**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное

учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Отчет по лабораторной работе №8 по дисциплине

**«Язык Kotlin и основы разработки»**

Сохранение состояния. ViewModel. LiveData

****

Выполнил:

Студент группы 1А22 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.К. Кравцов

Проверил:

Ст. преп. ОИТ ИШИТР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Дорофеев

Томск 2025

# Задание

Разработайте приложение для конвертации чего-нибудь во что-нибудь, из одной величины в несколько других величин. Главная активность приложения должна выглядеть примерно следующим образом:



Рисунок 1 – Пример интерфейса

При редактировании значения в одном из полей – в других полях должно сразу же появляться новое преобразованное значение.

При изменении конфигурации устройства (смене ориентации, темы оформления со светлой на тёмную и т.д.) значения во всех полях должны сохраняться.

Поля для сохранения данных должны быть либо в объекте ViewModel, либо в синглтоне, на выбор. Там же должны быть методы и константы для преобразования значений из одного в другое.

# Ход работы

1. Создан проект Lab8 на основе Empty Views Activity
2. Подготовлены ресурсы:

* Строковые ресурсы в strings.xml

<resources>  
 <string name="app\_name">Lab8</string>  
 <string name="converter\_title">Конвертер длины</string>  
 <string name="micrometers">В микрометрах:</string>  
 <string name="mils">В милах:</string>  
 <string name="millimeters">В миллиметрах:</string>  
 <string name="lines">В линиях:</string>  
 <string name="inches">В дюймах:</string>  
</resources>

* Ресурсы размеров в dimens.xml

<resources>  
 <dimen name="medium\_margin">16dp</dimen>  
 <dimen name="large\_margin">24dp</dimen>  
 <dimen name="medium\_text">18sp</dimen>  
 <dimen name="large\_text">24sp</dimen>  
 <dimen name="small\_weight">2</dimen>  
 <dimen name="big\_weight">3</dimen>  
</resources>

1. Реализована разметка с использованием LinearLayout:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:layout\_marginHorizontal="@dimen/large\_margin"  
 android:gravity="center">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/converter\_title"  
 android:textSize="@dimen/large\_text"  
 android:textStyle="bold"  
 android:layout\_marginBottom="@dimen/large\_margin" />  
  
 <!-- Микрометры -->  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginBottom="@dimen/medium\_margin">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/small\_weight"  
 android:text="@string/micrometers"  
 android:textSize="@dimen/medium\_text"  
 android:gravity="start"  
 android:labelFor="@+id/etMicrometers" />  
  
 <EditText  
 android:id="@+id/etMicrometers"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/big\_weight"  
 android:importantForAutofill="no"  
 android:inputType="numberDecimal"  
 android:textSize="@dimen/medium\_text" />  
 </LinearLayout>  
  
 <!-- Милы -->  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginBottom="@dimen/medium\_margin">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/small\_weight"  
 android:text="@string/mils"  
 android:textSize="@dimen/medium\_text"  
 android:gravity="start"  
 android:labelFor="@+id/etMils" />  
  
 <EditText  
 android:id="@+id/etMils"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/big\_weight"  
 android:importantForAutofill="no"  
 android:inputType="numberDecimal"  
 android:textSize="@dimen/medium\_text" />  
 </LinearLayout>  
  
 <!-- Миллиметры -->  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginBottom="@dimen/medium\_margin">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/small\_weight"  
 android:text="@string/millimeters"  
 android:textSize="@dimen/medium\_text"  
 android:gravity="start"  
 android:labelFor="@+id/etMillimeters" />  
  
 <EditText  
 android:id="@+id/etMillimeters"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/big\_weight"  
 android:importantForAutofill="no"  
 android:inputType="numberDecimal"  
 android:textSize="@dimen/medium\_text" />  
 </LinearLayout>  
  
 <!-- Линии -->  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginBottom="@dimen/medium\_margin">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/small\_weight"  
 android:text="@string/lines"  
 android:textSize="@dimen/medium\_text"  
 android:gravity="start"  
 android:labelFor="@+id/etLines" />  
  
 <EditText  
 android:id="@+id/etLines"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/big\_weight"  
 android:importantForAutofill="no"  
 android:inputType="numberDecimal"  
 android:textSize="@dimen/medium\_text" />  
 </LinearLayout>  
  
 <!-- Дюймы -->  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginBottom="@dimen/medium\_margin">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/small\_weight"  
 android:text="@string/inches"  
 android:textSize="@dimen/medium\_text"  
 android:gravity="start"  
 android:labelFor="@+id/etInches" />  
  
