МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматики и вычислительной техники** |
| Кафедра | **Информатики** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка комиссии: |  | | | Рейтинг: |  |
| Подписи членов комиссии: | | | | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | | | | | |
| (дата) | | | | | |
|  | |  |  | | |

**КУРСОВАЯ РАБОТА**

|  |  |
| --- | --- |
| по дисциплине | Разработка мобильных и WEB приложений |
|  | |

|  |  |
| --- | --- |
| на тему | Мобильное приложение-каталог по игре «Genshin Impact» |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| «К ЗАЩИТЕ» |  | ВЫПОЛНИЛ: |  |
|  |  | Студент группы | **АА-20-05** |
|  |  |  | (номер группы) |
|  |  | Устинова Ульяна Павловна | |
| (должность, ученая степень; фамилия, и.о.) |  | (фамилия, имя, отчество) | |
|  |  |  | |
| (подпись) |  | (подпись) | |
|  |  | **14.05.2023** | |
| (дата) |  | (дата) | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Москва, 20 | 23 |  |

|  |
| --- |
|  |

МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматики и вычислительной техники** |
| Кафедра | **Информатики** |

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

|  |  |
| --- | --- |
| по дисциплине | Разработка мобильных и WEB приложений |
|  | |

|  |  |
| --- | --- |
| на тему | Мобильное приложение-каталог по игре «Genshin Impact» |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| ДАНО студенту | **Устиновой Ульяне Павловне** | группы | **АА-20-05** |
|  | (фамилия, имя, отчество в дательном падеже) |  | (номер группы) |

**Содержание работы:**

|  |  |
| --- | --- |
|  | Теоретическая часть |
|  | Практическая часть |
|  |  |

**Исходные данные для выполнения работы:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Рекомендуемая литература:**

|  |  |
| --- | --- |
|  | Саша Окунев «Руководство по Figma» |
|  | Antonio Leiva «Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App» |
|  |  |

**Графическая часть:**

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель: |  |  |  |  |  |  |  |
|  | (уч.степень) |  | (должность) |  | (подпись) |  | (фамилия, имя, отчество) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Задание принял к исполнению: | студент |  |  |  | Устинова Ульяна Павловна |
|  | |  | (подпись) |  | (фамилия, имя, отчество) |

**Оглавление**

[**Введение** 4](#_Toc135829082)

[**Теоретическая часть** 5](#_Toc135829083)

[**Практическая часть** 10](#_Toc135829084)

[**Заключение** 32](#_Toc135829085)

[**Список литературы** 33](#_Toc135829086)

## **Введение**

Современные люди, дети и взрослые, в течение своего дня занимаются учебой или работой, устают, тонут в рутине. Поэтому очень важно добавлять в свою жизнь развлечения, такие как, к примеру, видеоигры. "Genshin Impact" - одна из самых популярных игр на сегодняшний день. Это игра в жанре action-adventure с открытым миром и элементами RPG. Чтобы получать удовольствие от игры, необходимо знать её механики и правильно улучшать персонажей. Поэтому мобильные приложения с гайдами на персонажей являются актуальными и необходимыми для каждого игрока.

Цель данной курсовой работы - разработка и реализация мобильного приложения "GenshinChar" с информацией о персонажах игры "Genshin Impact" на платформе Android. Приложение предоставляет все необходимые игрокам знания о необходимых ресурсов для их персонажей.

Задачи, решаемые в этой работе, включают в себя составление технического задания к мобильному приложению, проектирование дизайна экранов, разработку красивого и удобного мобильного приложения и его тестирование с целью выявления и исправления ошибок, улучшения интерфейса.

Таким образом, приложение "GenshinChar" будет предоставлять собой удобный помощник для игроков, дающий возможность быстро получить необходимую, точную информацию о своём персонаже, чтобы приступить к его улучшению и получить удовольствие от игры.

## **Теоретическая часть**

Техническое задание (ТЗ, техзадание) — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями. Иными словами — это инструмент коммуникации между заказчиком и исполнителем, который помогает выстроить линию общения.

При составлении технического задания по выбранной теме необходимо:

1. Составить подробное описание функциональности будущего приложения.

