Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетное образовательное учреждение высшего образования

«Российской экономический университет имени Г. В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ.**

ПРАКТИЧЕСКИЕ РАБОТЫ № 15

ДИСЦИПЛИНА: «РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ»

Тема: Работа с датчиками и создание гироскопа.

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

Листов:

|  |  |  |
| --- | --- | --- |
| Выполнил студент |  | Проверил преподаватель |
| Группа П50-3-18 |  | \_\_\_\_\_\_\_\_\_\_\_А.О.Лясников |
| Пахомов Даниил Александрович |  | «\_\_\_» \_\_\_\_\_\_\_\_2020 года |

Москва 2021

ПРАКТИЧЕСКАЯ РАБОТА №15

Работа с Widget

Цель: создать программу, которая будет взаимодействовать с widget и в widget нужно добавить список и выгрузить в него данные по MVVM модели.

1. Build.gradle

|  |
| --- |
| plugins **{** id 'com.android.application'  id 'kotlin-android'  id 'kotlin-android-extensions'  id 'kotlin-kapt' **}** android **{** compileSdkVersion 30  buildToolsVersion "30.0.2"   defaultConfig **{** applicationId "com.example.widgetmvvm"  minSdkVersion 16  targetSdkVersion 30  versionCode 1  versionName "1.0"   testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  **}** buildFeatures  **{** dataBinding = true  **}** buildTypes **{** release **{** minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  **}  }** compileOptions **{** sourceCompatibility JavaVersion.VERSION\_1\_8  targetCompatibility JavaVersion.VERSION\_1\_8  **}** kotlinOptions **{** jvmTarget = '1.8'  **} }** dependencies **{** implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin\_version"  implementation 'androidx.core:core-ktx:1.3.2'  implementation 'androidx.appcompat:appcompat:1.2.0'  implementation 'com.google.android.material:material:1.3.0'  implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  testImplementation 'junit:junit:4.+'  androidTestImplementation 'androidx.test.ext:junit:1.1.2'  androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'   kapt 'com.android.databinding:compiler:4.1.1'  implementation 'android.arch.lifecycle:extensions:1.1.1' **}** |

1. MainActivity

|  |
| --- |
| package com.example.widgetmvvm  import android.content.Intent import android.os.Bundle import android.view.View import android.widget.ListView import android.widget.Toast import androidx.appcompat.app.AppCompatActivity import com.example.widgetmvvm.adapter.ListViewAdapter import com.example.widgetmvvm.adapter.UserDto import com.example.widgetmvvm.model.Data import com.example.widgetmvvm.model.DataBaseHelper import kotlinx.android.synthetic.main.toolbar.\*   class MainActivity : AppCompatActivity() {  var adapter: ListViewAdapter? = null  var DB = DataBaseHelper(this)   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  cancel\_button.visibility = View.GONE   var listView: ListView? = null  listView = findViewById<ListView>(R.id.listview)   adapter = ListViewAdapter(this,DB.select())   listView?.adapter = adapter  adapter?.notifyDataSetChanged()  }   override fun onRestart() {  adapter?.notifyDataSetChanged()  super.onRestart()  }   fun addtask(v: View) {  Toast.makeText(this, "click", Toast.LENGTH\_SHORT).show()  var intent = Intent(this, AddTask::class.java)  startActivity(intent)  } } |