 <EditText  
 android:id="@+id/etInches"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="@dimen/big\_weight"  
 android:importantForAutofill="no"  
 android:inputType="numberDecimal"  
 android:textSize="@dimen/medium\_text" />  
 </LinearLayout>  
  
</LinearLayout>

1. Реализована логика конвертации:

* Создан класс ConversionViewModel, наследник ViewModel
* Использованы LiveData для хранения значений каждого поля
* Реализованы методы конвертации между всеми единицами измерения
* Добавлен флаг isUpdating для предотвращения бесконечных циклов обновления

package ru.olegkravtsov.lab8  
  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.ViewModel  
  
class ConversionViewModel : ViewModel() {  
 private val \_micrometers = MutableLiveData<String>()  
 val micrometers: LiveData<String> get() = \_micrometers  
  
 private val \_mils = MutableLiveData<String>()  
 val mils: LiveData<String> get() = \_mils  
  
 private val \_millimeters = MutableLiveData<String>()  
 val millimeters: LiveData<String> get() = \_millimeters  
  
 private val \_lines = MutableLiveData<String>()  
 val lines: LiveData<String> get() = \_lines  
  
 private val \_inches = MutableLiveData<String>()  
 val inches: LiveData<String> get() = \_inches  
  
 private var isUpdating = false  
  
 // Конвертация из микрометров  
 fun updateMicrometers(value: String) {  
 if (!isUpdating) {  
 isUpdating = true  
 \_micrometers.value = value  
 convertFromMicrometers(value)  
 isUpdating = false  
 }  
 }  
  
 private fun convertFromMicrometers(micrometerValue: String) {  
 val micrometers = micrometerValue.toDoubleOrNull() ?: return  
  
 val mils = micrometers / 25.4  
 val millimeters = micrometers / 1000.0  
 val lines = micrometers / 2540.0  
 val inches = micrometers / 25400.0  
  
 \_mils.value = mils.toString()  
 \_millimeters.value = millimeters.toString()  
 \_lines.value = lines.toString()  
 \_inches.value = inches.toString()  
 }  
  
 // Конвертация из милов  
 fun updateMils(value: String) {  
 if (!isUpdating) {  
 isUpdating = true  
 \_mils.value = value  
 convertFromMils(value)  
 isUpdating = false  
 }  
 }  
  
 private fun convertFromMils(milValue: String) {  
 val mils = milValue.toDoubleOrNull() ?: return  
  
 val micrometers = mils \* 25.4  
 val millimeters = mils \* 0.0254  
 val lines = mils / 100.0  
 val inches = mils / 1000.0  
  
 \_micrometers.value = micrometers.toString()  
 \_millimeters.value = millimeters.toString()  
 \_lines.value = lines.toString()  
 \_inches.value = inches.toString()  
 }  
  
 // Конвертация из миллиметров  
 fun updateMillimeters(value: String) {  
 if (!isUpdating) {  
 isUpdating = true  
 \_millimeters.value = value  
 convertFromMillimeters(value)  
 isUpdating = false  
 }  
 }  
  
 private fun convertFromMillimeters(millimeterValue: String) {  
 val millimeters = millimeterValue.toDoubleOrNull() ?: return  
  
 val micrometers = millimeters \* 1000.0  
 val mils = millimeters / 0.0254  
 val lines = millimeters / 2.54  
 val inches = millimeters / 25.4  
  
 \_micrometers.value = micrometers.toString()  
 \_mils.value = mils.toString()  
 \_lines.value = lines.toString()  
 \_inches.value = inches.toString()  
 }  
  
 // Конвертация из линий  
 fun updateLines(value: String) {  
 if (!isUpdating) {  
 isUpdating = true  
 \_lines.value = value  
 convertFromLines(value)  
 isUpdating = false  
 }  
 }  
  
 private fun convertFromLines(lineValue: String) {  
 val lines = lineValue.toDoubleOrNull() ?: return  
  
 val micrometers = lines \* 2540.0  
 val mils = lines \* 100.0  
 val millimeters = lines \* 2.54  
 val inches = lines / 10.0  
  
 \_micrometers.value = micrometers.toString()  
 \_mils.value = mils.toString()  
 \_millimeters.value = millimeters.toString()  
 \_inches.value = inches.toString()  
 }  
  
 // Конвертация из дюймов  
 fun updateInches(value: String) {  
 if (!isUpdating) {  
 isUpdating = true  
 \_inches.value = value  
 convertFromInches(value)  
 isUpdating = false  
 }  
 }  
  
 private fun convertFromInches(inchValue: String) {  
 val inches = inchValue.toDoubleOrNull() ?: return  
  
 val micrometers = inches \* 25400.0  
 val mils = inches \* 1000.0  
 val millimeters = inches \* 25.4  
 val lines = inches \* 10.0  
  
