|  |
| --- |
|  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

|  |
| --- |
|  |

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**

**“Стартовый проект на Android”**

по дисциплине

**«Разработка мобильных приложений»**

|  |  |
| --- | --- |
| Выполнил студент группы ИНБО-03-22 | Алтухов А.В. |
|  |  |
| Принял ассистент кафедры МОСИТ | Шешуков Л.С. |

|  |  |  |
| --- | --- | --- |
| Практическая  работа выполнена | «\_\_» \_\_\_\_\_\_\_\_\_2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
| «Зачтено» | «\_\_» \_\_\_\_\_\_\_\_\_2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |

Москва 2024

СОДЕРЖАНИЕ

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc159059665)

[2. УСТАНОВКА ПРИЛОЖЕНИЯ И СОЗДАНИЕ ПРОЕКТА 4](#_Toc159059666)

[3. НАПОЛНЕНИЕ ПРОЕКТА 8](#_Toc159059667)

[4. ДОБАВЛЕНИЕ TOAST-ОВ И ЖИЗНЕННЫЙ ЦИКЛ ПРИЛОЖЕНИЯ 10](#_Toc159059668)

[5. ДОБАВЛЕНИЕ LOGCAT 17](#_Toc159059669)

[6. ВЫВОДЫ 19](#_Toc159059670)

[ПРИЛОЖЕНИЕ 1 20](#_Toc159059671)

# ПОСТАНОВКА ЗАДАЧИ

В ходе выполнения данной практической работы необходимо выполнить следующий перечень задач:

* Создать проект;
* Описать в отчете наполнение проекта;
* На всех этапах жизненного цикла MainActivity отобразить Toast, а также вывести в Logcat строки логов при помощи методов Log.d, Log.e и т. д.

# УСТАНОВКА ПРИЛОЖЕНИЯ И СОЗДАНИЕ ПРОЕКТА

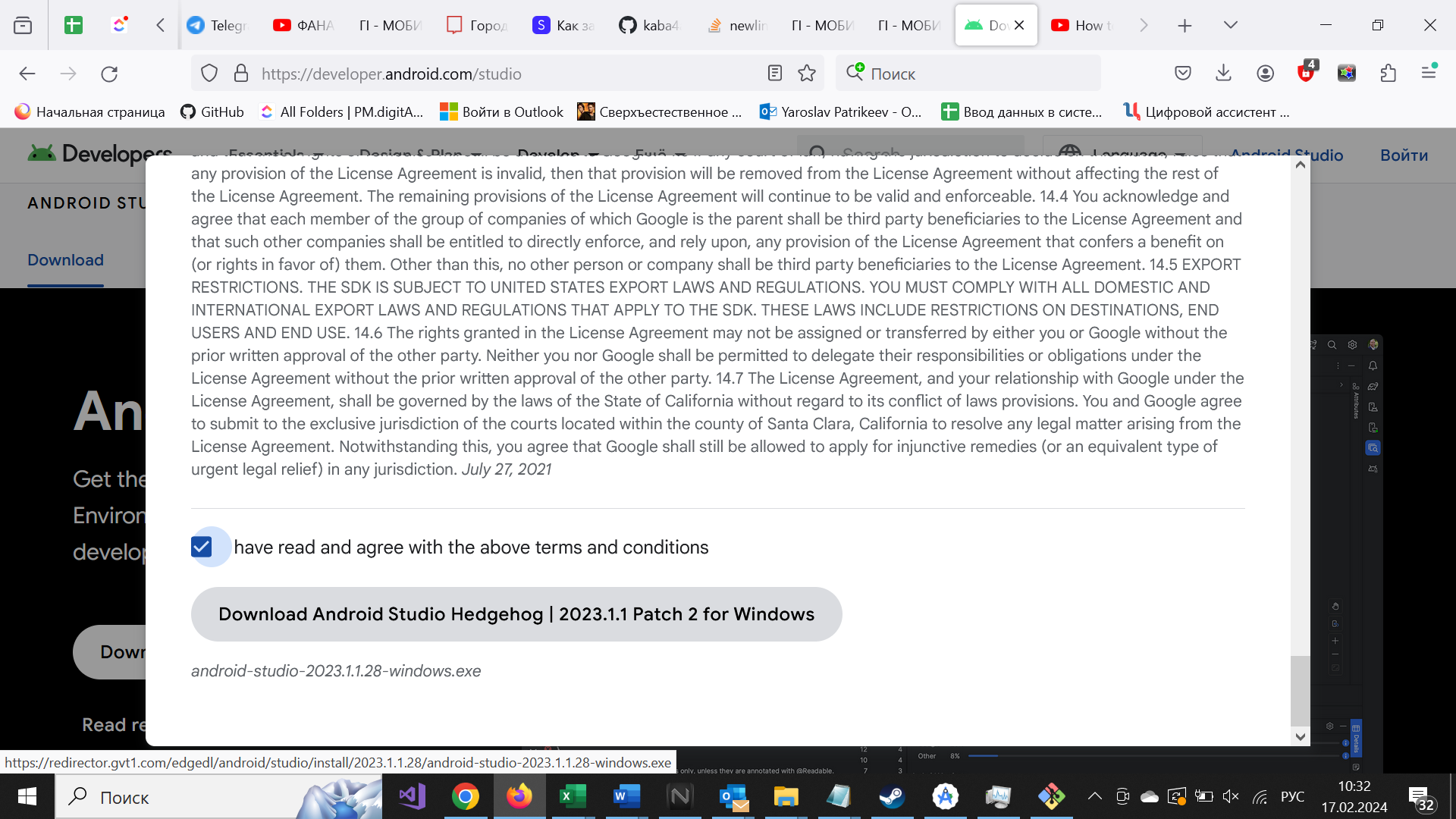
Приложения для Android могут быть написаны с использованием языков Kotlin, Java и C++. Инструменты Android SDK компилируют наш код вместе с любыми файлами данных и ресурсов в APK или пакет приложений для Android. В моём случае, я предпочту язык Java.