2. Определить персонажи пользователей, описать пользовательские истории (User Story), составить карту путешествия пользователей (Customer Journey Map).

3. Сформировать технические требования к сервису.

Дизайн (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) design — проектировать, чертить, задумать, а также проект, план, рисунок) — деятельность по [проектированию](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [эстетических](https://ru.wikipedia.org/wiki/%D0%AD%D1%81%D1%82%D0%B5%D1%82%D0%B8%D0%BA%D0%B0) свойств [промышленных изделий](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82_(%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81)) («художественное конструирование»), а также результат этой деятельности.

Figma — онлайн-сервис для разработки интерфейсов и [прототипирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) сайтов и приложений. Интерфейс — это внешний вид продукта: кнопки, формы, иконки. Их можно создавать прямо в Figma и добавлять эффекты: создавать анимацию для баннеров, делать кнопки кликабельными. Прототип — модель будущего приложения или сайта. Это упрощённая версия продукта, по которой уже понятно, как люди будут пользоваться им. Ещё прототип можно адаптировать под планшеты и смартфоны.

Мобильное приложение ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) «Mobile application») или приложение для мобильных устройств — программное изделие, разновидность прикладного [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предназначенная для работы на смартфонах, [планшетах](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D1%88%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и других мобильных (портативных, переносных, карманных) устройствах. Обеспечивает без привязки к [стационарному компьютеру](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), «на ходу» необходимые пользователю взаимодействия со [Всемирной Сетью](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) (либо устанавливается на устройстве благодаря такому взаимодействию — после скачивания на носимое пользователем устройство).

Android Studio — Это IDE, созданная компанией Google для разработки приложений под операционную систему Android. [IDE](https://timeweb.com/ru/community/articles/5-luchshih-ide-1) – это набор инструментов для программиста, помогающих ему «написать» логику поведения программы, добавить в нее визуальные элементы, сделать их интерактивными и превратить задумку в полноценный продукт, который можно выставить на продажу в Google Play. Android Studio создана специально для взаимодействия с такими языками программирования, как Java и Kotlin.  Android Studio предлагает обширный набор функций и возможностей, которые упрощают и ускоряют процесс разработки. Включая мощный редактор кода с подсветкой синтаксиса и автодополнением, интегрированную систему сборки Gradle, эмулятор Android для запуска и тестирования приложений без физического устройства, инструменты для разработки пользовательского интерфейса и управления ресурсами, а также множество других инструментов для отладки и профилирования приложений.

RecyclerView — это мини-приложение Android, которое отображает коллекцию элементов в списке или сетке, позволяя пользователю прокручивать коллекцию.

RecyclerView предлагает две убедительные функции:

* Она имеет гибкую архитектуру, которая позволяет изменять его поведение путем подключения к предпочитаемым компонентам.
* Она эффективна с большими коллекциями, так как она повторно использует представления элементов и требует использования держателей представлений для кэширования ссылок на представления.

RecyclerView работает следующим образом (Рисунок 1): на экране устройства отображаются видимые элементы списка; при прокрутке списка верхний элемент уходит за пределы экрана и очищается, а после помещается вниз экрана и заполняется новыми данными.