 \_micrometers.value = micrometers.toString()  
 \_mils.value = mils.toString()  
 \_millimeters.value = millimeters.toString()  
 \_lines.value = lines.toString()  
 }  
}

1. Настроено взаимодействие между ViewModel и UI:

* В MainActivity подписаны EditText на изменения в ViewModel
* Реализованы слушатели doAfterTextChanged для обновления данных

package ru.olegkravtsov.lab8  
  
import android.os.Bundle  
import android.widget.EditText  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.widget.doAfterTextChanged  
import androidx.lifecycle.ViewModelProvider  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var viewModel: ConversionViewModel  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 enableEdgeToEdge()  
 setContentView(R.layout.activity\_main)  
  
 viewModel = ViewModelProvider(this)[ConversionViewModel::class.java]  
  
 val etMicrometers: EditText = findViewById(R.id.etMicrometers)  
 val etMils: EditText = findViewById(R.id.etMils)  
 val etMillimeters: EditText = findViewById(R.id.etMillimeters)  
 val etLines: EditText = findViewById(R.id.etLines)  
 val etInches: EditText = findViewById(R.id.etInches)  
  
 // Наблюдаем за изменениями в микрометрах  
 viewModel.micrometers.observe(this) { value ->  
 if (etMicrometers.text.toString() != value) {  
 etMicrometers.setText(value)  
 }  
 }  
  
 // Наблюдаем за изменениями в милах  
 viewModel.mils.observe(this) { value ->  
 if (etMils.text.toString() != value) {  
 etMils.setText(value)  
 }  
 }  
  
 // Наблюдаем за изменениями в миллиметрах  
 viewModel.millimeters.observe(this) { value ->  
 if (etMillimeters.text.toString() != value) {  
 etMillimeters.setText(value)  
 }  
 }  
  
 // Наблюдаем за изменениями в линиях  
 viewModel.lines.observe(this) { value ->  
 if (etLines.text.toString() != value) {  
 etLines.setText(value)  
 }  
 }  
  
 // Наблюдаем за изменениями в дюймах  
 viewModel.inches.observe(this) { value ->  
 if (etInches.text.toString() != value) {  
 etInches.setText(value)  
 }  
 }  
  
 // Слушаем изменения в полях ввода  
 etMicrometers.doAfterTextChanged {  
 viewModel.updateMicrometers(it.toString())  
 }  
  
 etMils.doAfterTextChanged {  
 viewModel.updateMils(it.toString())  
 }  
  
 etMillimeters.doAfterTextChanged {  
 viewModel.updateMillimeters(it.toString())  
 }  
  
 etLines.doAfterTextChanged {  
 viewModel.updateLines(it.toString())  
 }  
  
 etInches.doAfterTextChanged {  
 viewModel.updateInches(it.toString())  
 }  
 }  
}

**Результат работы**

Приложение успешно запускается и выполняет конвертацию между единицами измерения:

* Микрометры (µm)
* Милы (mil)
* Миллиметры (mm)
* Линии (line)
* Дюймы (inch)

При изменении значения в любом поле остальные поля автоматически обновляются. При повороте экрана или изменении конфигурации устройства введённые значения сохраняются (рис. 2).