Среди доступных разработчикам Android-приложений IDE можно назвать несколько:

* **Android Studio** – с 2016 года единственная официальная IDE для разработки под Android от компании Google (разработчика ОС Android).
* **IntelliJ IDEA** – Android Studio основан на технологии IntelliJ.
* **NetBeans** – ещё одна свободно распространяемая IDE, которая может быть использована для разработки мобильных приложений.
* **Eclipse** – до 2016 года на протяжении многих лет была самой популярной IDE для разработки под Android, множество учебных материалов в Сети сделаны на ней. Для полноценной работы требует установки специального плагина ADT (Android Development Tools), который с 2016 года самой компанией Google не поддерживается.

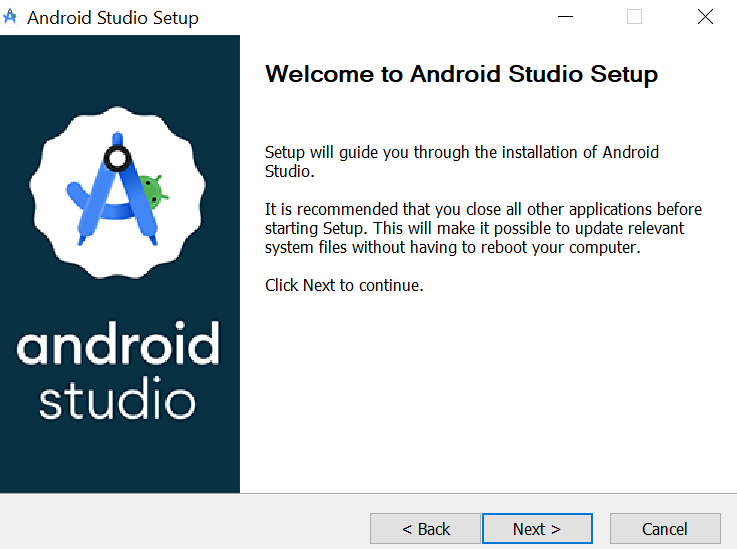
Для выполнения заданий этого практикума выбрана официальная среда разработки Android Studio.

С помощью ссылки в “методичке” необходимо скачать установщик с официального сайта. Подтвердим условия пользовательского соглашения и скачаем установщик (Рисунок 2.1).