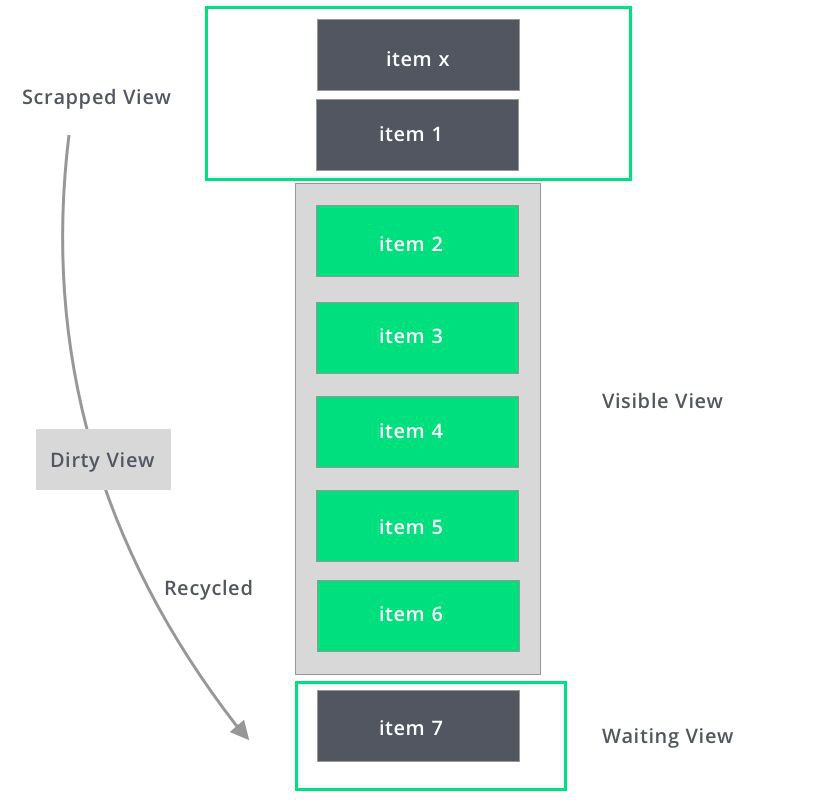


Рисунок 1. Работа RecyclerView

Для корректной работы RecyclerView необходимо реализовать следующие компоненты:

* RecyclerView, который необходимо добавить в макет нашего Activity;
* Adapter, который содержит, обрабатывает и связывает данные со списком;
* ViewHolder, который служит для оптимизации ресурсов и является своеобразным контейнером для всех элементов, входящих в список;
* ItemDecorator, который позволяет отрисовать весь декор;
* ItemAnimator, который отвечает за анимацию элементов при добавлении, редактировании и других операций;
* DiffUtil, который служит для оптимизации списка и добавления стандартных анимаций.

**ScrollView**и**HorizontalScrollView** — это две разметки, которые позволяют прокручивать содержимое в вертикальном и горизонтальном направлениях соответственно.

* Позволяет просматривать контент, который не вмещается на экране без прокрутки.
* Добавляет полосу прокрутки, когда содержимое превышает размер экрана устройства.
* Может содержать только один дочерний элемент или контейнер ViewGroup.

**ScrollView**и**HorizontalScrollView** достаточно просты в использовании и предназначены для небольших списков или блоков текста, которые не требуют многократного повторения.

ViewPager — один из самых известных и широко используемых компонентов Android Support Library. Все простейшие карусели, онбординги и слайдеры в [мобильных приложениях для Android](https://surf.ru/?utm_source=habr&utm_medium=organic&utm_campaign=viewpager2_novaya_funktsionalnost_v_staroy_obertke) выполнены именно на нем. В феврале 2019 года команда разработки AndroidX выпустила ViewPager2.  **Android ViewPager** позволяет пользователю свайпать (swipe) влево или вправо, чтобы посмотреть абсолютно новую страницу (экран).

APK ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Android Package Kit) — формат архивных исполняемых файлов-приложений для [Android](https://ru.wikipedia.org/wiki/Android) и ряда других операционных систем, основанных на [Android](https://ru.wikipedia.org/wiki/Android). Каждое приложение Android [скомпилировано](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [упаковано](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в один файл, который включает в себя весь код приложения ([.DEX](https://ru.wikipedia.org/w/index.php?title=.DEX&action=edit&redlink=1)-файлы), ресурсы, активы (assets), файл манифеста AndroidManifest.xml и нативные библиотеки (jniLibs). Файл приложения может иметь любое имя, но расширение должно быть .APK, например myAppFile.apk.

APK файл (Android Package) – это стандартный формат приложений для Android. По сути, APK файл — это сжатый архив, который включает ресурсы и код мобильного приложения. Разработчики приложений и игр загружают эти файлы в Google Play Маркет, а в дальнейшем, пользователи могут устанавливать эти приложения на свой смартфон и планшет.

## **Практическая часть**

**Техническое задание**

**1. Вводная информация**

GenshinChar – мобильное приложение-каталог по игре Genshin Impact, содержащее всю информацию о персонажах: стихия, лучшее оружие и артефакты, ресурсы для прокачки. Поможет игрокам получить всю необходимую информацию о том, как улучшить своих любимых героев.

