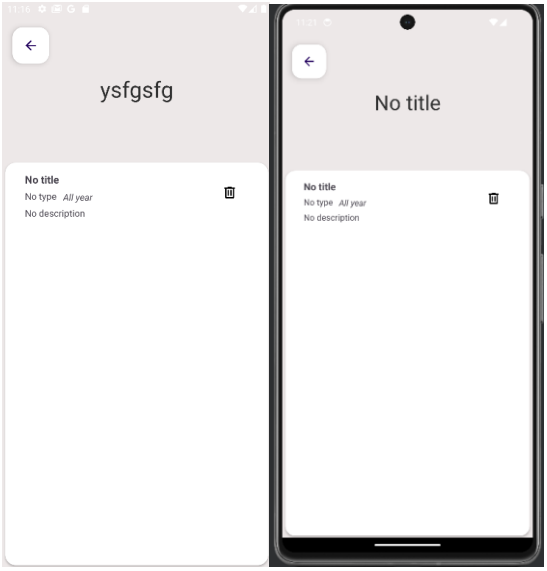


Рисунок 21 – Экраны просмотра одежды в образе

Экраны отображены корректно, все элементы сохраняют свое положение в соответствии с разметкой.

Также в приложении предусмотрена локализация на русский язык, который из-за его различий с английским может занимать различное пространство в разметке

Заключение

В ходе разработки мобильного приложения для систематизации одежды учтены предпочтения и потребности пользователей. Реализована система CRUD для основных сущностей базы данных.

Интерфейс приложения разработан с учетом удобства использования и минимального количества действий для достижения пользовательской цели. Основные экраны спроектированы таким образом, чтобы каждый элемент являлся функциональным.

Важным элементом концепции приложения является отсутствие ограничений накладываемых на пользователя приложениями-конкурентами, такие как: обязательное подключение к сети, ограничение бесплатного функционала приложения и.т.д.

Важным этапом в разработке приложения было тестирование дизайна, которое подтвердило его корректность и успешное функционирование. Это гарантирует, что разработанное приложение соответствует поставленным целям и требованиям.

В целом разработка электронного гардероба позволила создать функциональное и удобное приложение, учитывающее потребности пользователей. Проделанная работа успешно достигла поставленной цели курсового проекта и является основой для дальнейшего развития и улучшения приложения.

Библиография

Нормативно-правовые акты:

1 ГОСТ Р 2.105-2019. ЕСКД. Общие требования к текстовым документам. – Москва: Стандартинформ, 2019. – 36 с

Электронные ресурсы:

1 Android Developers [Электронный ресурс]. – Документация Android Studio. – URL: https://developer.android.com/develop (дата обращения: 26.04.2024)

2 KotlinLang [Электронный ресурс]. – Документация Kotlin. – URL: https://kotlinlang.org/docs/home.html (дата обращения: 26.04.2024)

3 Developer android [Электронный ресурс]. – Документация Room. Вытягивание данных с базы данных – URL: https://developer.android.com/training/data-storage/room (дата обращения: 26.04.2024)

4 Developer android [Электронный ресурс]. – Документация Navigation – URL: https://developer.android.com/guide/navigation (дата обращения: 26.04.2024)

5 Material Design [Электронный ресурс]. – Bottom Navigation. – URL: https://material.io/components/bottom-navigation/android (дата обращения: 26.04.2024)

Приложение А

(обязательное)

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

РАЗРАБОТКА МОБИЛЬНОЙ ВЕРСИИ ВИДЕОХОСТИНГА

Техническое задание

НАТКиГ.202000.010.000ПЗ

Выполнил:

Студент группы ПР-21.101

Некрасов Я. А

2024

Содержание

Введение

Настоящее техническое задание распространяется на разработку мобильного приложения «Электронный гардероб», используемого для просмотра систематизации одежды.

Наименование приложения: «Wardrobe» (при установленном на устройстве русском языке - Гардероб).