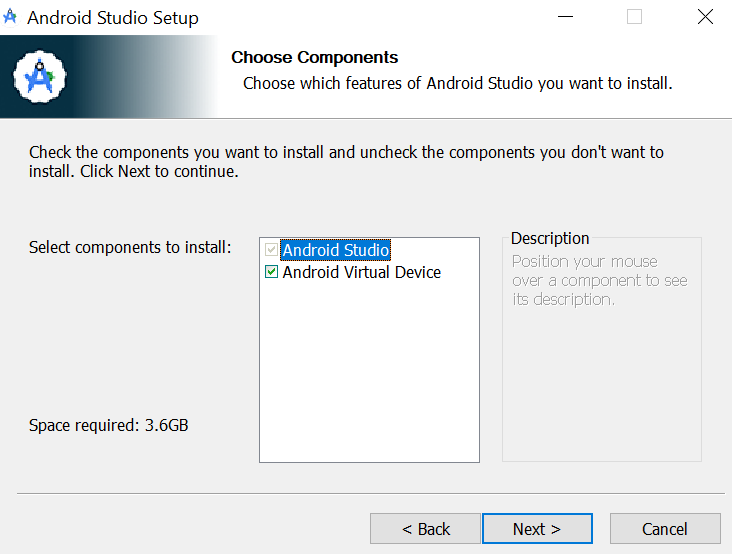


**Рисунок 2.1 - Пользовательское соглашение и загрузка файла**

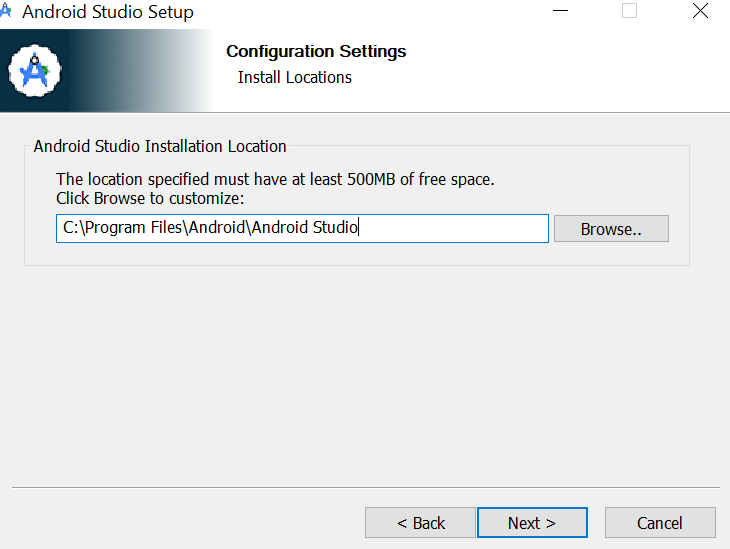
Далее перейдем к установке приложения через установщик. Поздороваемся с инсталлятором (Рисунок 2.2), выберем установку нужных рекомендованных компонентов (Рисунок 2.3), выберем путь к папке с хранилищем будущих проектов (Рисунок 2.4), ничего не будем трогать в окне с ярлыками, оставив значение по умолчанию (Рисунок 2.5) и нажмем кнопку установки (Рисунок 2.6).



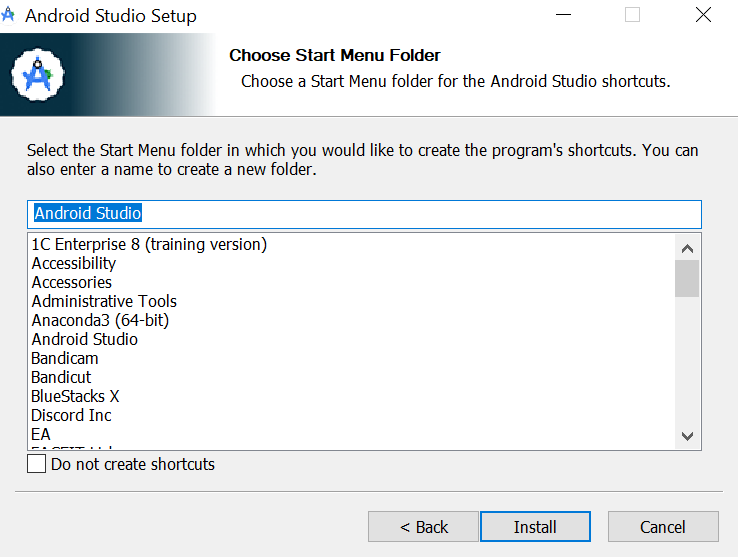
**Рисунок 2.2 – Приветствие установщика**



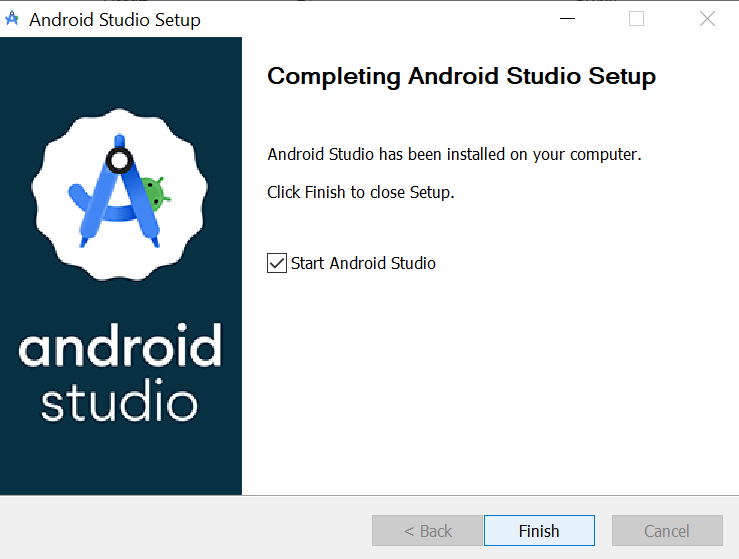
**Рисунок 2.3 – Установка необходимых компонентов**



**Рисунок 2.4 – Выбор пути к папке**



**Рисунок 2.5 – Выбор ярлыка**



**Рисунок 2.6 – Финальный этап установки**

Для создания нового проекта выберем в стартовом окне Android Studio, в разделе по умолчанию “Projects” команду “New Project”, далее нас встречает окно с готовыми шаблонами Activity (Рисунок 2.7).