**2. Функциональность**

Приложение должно обладать следующими возможностями:

- Выбор героя в меню персонажей

- Просмотр персонажа и информации о нём:

* Общая информация
* Материалы для повышения уровня
* Материалы для улучшения навыков
* Лучшее оружие
* Лучшие артефакты

**3. Целевая аудитория**

Игроки Genshin Impact - как новички, так и опытные, так как игра постоянно обновляется и выходят новые персонажи, оружия и сеты артефактов, - желающие быстро получить информацию о своём персонаже, научиться правильно и эффективно им играть.

**4. User story**

Я как игрок-новичок хочу узнать, какие персонажи мне могут понравиться, каких будет проще улучшить, чтобы принять решение о том, каких персонажей в игре я хочу получить, и начать идти к их получению. Также после получения персонажа мне потребуется информация о том, как эффективно его улучшить, чтобы легко проходить игру и наслаждаться сюжетом и битвами.

Я как опытный игрок хочу получить информацию об идеальном улучшении своих персонажей, чтобы применить её на своём аккаунте и получать большие цифры урона в испытаниях, с легкостью проходя весь контент игры и получая самый высокий рейтинг.

**5. Карта путешествий (Рисунок 1)**

Рисунок 1. Карта путешествий

**6. Технические требования**

- Возможность использования приложения на различных мобильных устройствах с платформой Android

- Соответствие дизайну, разработанному в Figma

**Дизайн приложения**

Дизайн мобильного приложения, разработанный в Figma (Рисунки 2-5):

Изображение выглядит как текст, мультфильм, фантастика, графический дизайн

Автоматически созданное описание

Рисунок 2. Дизайн стартового экрана

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3. Меню выбора элемента персонажа

Изображение выглядит как текст, снимок экрана, мультфильм

Автоматически созданное описание

Рисунок 4. Меню выбора персонажа

Изображение выглядит как текст, снимок экрана, Брошюра, Печать

Автоматически созданное описание

Рисунок 5. Страница персонажа

В разделах «Оружие» и «Артефакты» (Рисунок 5) находятся слайдеры, содержащие соответственно топ-3 лучших оружия и артефактов для выбранного персонажа.

Задано изменение дизайна кнопок при нажатии (Рисунок 6).

 Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, логотип, символ

Автоматически созданное описаниеИзображение выглядит как Шрифт, логотип, символ, дизайн

Автоматически созданное описание

Рисунок 6. Дизайн эффекта нажатия кнопки

**Разработка приложения**

Структура проекта выглядит следующим образом (Рисунок 6).