Краткая характеристика области применения: мобильное приложение предоставляет систематизации предметов одежды пользователя, поиск и объединение в образы

Основанием для проведения разработки является Протокол №6 от 21 февраля 2024 года.

Наименование темы разработки – «Разработка мобильного приложения «Электронный гардероб»».

Условное обозначение темы разработки – «Электронный гардероб».

1 Назначение разработки

Основное назначение приложения заключается в:

– обеспечении удобного и понятного интерфейса приложения для пользователя;

– предоставление большой библиотеки видеороликов.

Лица, которые могут работать с данной системой:

пользователь приложения – добавлять, удалять, редактировать записи об одежде и образах, осуществлять поиск и сортировать их. Объединять несколько предметов одежды в образ.

2 Требования к мобильному приложению

2.1 Требования к функциональным характеристикам

Требования к составу выполняемых функций:

– реализация взаимодействия с одеждой;

– реализация взаимодействия с образами;

– реализация связи между сущностями базы данных;

2.3 Условия эксплуатации

Пользователь должен иметь практические навыки использования мобильного устройства под управлением операционной системы Android.

2.4 Требования к составу и параметрам технических средств

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android с API не ниже 28.

2.5 Требования к информационной и программной совместимости

Проектирование взаимодействия с файловой системой должно быть выполнено в рамках разработки курсового проекта. При разработке взаимодействия с файловой системой должен быть использован язык программирования Kotlin.

2.6 Требования к защите информации

Вся информация, хранимая в базе данных расположена локально. Приложение не требует от пользователя дополнительных разрешений.

2.7 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

3 Требования к программной документации

Состав программной документации должен включать в себя:

– техническое задание;

– пояснительная записка.

4 Технико-экономические показатели

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

5 Стадии и этапы разработки

Таблица 1 – Стадии разработки

№	Этапы разработки КП	Сроки выполнения	Отчётность
1	Определение цели и задач, объекта и предмета исследования	12.11.2023	Пояснительная записка
2	Описание предметной области	17.11.2023	Пояснительная записка
3	Выбор технологии, языка и среды программирования	18.11.2023	Пояснительная записка
4	Оформление технического задания	7.12.2023	Техническое задание
5	Проектирование UI/UX дизайна	01.01.2024	Спецификации программного обеспечения
6	Разработка мобильного приложения	15.01.2024	Схема структурная системы и спецификации компонентов
7	Разработка базы данных	30.04.2024	Программный продукт
8	Отладка и тестирование приложения	15.04.2024	Тексты программных компонентов
9	Оформление документации	24.04.2024	Программная документация
10	Защита	30.04.2024	