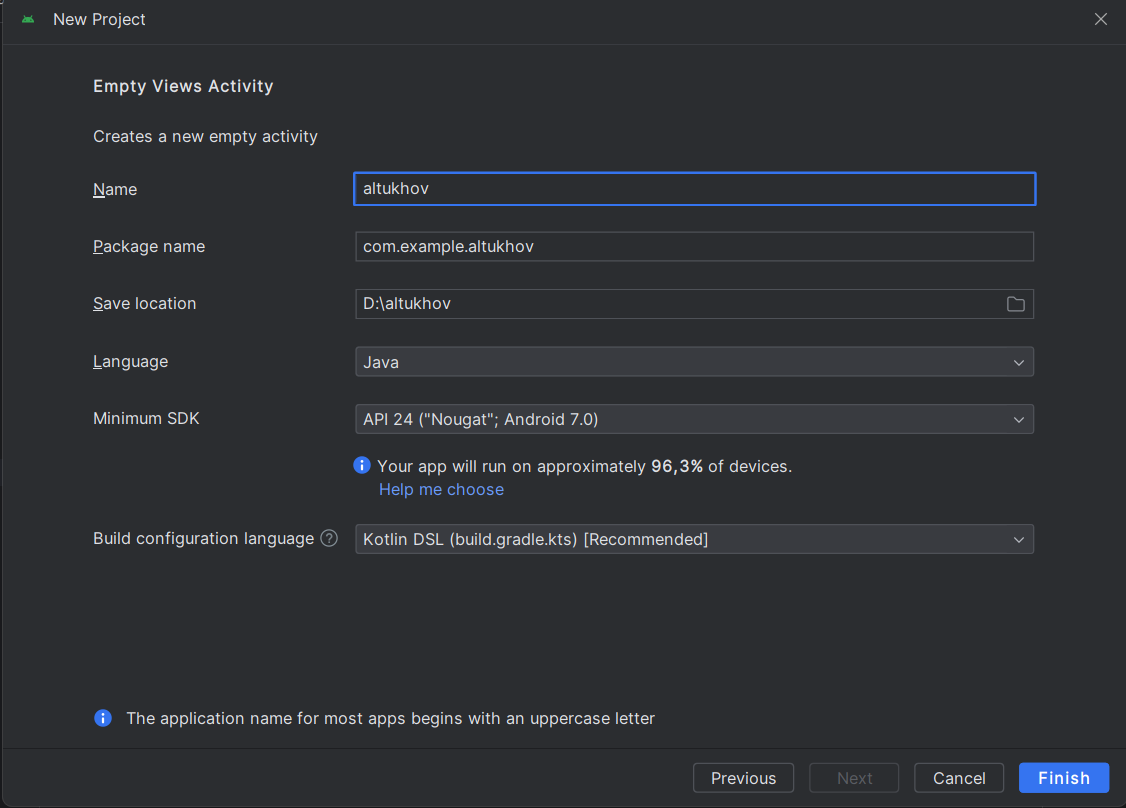
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Рисунок 2.7 — Окно “New Project”**

Выберем шаблон «Empty Views Activity», в результате будет создано приложение с одним Аctivity этого типа.

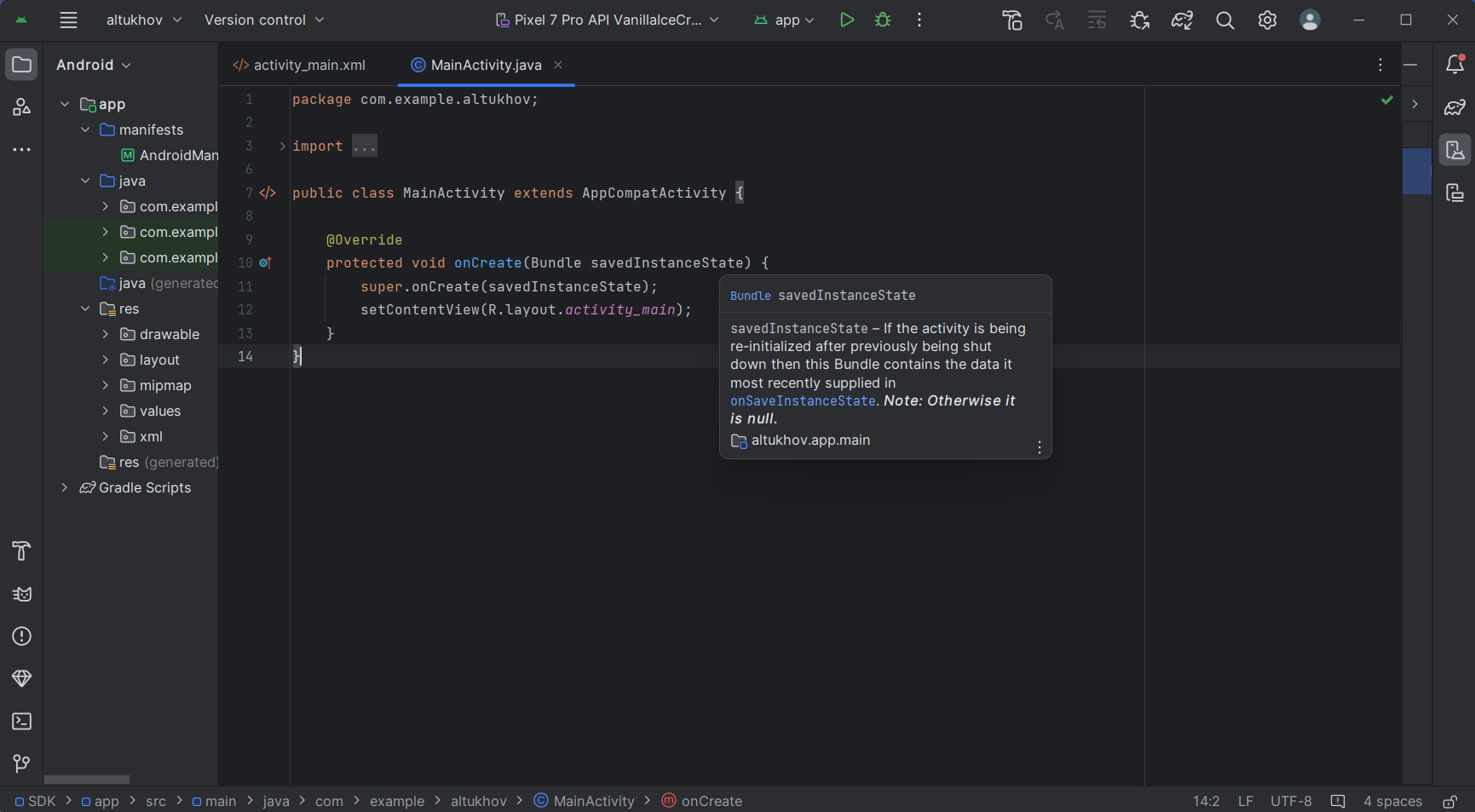
Далее мы переходим к окну заполнения сведений о нашем проекте (Рисунок 2.8). Укажем название – altukhov, имя пакета и место для сохранения оставим по умолчанию. Нам предоставляется выбор языка программирования: Java или Kotlin. В рамках данного курса изучается разработка мобильных приложений на Java, следовательно делаем выбор в пользу этого языка программирования. Также необходимо выбрать минимальную версию операционной системы, которая будет поддерживать наше приложение. Я выбрал API 24 (“Nougat”; Android 7.0), так как приложение сможет покрыть наибольшее количество устройств.