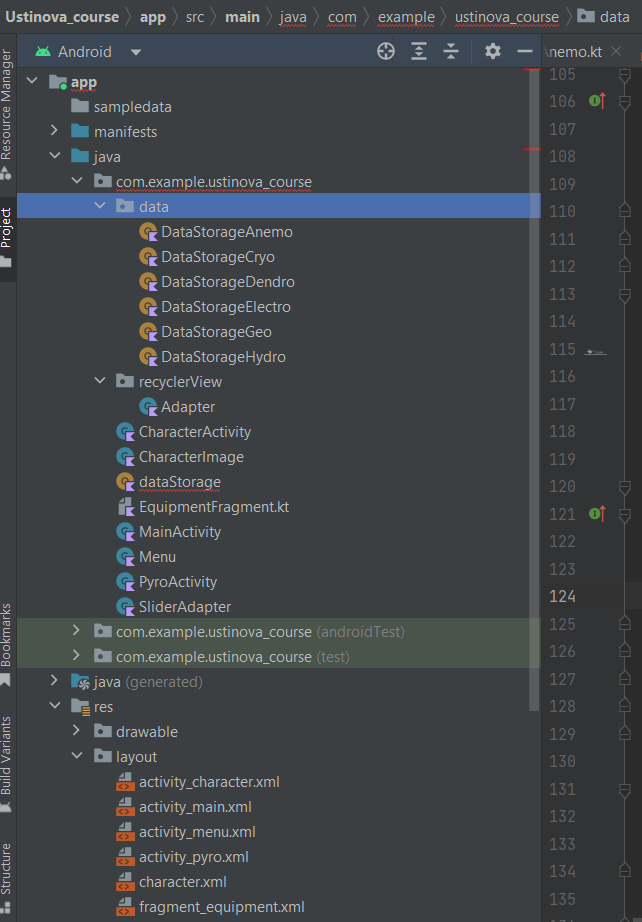


Рисунок 6. Структура проекта

В папке “java/com.examole.ustinova\_course” находятся все основные файлы приложения - активности, внутри в папке “data” содержатся хранилища данных с персонажами, а внутри папки recyclerAdapter – адаптер для RecyclerView. В папке “res/drawable” находятся все картинки, используемые в приложении, а также описания изменения стилей кнопок при нажатии. В папке “res/layout” содержатся все файлы разметок страниц приложения.

Разработка стартового экрана начинается с создания соответствующей активности MainActivity.kt и разметки страницы activity\_main.xml.

В ConstraitLayout добавим тестовое представление TextView для отображения названия приложения, ImageView для фонового прямоугольника, текст для отображения надписи «Добро пожаловать!» и кнопку-картинку ImageButton для перехода в меню.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7. Архитектура главного экрана

Для установки фонового изображения используем свойство android:background объекта ConstraitLayout, в котором находятся все компоненты (Скрипт 1). Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

Скрипт 1. Установка фонового изображения

Для перехода по кнопке этой кнопке нужно задать id, затем в Аctivity необходимо написать код перехода на следующий экран (Скрипт 2): находим кнопку по id, при нажатии кнопки создаём цель (intent) перехода: с текущего экрана нужно перейти в активность «Menu». Затем с помощью функции startActivity(intent) выполняется переход.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 2. Код для смены экрана по нажатию кнопки

Для изменения стиля кнопки при нажатии необходимо создать drawable-файл и описать стиль кнопки при нажатии, без нажатия, при наведении и в обычном режиме (Скрипт 3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 3. Смена стиля кнопки при взаимодействии

Этот файл необходимо задать источником изображения для кнопки (Скрипт 4).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 4. Установка фонового изображения

Главное меню создается файлами Menu.kt (активити) и activity\_menu.xml. Экран Меню содержит ImageView с картинкой заголовка красивым шрифтом и полоской, затем идёт ScrollView со всеми кнопками категорий персонажей, и внизу – кнопка перехода назад на предыдущий экран.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 8. Архитектура экрана главного меню

В активности для каждой кнопки пишем переход на другую страницу при их нажатии (Скрипт 5). Здесь мы передаём следующему экрану дополнительную переменную element с текстовым названием элемента (категории персонажа).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 5. Установка фонового изображения

При переходе из меню элементов на экран меню персонажей по элементу активность второго меню (PyroActivity.kt) получает информацию о необходимом элементе через переменную element и подгружает персонажей этого элемента из хранилища (для каждого элемента – своё хранилище персонажей; кроме того, устанавливается картинка соответствующего элемента в верхнюю часть экрана (Скрипт 6).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 6. Получение и применение информации, переданной с прошлого экрана

Меню персонажей по элементу создаётся с помощью файлов PyroActivity.kt и activity\_pyro.xml и состоит из картинки элемента в верху экрана, списка персонажей, реализованного с помощью RecyclerView, и кнопки «назад» (Рисунок 9).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 9. Архитектура меню персонажей по элементу

В коде RecyclerView нужно назначить менеджера, который отвечает за отображение элементов списка (Скрипт 7).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Скрипт 7. Назначение LayoutManager для RecyclerView

Также для описания отображения элементов нужно написать адаптер списка (в папке RecyclerView файл и EquipmentFragment.ktAdapter.kt) (Скрипт 8). Метод onСreateViewHolder передаёт обьекту ViewHolder каждый дочерний элемент списка для сохранения. В классе ViewHolder инициализируются элементы списка, связываются данные с помощью фукции bind. Метод onBindViewHolder извлекает соответствующие данные и использует для заполнения разметки. Также небходимо инициализировать onClickLictener для обработки нажатия на элементы списка.

class Adapter (Context: Context, private val items: List<CharacterImage>,)  
 :RecyclerView.Adapter<Adapter.ViewHolder> () {  
 private var onClickListener: OnClickListener?=null  
 private val inflater: LayoutInflater = LayoutInflater.from(Context)  
   