1. AddTask

|  |
| --- |
| package com.example.widgetmvvm  import android.content.Intent import android.os.Bundle import android.view.View import android.widget.Toast import androidx.appcompat.app.AppCompatActivity import androidx.databinding.DataBindingUtil import androidx.lifecycle.ViewModelProviders import com.example.widgetmvvm.databinding.ActivityAddTaskBinding import com.example.widgetmvvm.viewmodel.DataViewModel import com.example.widgetmvvm.viewmodel.DataViewModelFactory import kotlinx.android.synthetic.main.toolbar.\*  class AddTask : AppCompatActivity(), LoginResultCallBacks {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_add\_task*)   val activityAddTaskBinding =  DataBindingUtil.setContentView<ActivityAddTaskBinding>(this, R.layout.*activity\_add\_task*)  activityAddTaskBinding.*viewmodel* =  ViewModelProviders.of(this, DataViewModelFactory(this)).get(DataViewModel::class.*java*)   add.*visibility* = View.*GONE* }   override fun onSuccess(message: String) {  Toast.makeText(this, message, Toast.*LENGTH\_SHORT*).show()  }   override fun onError(message: String) {  Toast.makeText(this, message, Toast.*LENGTH\_SHORT*).show()  }   fun Cansel(v:View)  {  Toast.makeText(this, "click", Toast.*LENGTH\_SHORT*).show()  var intent = Intent(this, MainActivity::class.*java*)  startActivity(intent)  } } |

1. Model.Data

|  |
| --- |
| class Data(private var text: String, private var check: Int) : BaseObservable() {   val isDataValid: Boolean  get() = (!TextUtils.isEmpty(text))   fun getText(): String  {  return text  }   fun setText(text:String)  {  this.text = text  }   fun getCheck(): Int  {  return check  }   fun setCheck(check: Int)  {  this.check = check  } } |

1. Model.DataBaseHelper

|  |
| --- |
| package com.example.widgetmvvm.model  import android.content.ContentValues import android.content.Context import android.database.Cursor import android.database.sqlite.SQLiteDatabase import android.database.sqlite.SQLiteOpenHelper  class DataBaseHelper(context: Context?) : SQLiteOpenHelper(context, "data.db", null, 1) {   val TABLE\_NAME = "data\_table"  val COL = "TEXT\_TASK"  val COL2 = "CHEKED"   override fun onCreate(p0: SQLiteDatabase?) {  val createTable =  "CREATE TABLE $TABLE\_NAME (ID INTEGER PRIMARY KEY AUTOINCREMENT, $COL TEXT,$COL2 INTEGER)"  p0!!.execSQL(createTable)  }   override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {  }   fun addData(text: String, check: Int): Boolean {  val db = this.*writableDatabase* val contentValues = ContentValues()  contentValues.put(COL, text)  contentValues.put(COL2, check)  val result = db.insert(TABLE\_NAME, null, contentValues)  return result != -1L;  }   fun select(): ArrayList<Data> {  var arrayList: ArrayList<Data> = ArrayList()  val db = this.*writableDatabase* var cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)    if (cursor!!.getCount() == 0) {  return ArrayList()  }   while (cursor.moveToNext()) {  arrayList.add(Data(cursor.getString(1), cursor.getInt(2)))  }   return arrayList  }   fun selectTrue(): ArrayList<Data> {  var arrayList: ArrayList<Data> = ArrayList()  val db = this.*writableDatabase* var cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)    if (cursor!!.getCount() == 0) {  return ArrayList()  }   while (cursor.moveToNext()) {  if (cursor.getInt(2) == 1)  arrayList.add(Data(cursor.getString(1), cursor.getInt(2)))  }   return arrayList  }  } |

1. Viewmodel.DataViewModel

|  |
| --- |
| package com.example.widgetmvvm.viewmodel  import android.content.Context import android.content.Intent import android.text.Editable import android.text.TextWatcher import android.view.View import android.widget.CheckBox import androidx.lifecycle.ViewModel import com.example.widgetmvvm.LoginResultCallBacks import com.example.widgetmvvm.MainActivity import com.example.widgetmvvm.adapter.ListViewAdapter import com.example.widgetmvvm.model.Data import com.example.widgetmvvm.model.DataBaseHelper import java.util.\*  class DataViewModel(private val listener: LoginResultCallBacks) : ViewModel() {   private val data: Data   init {  this.data = Data("", 0);  }   val TextWatcher: TextWatcher  get() = object : TextWatcher {  override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun afterTextChanged(p0: Editable?) {  data.setText(p0.toString())  }  }   fun test(v: View) {  var check = v as CheckBox  if (v.isChecked) {  listener.onSuccess("true")  data.setCheck(1)  } else {  listener.onSuccess("false")  data.setCheck(0)  }  }   var DB: DataBaseHelper? = null  var adapter: ListViewAdapter? = null  fun onAddClicked(v: View) {  if (data.isDataValid) {  DB = DataBaseHelper(v.context)  DB!!.addData(data.getText(), data.getCheck())  v.getContext().startActivity(Intent(v.getContext(), MainActivity::class.java))  listener.onSuccess("Данные были добавлены")  } else {  listener.onError("Ошибка")  }   }  } |