**Рисунок 2.8 — Конечная настройка проекта**

Далее нажимаем кнопку “Finish”. Дожидаемся, пока Android Studio создаст новый проект с указанными характеристиками и загрузит необходимые компоненты.

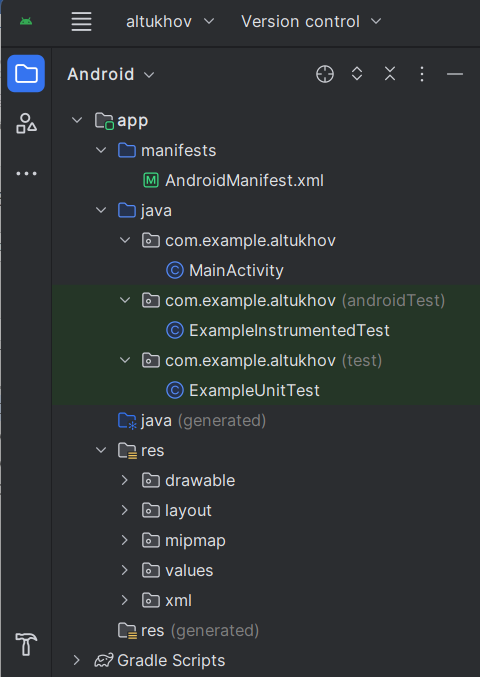
Вид окна Android Studio с открытым в нём новым проектом (Рисунок 2.9).



**Рисунок 2.9 — Проект в окне Android Studio**

# 3. НАПОЛНЕНИЕ ПРОЕКТА

После создания проекта, внизу нас встречает панель выполнения и строка состояния. В правой части рабочей области в редакторе открывается файл MainActivity.java. Слева расположена структура проекта, разберемся представляют из себя основные из них (Рисунок 3.1).



**Рисунок 3.1 — Файлы в Android Studio**

Файл **AndroidManifest.xml** является одним из самых важных в проекте Android приложения. В нем содержится информация о пакетах приложения, компонентах, таких как Activity, Service и других.

Файл **AndroidManifest.xml** выполняет следующие задачи:

* Описывает разрешения, необходимые приложению для работы, например на доступ к камере или внутреннему хранилищу;
* Определяет, как будет запускаться приложение, например, какое Activity должно быть запущено при нажатии на иконку в списке приложений (какие фильтры использовать).

Папка **java** содержит исходный код приложения. Классы могут быть расположены в различных пакетах, но обязательно внутри папки java.

В папке **res** расположены все используемые приложением ресурсы, включая изображения, различные xml файлы, анимации, звуковые файлы и многие другие. Внутри папки res эти все ресурсы распределены по своим папкам:

* Папка **drawable** содержит файлы с изображениями, которые будет использоваться в приложении;
* Папка **layout** располагает xml файлами, которые используются для построения пользовательского интерфейса Android приложения;
* В папке **menu** находятся xml файлы, используемые только для создания меню;
* В **mipmap** папке хранят только значки приложения. Любые другие drawable элементы должны быть размещены в своей папке;
* **values** хранит те xml файлы, в которых определяются простые значения типа строк, массивов, целых чисел, размерностей, цветов и стилей.

Скрипты **Gradle** используются для автоматизации сборки проекта. Android Studio выполняет сборку приложения в фоновом режиме без какого-либо вмешательства со стороны разработчика.