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 return ViewHolder(inflater.inflate(R.layout.*character*, parent, false))  
 }  
  
 override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val item = items[position]  
 holder.imagebtn1.setImageResource(getItem(position).ImageID1)  
 holder.imagebtn1.setOnClickListener **{** if (onClickListener != null) {  
 onClickListener!!.onClick(position, item )  
 }  
 **}** }  
 override fun getItemCount(): Int {  
 return items.size  
 }  
 interface OnClickListener {  
 fun onClick(position: Int, model: CharacterImage)  
 }  
 private fun getItem(position:Int) : CharacterImage = items[position]  
  
 fun setOnClickListener(onClickListener: OnClickListener) {  
 this.onClickListener = onClickListener  
 }  
 class ViewHolder (itemView: View) : RecyclerView.ViewHolder(itemView) {  
 val imagebtn1 : ImageButton = itemView.findViewById(R.id.*charbtn*)  
 }  
  
}

Скрипт 8. Адаптер списка

Дочерний элемент списка (описывается файлом разметки character.xml) выглядит как кнопка с картинкой (Рисунок 10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

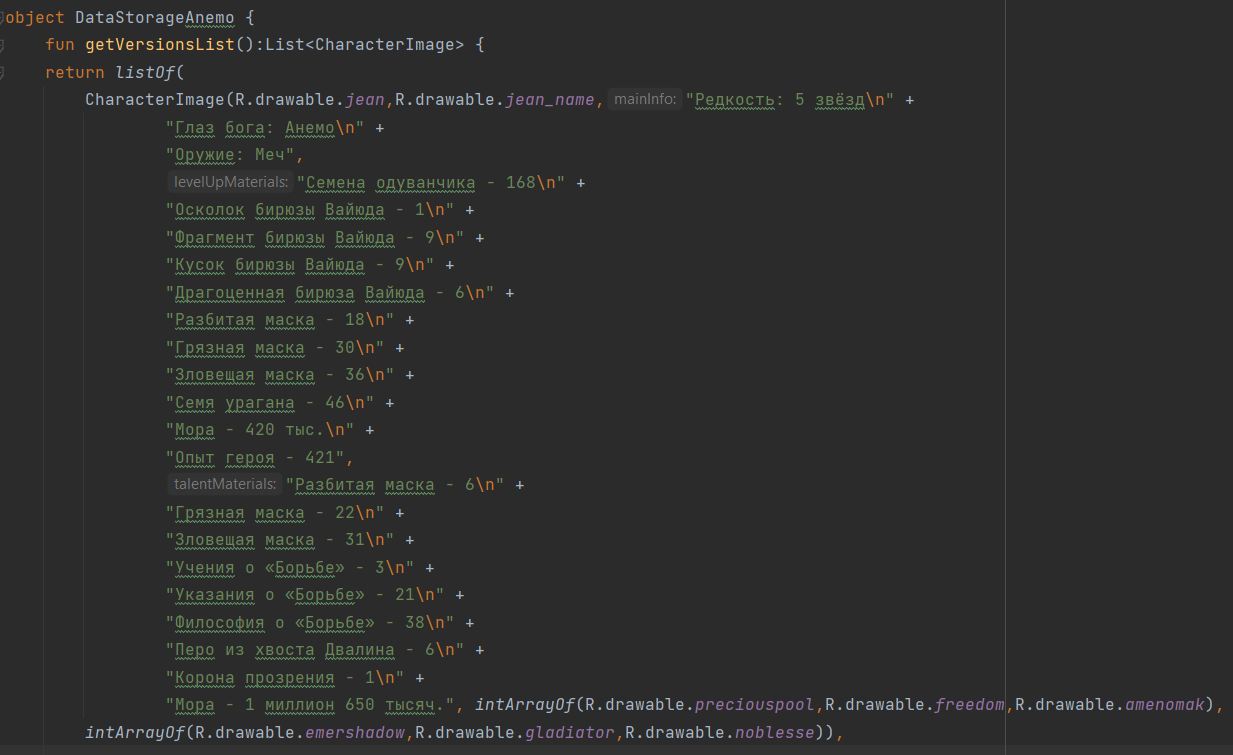
Рисунок 10. Архитектура дочернего элемента списка