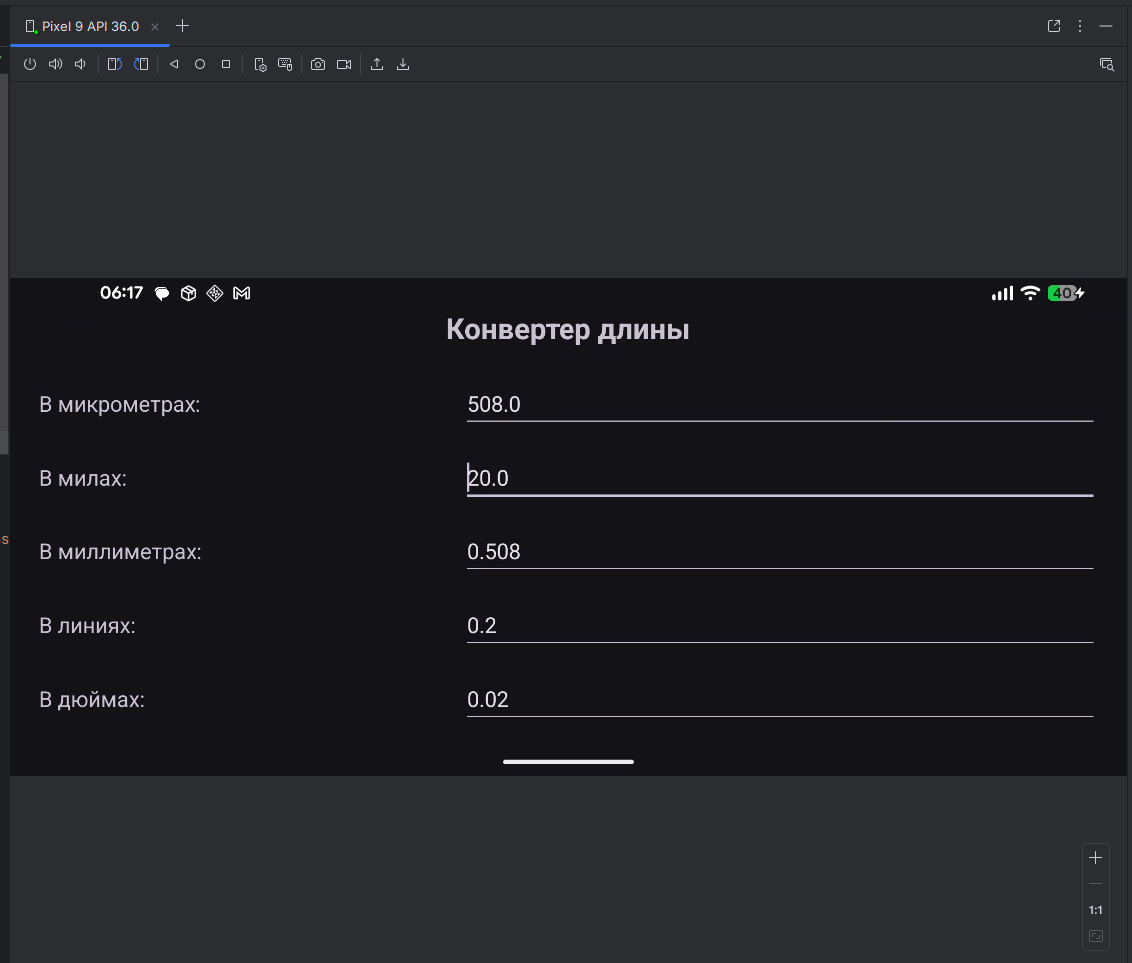
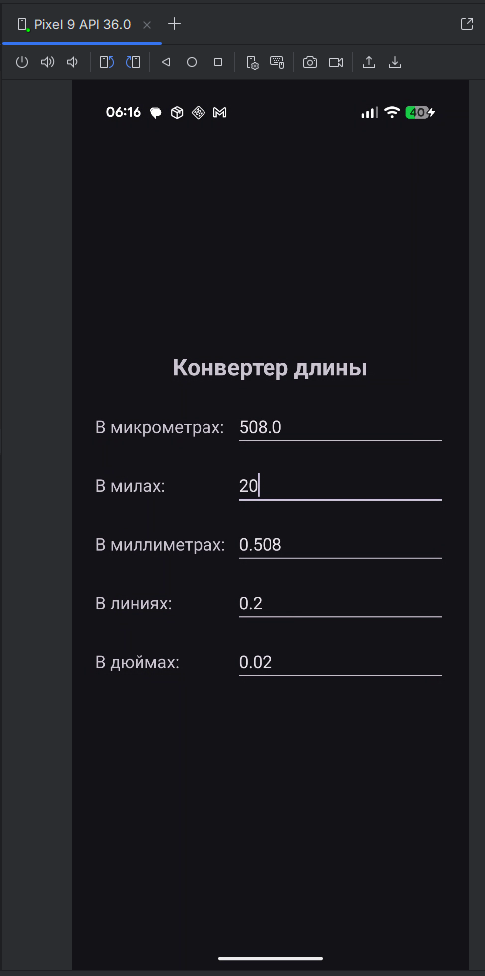


Рисунок 2 - Результат

# Выводы

В ходе выполнения лабораторной работы №8 были освоены принципы сохранения состояния данных в Android-приложениях с использованием архитектурного компонента ViewModel и подхода на основе LiveData. Разработанное приложение для конвертации единиц длины демонстрирует эффективное разделение ответственности между пользовательским интерфейсом и логикой, что позволяет корректно обрабатывать изменения конфигурации устройства без потери введённых данных. Реализована автоматическая синхронизация значений во всех полях ввода. Применение флага isUpdating предотвратило циклические обновления и обеспечило стабильность работы. В результате было создано отзывчивое приложение с возможностью мгновенного пересчёта значений и сохранения состояния при любых изменениях конфигурации.