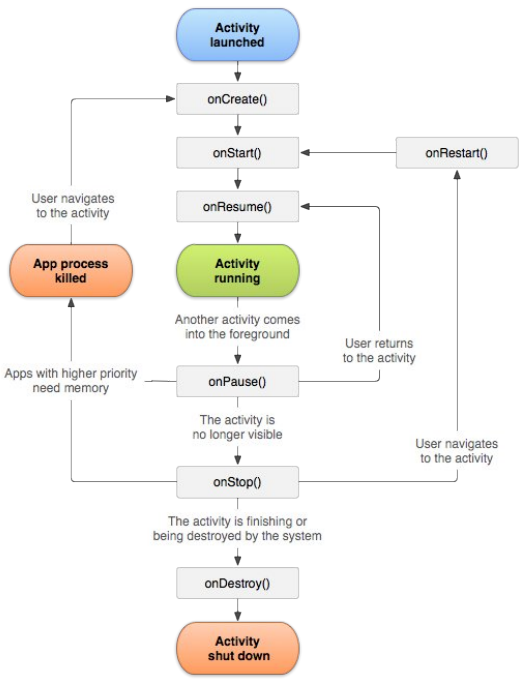
1. **ДОБАВЛЕНИЕ TOAST-ОВ И ЖИЗНЕННЫЙ ЦИКЛ ПРИЛОЖЕНИЯ**

По условию задания, необходимо добавить в наше приложение всплывающие окна **Toast** для каждого пункта жизненного цикла приложения.

Жизненный цикл приложения состоит из нескольких этапов, методы для обратного вызова которых представлены ниже:

* **onCreate():** вызывается при создании активности. Здесь происходит инициализация основных компонентов.
* **onStart():** вызывается, когда активность становится видимой пользователю.
* **onResume():** вызывается, когда активность начинает взаимодействие с пользователем. Она становится на передний план и получает фокус.
* **onPause():** вызывается, когда активность теряет фокус, но остаётся видимой пользователю (например, при открытии диалогового окна).
* **onStop():** вызывается, когда активность больше не видима для пользователя.
* **onDestroy():** вызывается перед уничтожением активности. Здесь освобождаются ресурсы.

Жизненный цикл активности мобильного приложения (рисунок 4.1).



**Рисунок 4.1 — Иллюстрация жизненного цикла активности**

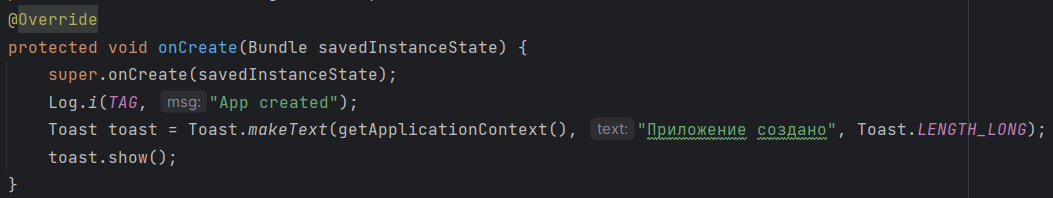
Класс **Toast** предоставляет простую обратную связь об операции в небольшом всплывающем окне. Он заполняет только пространство, необходимое для сообщения, а текущая активность остается видимой и интерактивной. Toast’ы автоматически исчезают по истечении тайм-аута.

Чтобы создать объект класса Toast, необходимо использовать метод makeText(), который принимает следующие параметры:

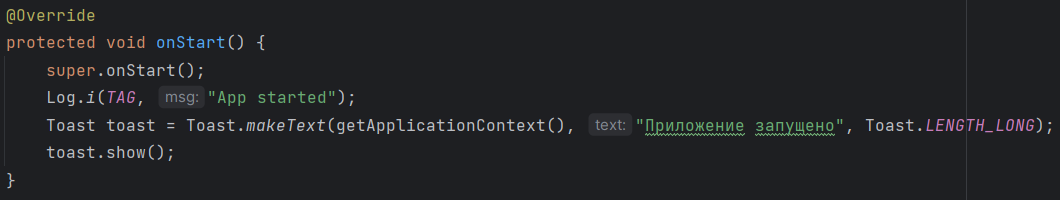
1. Контекст приложения.
2. Текст, который должен отображаться для пользователя.
3. Время, в течение которого всплывающее уведомление должно оставаться на экране.

Метод makeText() возвращает правильно инициализированный Toast объект.

Для отображения всплывающего уведомления, необходимо вызвать метод show(). Создадим и выведем объекты класса Toast в каждом этапе жизненного цикла приложения (Рисунок 4.2 – 4.7).