Экземпляр класса элемента списка (класс CharacterImage.kt) содержит id картинки для кнопки, id картинки с красивым именем персонажа, тексты с основной информацией, материалами возвышения, материалами улучшения навыков, массивы картинок с оружием и артефактами (Скрипт 9a).

data class CharacterImage (  
 @DrawableRes val ImageID1 : Int,  
 @DrawableRes val nameImage:Int,  
 val mainInfo:String,  
 val levelUpMaterials: String,  
 val talentMaterials: String,  
 val imagesWeapon: IntArray,  
 val imagesArts: IntArray,  
): java.io.Serializable

Скрипт 9a. Описание класса элемента списка

Данные о персонажах хранятся в файлах DataStorage (Скрипт 9б).



Скрипт 9б. Хранимые данные элемента списка

В коде активити PyroActivity.kt находим RecyclerView и подключаем ему адаптер. Здесь же расписываем функцию onClick, выполняющуюся при нажатии на элемент: переход на страницу персонажа, при этом дополнительно передаётся вся информация о персонаже (Скрипт 10).

val list = findViewById<RecyclerView>(R.id.*charList*)

val adapter = Adapter(this, characters);  
list.*adapter* = adapter  
adapter.setOnClickListener(object :  
 Adapter.OnClickListener {  
 override fun onClick(position: Int, model: CharacterImage) {  
 val intent = Intent(this@PyroActivity, CharacterActivity::class.*java*)  
 intent.putExtra(NEXT\_SCREEN, model)  
 startActivity(intent)  
 }  
})

Скрипт 10. Подключение адаптера и функция OnClick

Страница персонажа (файлы CharacterActivity.kt и activity\_character.xml) содержит ImageView с картинкой имени в верху экрана, ScrollView со всей текстовой информацией о персонаже и двумя слайдерами – pager, кнопку «Назад» (Рисунок 11).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 11. Архитектура экрана с информацией о персонаже

В активности вся получаемая информация передаётся всем TextView, ImageView и слайдерам (Скрипт 11).

//получение данных передаваемого персонажа  
var emplist:CharacterImage?=null  
if(*intent*.hasExtra(PyroActivity.NEXT\_SCREEN)){  
 emplist =  
 *intent*.getSerializableExtra(PyroActivity.NEXT\_SCREEN) as CharacterImage?  
}  
//передача данных в компоненты экрана  
if(emplist!=null){  
 val textName = findViewById<ImageView>(R.id.*imageViewName*)  
 textName.setImageResource(emplist!!.nameImage)  
 val textMainInfo=findViewById<TextView>(R.id.*textViewMainInfo*)  
 textMainInfo.setText(emplist!!.mainInfo)  
 val textLevelMats = findViewById<TextView>(R.id.*textViewLevelUpMats*)  
 textLevelMats.setText(emplist!!.levelUpMaterials)  
 val textTalentMats=findViewById<TextView>(R.id.*textViewTalentsMats*)  
 textTalentMats.setText(emplist!!.talentMaterials)

Скрипт 11. Получение данных персонажа и заполнение списка

Для реализации слайдера необходимо создать фрагмент слайдера-картинку (файлы fragment\_eqipment.xml и EquipmentFragment.kt) (Рисунок 12):

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 12. Фрагмент слайдера

Адаптер для слайдера (SliderAdapter.kt) (Скрипт 12).

class SliderAdapter(fragment: FragmentActivity, private val images1: IntArray ) : FragmentStateAdapter(fragment) {  
 override fun getItemCount(): Int = 3 //устанавливаем макисмальное число картинок = 3  
 override fun createFragment(position: Int): Fragment {  
 val fragment = EquipmentFragment()  
 fragment.*arguments* = Bundle().*apply* **{** putInt(*ARG\_IMAGE1*, images1[position])//передаём фрагменту картинку  
 **}** return fragment  
 }  
}

Скрипт 12. Адаптер слайдера

В активности фрагмента устанавливаем полученную картинку в ImageView (Скрипт 13).