6 Порядок контроля и приёмки

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

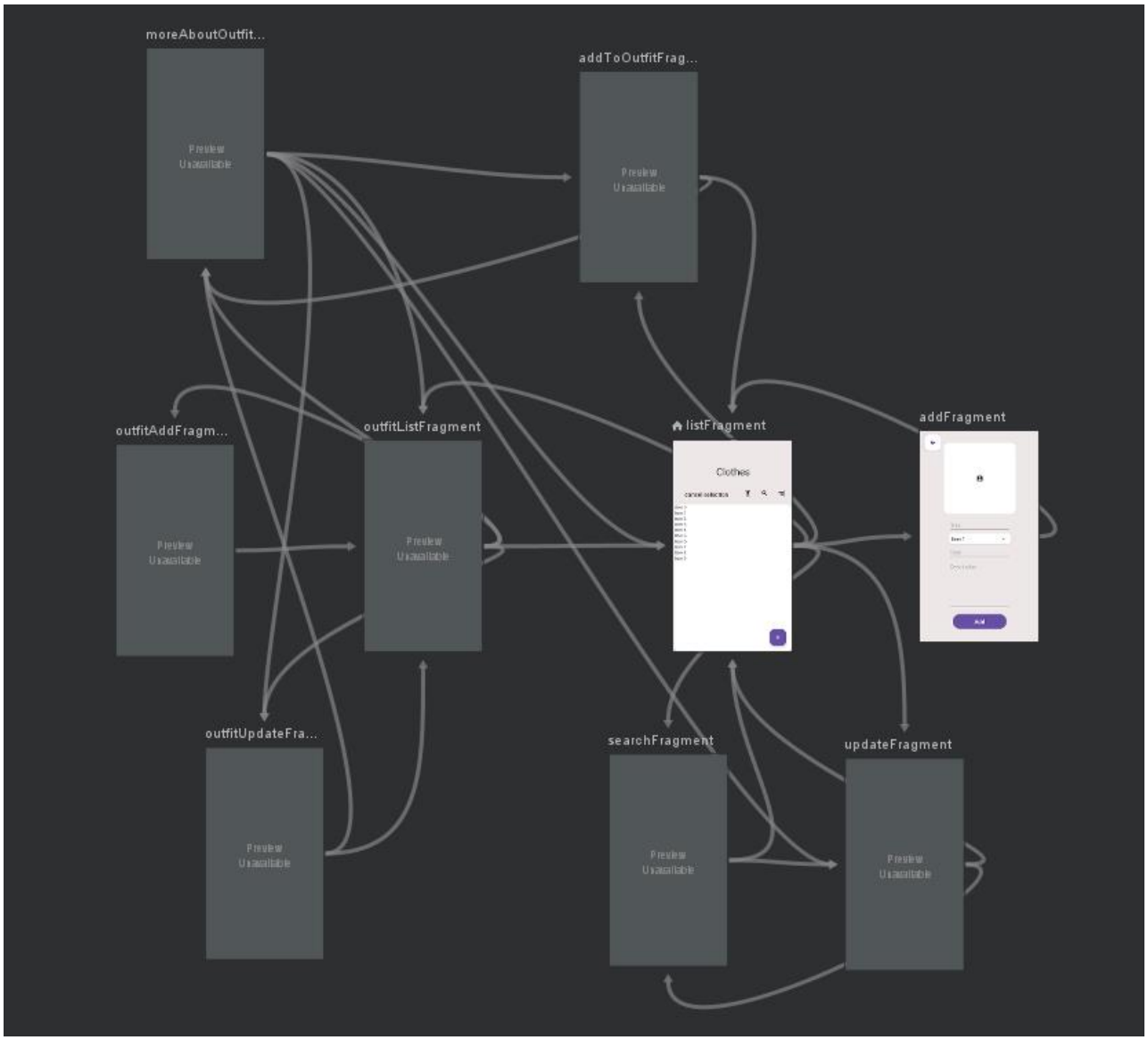
– техническое задание;

– пояснительная записка;

– программный продукт.