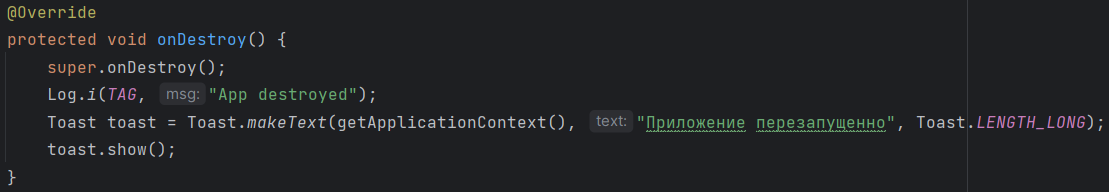
**Рисунок 4.2 — Создание объекта Toast и его отображение**



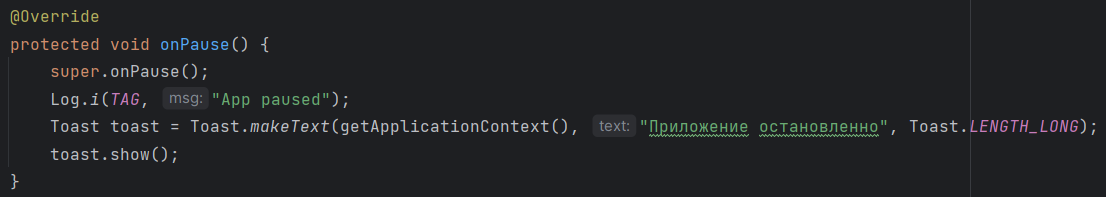
**Рисунок 4.3 — Создание объекта Toast и его отображение**



**Рисунок 4.4 — Создание объекта Toast и его отображение**



**Рисунок 4.5 — Создание объекта Toast и его отображение**

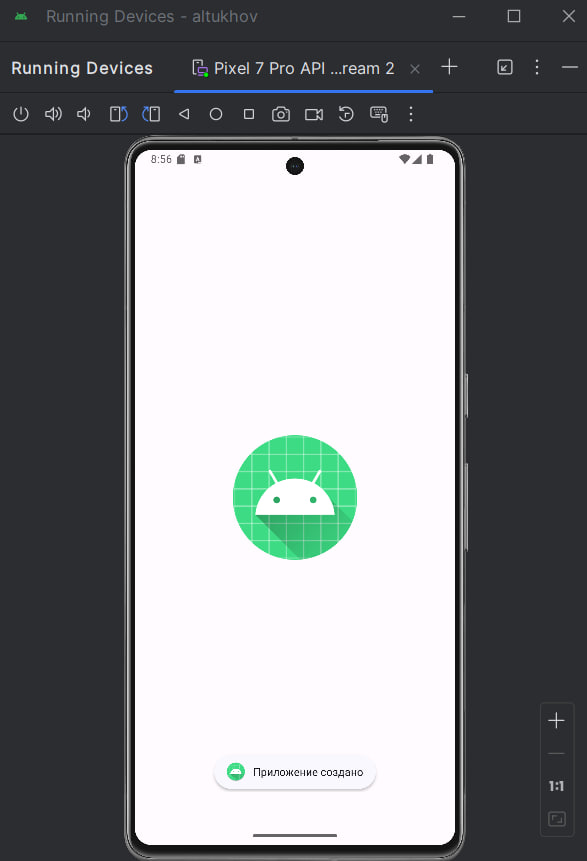


**Рисунок 4.6 — Создание объекта Toast и его отображение**

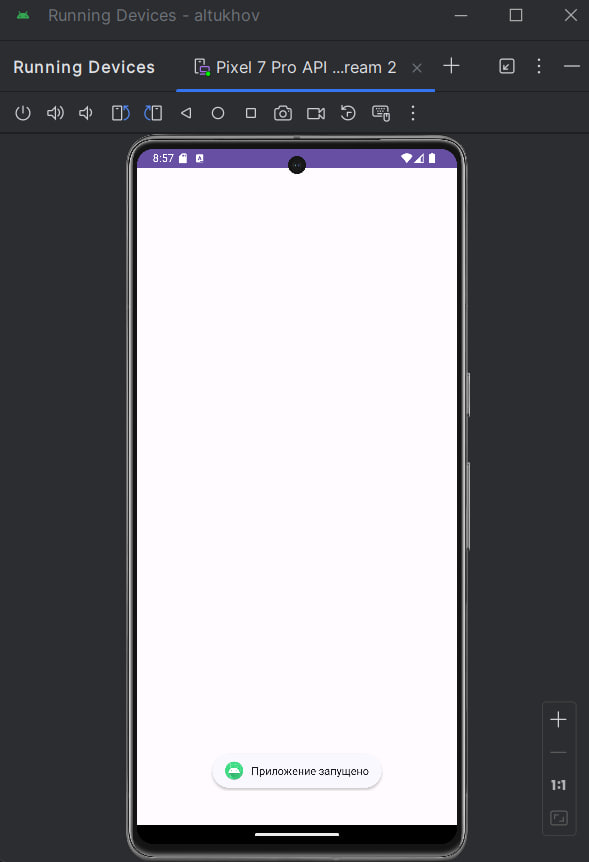


**Рисунок 4.7 — Создание объекта Toast и его отображение**

Теперь посмотрим как это будет выглядеть в самом приложении (Рисунок 4.8 – 4.12).