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 *arguments*?.*takeIf* **{ it**.containsKey(*ARG\_IMAGE1*)**}**?.*apply* **{** val imageView: ImageView = view.findViewById(R.id.*imageView*)  
 val name: Int = getInt(*ARG\_IMAGE1*)  
 imageView.setImageResource(name)  
 **}**}

Скрипт 13. Изменение картинки в слайдере

Приложение готово. С помощью запуска приложения на эмуляторе можно проверить все функции (Рисунки 13-20).

Изображение выглядит как текст, мультфильм, снимок экрана, фантастика

Автоматически созданное описание

Рисунок 13. Запуск приложения: стартовый экран

Изображение выглядит как текст, мультфильм, снимок экрана

Автоматически созданное описание

Рисунок 14. Запуск приложения: нажатие на кнопку

Изображение выглядит как текст, снимок экрана, Мобильный телефон, Мобильное устройство

Автоматически созданное описание

Рисунок 15. Запуск приложения: меню элементов

Изображение выглядит как текст, Мобильный телефон, снимок экрана, мультфильм

Автоматически созданное описание

Рисунок 16. Запуск приложения: меню персонажей

Изображение выглядит как текст, Мобильный телефон, гаджет, Мобильное устройство

Автоматически созданное описание

Рисунок 17. Запуск приложения: пролистывание меню и нажатие на элемент

Изображение выглядит как текст, Устройство связи, гаджет, снимок экрана

Автоматически созданное описание

Рисунок 18. Запуск приложения: страница персонажа

Изображение выглядит как текст, снимок экрана, Мобильный телефон, Мобильное устройство

Автоматически созданное описание

Рисунок 19. Запуск приложения: пролистывание страницы персонажа

Изображение выглядит как текст, снимок экрана, мультимедиа, Устройство связи

Автоматически созданное описание

Рисунок 20. Запуск приложения: работа слайдера

Сформировать APK-файл приложения можно в разделе сборки (Рисунки 21-22).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 21. Формирование APK-файла

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 22. APK-файл в директории

Открываем файл на телефоне, устанавливаем приложение и получаем возможность им пользоваться:

Изображение выглядит как текст, фантастика, мультфильм, графический дизайн

Автоматически созданное описание

Рисунок 22. Скриншот главного экрана

Изображение выглядит как текст, снимок экрана, Реклама в Интернете, Веб-сайт

Автоматически созданное описание

Рисунок 23. Скриншот страницы персонажа

## **Заключение**

Разработка мобильного приложения – сложный процесс, состоящий из нескольких этапов: разработки технического задания, оформления дизайна приложения, непосредственного программирования в специальном ПО и экспорта приложения. Программирование приложения включает в себя реализацию разметки экранов и связанных с ними классов, а также дополнительных элементов.

В ходе работы были реализованы все этапы создания мобильного приложения. Был создан каталог по игре, предоставляющий информацию в удобном виде с использованием новейших технологий программирования в среде Android Studio.

## **Список литературы**

1. Figma — простое решение для дизайнера, сложное решение для верстальщика // Хабр URL: https://habr.com/ru/articles/463181/ (дата обращения: 15.05.2023).
2. RecyclerView // Microsoft URL: https://learn.microsoft.com/ru-ru/xamarin/android/user-interface/layouts/recycler-view/ (дата обращения: 15.05.2023).
3. ScrollView // Android Developers URL: https://developer.android.com/reference/android/widget/ScrollView (дата обращения: 15.05.2023).
4. ViewPager2 // Android Developers URL: https://developer.android.com/reference/androidx/viewpager2/widget/ViewPager2 (дата обращения: 15.05.2023).
5. Как пользоваться Android Studio // Timeweb Community URL: https://timeweb.com/ru/community/articles/kak-polzovatsya-android-studio (дата обращения: 15.05.2023).
6. Краткий обзор Layout XML в Android. Часть 2 // Хабр URL: https://habr.com/ru/articles/720134/ (дата обращения: 15.05.2023).
7. Первые шаги в дизайне: инструкция по базовым возможностям Figma // Яндекс Практикум URL: https://practicum.yandex.ru/blog/chto-takoe-figma-dlya-dizainera/#chto-takoe-figma (дата обращения: 15.05.2023).
8. Создание apk-файла в Android Studio // Java-online.ru URL: https://java-online.ru/android-apk.xhtml (дата обращения: 15.05.2023).