Приложение Б

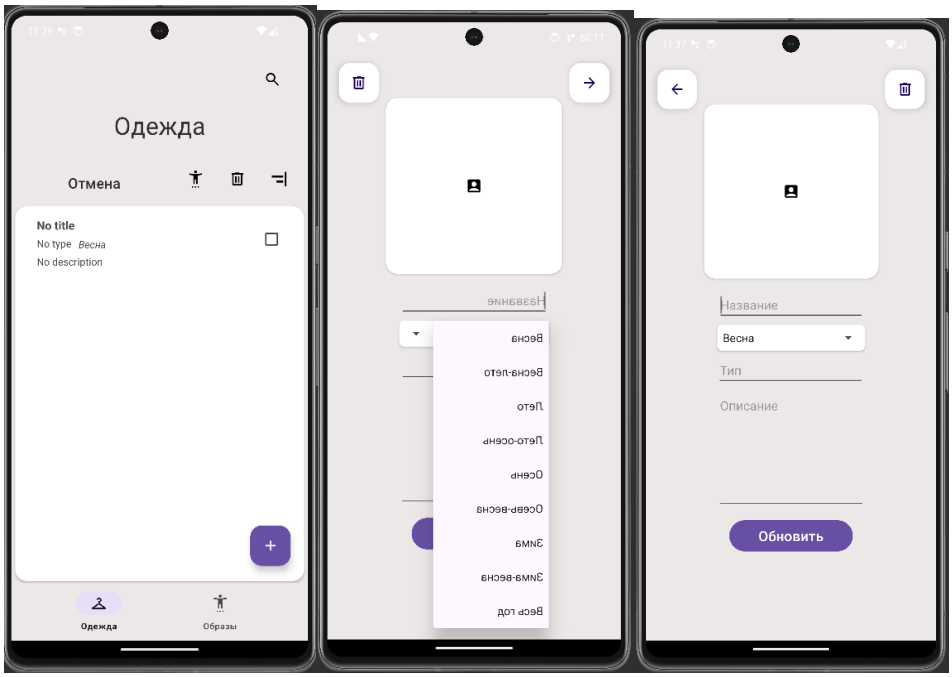
Навигация приложения через Navigation

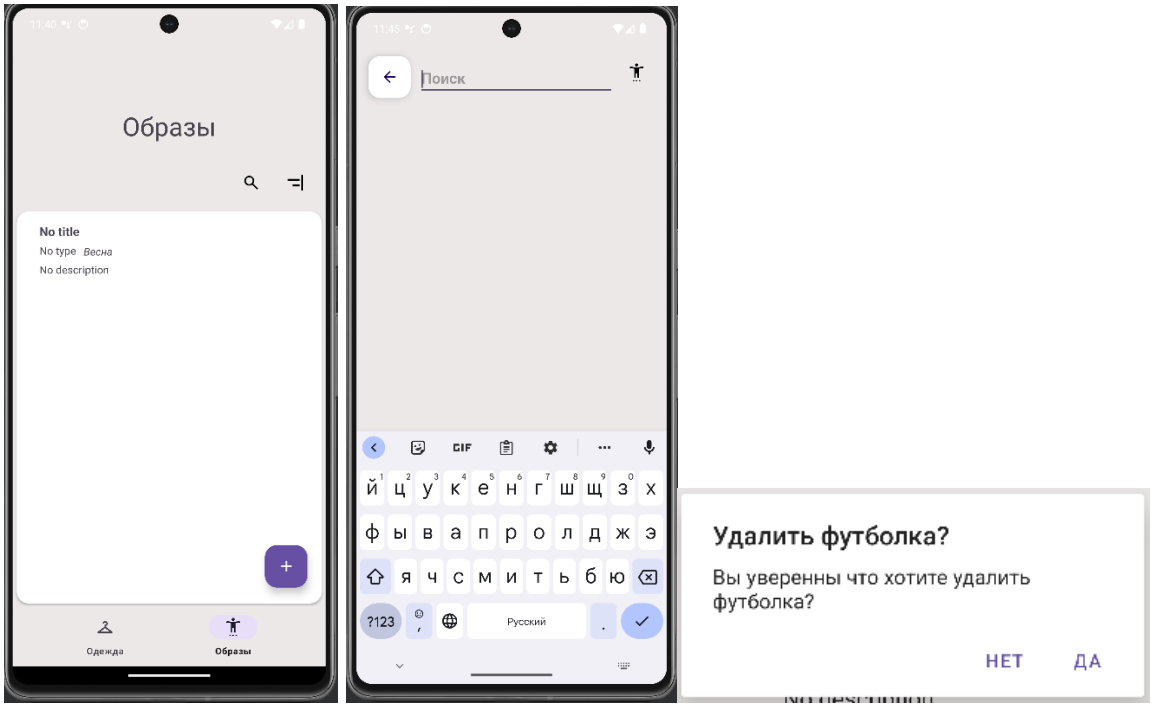


Приложение В

Локализация предложения

Ниже представлены рисунки 22 - 27, показывающие корректность работы разметки с локализованным текстов





Рисунки 22-27 – перевод приложения на русский язык