**Рисунок 4.8 — Всплывающие уведомление (создано) в приложении**



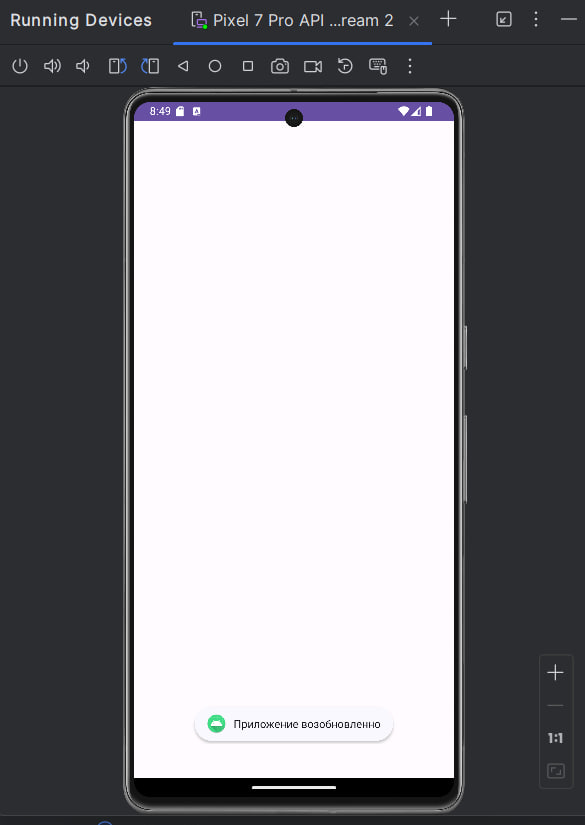
**Рисунок 4.9 — Всплывающие уведомление (запущено) в приложении**



**Рисунок 4.10 — Всплывающие уведомление (приостановлено) в приложении**



**Рисунок 4.11 — Всплывающие уведомление (остановлено) в приложении**



**Рисунок 4.12 — Всплывающие уведомление (возобновлено) в приложении**

# ДОБАВЛЕНИЕ LOGCAT

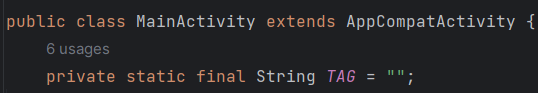
Окно **Logcat** в Android Studio помогает отлаживать наше приложение, отображая журналы с нашего устройства в режиме реального времени — например, сообщения, которые вы добавили в свое приложение с помощью класса, сообщения от служб, работающих на Android, или системные сообщения, например, при сборке мусора.

Когда приложение выдает исключение, Logcat показывает сообщение, за которым следует соответствующая трассировка стека, содержащая ссылки на строку кода.

Класс android.util.Log позволяет разбивать сообщения по категориям в зависимости от важности. Для разбивки по категориям используются специальные методы, которые легко запомнить по первым буквам, указывающие на категорию:

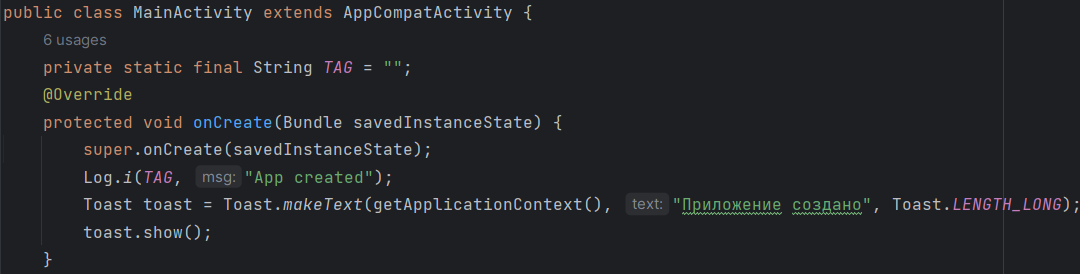
* **Log.e()** - ошибки (error);
* **Log.w()** - предупреждения (warning);
* **Log.i()** - информация (info);
* **Log.d()** - отладка (degub);
* **Log.v()** - подробности (verbose).

В первом параметре метода используется строка, называемая тегом. Обычно принято объявлять глобальную статическую строковую переменную TAG в начале кода (Рисунок 5.1).



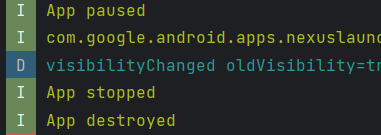
**Рисунок 5.1 — Объявленная переменная “TAG”**

Далее в любом месте программы вызываем нужный метод журналирования с этим тегом (Рисунок 5.2).



**Рисунок 5.2 — Пример использования логирования**

Пользователи не увидят этот журнал. Но, мы, как разработчик, можем увидеть его через программу LogCat, доступный в Android Studio (Рисунок 5.3).



**Рисунок 5.3 — Вывод логов через программу LogCat**

# 6. ВЫВОДЫ

В ходе выполнения данной практической работы был создан первый проект в Android Studio, настроенный согласно методическим рекомендациям. Были изучены основные файлы проекта. Освоена работа с классами Toast и Log. А также проведен процесс запуска и отладки приложения с помощью загруженного виртуального устройства.

# ПРИЛОЖЕНИЕ 1

Ссылка на проект <https://github.com/MLPeekk/rep_1>