1. Viewmodel.DataViewModleFactory

|  |
| --- |
| package com.example.widgetmvvm.viewmodel  import androidx.lifecycle.ViewModel import androidx.lifecycle.ViewModelProvider import com.example.widgetmvvm.LoginResultCallBacks  class DataViewModelFactory (private val listener: LoginResultCallBacks) :  ViewModelProvider.NewInstanceFactory() {  override fun <T : ViewModel?> create(modelClass: Class<T>): T {  return DataViewModel(listener)as T  } } |

1. Adapter.ListViewAdapter

|  |
| --- |
| package com.example.widgetmvvm.adapter  import android.app.Activity import android.content.Context import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import android.widget.BaseAdapter import android.widget.TextView import com.example.widgetmvvm.R import com.example.widgetmvvm.model.Data   class ListViewAdapter(private var activity: Context, private var items: ArrayList<Data>): BaseAdapter() {   private class ViewHolder(row: View?) {  var txtName: TextView? = null   init {  this.txtName = row?.findViewById<TextView>(R.id.*txtName*)  }  }   override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {  val view: View?  val viewHolder: ViewHolder  if (convertView == null) {  val inflater = activity?.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*) as LayoutInflater  view = inflater.inflate(R.layout.*list\_view*, null)  viewHolder = ViewHolder(view)  view?.*tag* = viewHolder  } else {  view = convertView  viewHolder = view.*tag* as ViewHolder  }   var userDto = items[position]  viewHolder.txtName?.*text* = userDto.getText()   return view as View  }   override fun getItem(i: Int): Data {  return items[i]  }   override fun getItemId(i: Int): Long {  return i.toLong()  }   override fun getCount(): Int {  return items.size  } }  class UserDto {  var name: String = ""   constructor() {}   constructor(name: String) {  this.name = name  } } |

1. Adapter.ListViewAdapterWidget

|  |
| --- |
| package com.example.widgetmvvm.adapter  import android.appwidget.AppWidgetManager import android.content.Context import android.content.Intent import android.widget.RemoteViews import android.widget.RemoteViewsService import android.widget.RemoteViewsService.RemoteViewsFactory import com.example.widgetmvvm.R import com.example.widgetmvvm.model.Data import com.example.widgetmvvm.model.DataBaseHelper   class ListViewAdapterWidget(context: Context, intent: Intent) :  RemoteViewsService.RemoteViewsFactory {   private lateinit var data: ArrayList<Data>  private var context: Context  private var widgetID: Int  private var DB: DataBaseHelper   init {  this.context = context  widgetID = intent.getIntExtra(  AppWidgetManager.EXTRA\_APPWIDGET\_ID,  AppWidgetManager.INVALID\_APPWIDGET\_ID  )  DB = DataBaseHelper(context)  }   override fun onCreate() {  data = ArrayList<Data>()  }    override fun onDataSetChanged() {  data.clear()  DB = DataBaseHelper(context)  data = DB.selectTrue()  }   override fun onDestroy() {  }   override fun getCount(): Int {  return data.size  }   override fun getViewAt(position: Int): RemoteViews {  var remoteViews = RemoteViews(context.packageName, R.layout.list\_view)  remoteViews.setTextViewText(R.id.txtName, data!![position].getText())  return remoteViews  }   override fun getLoadingView(): RemoteViews? {  return null  }   override fun getViewTypeCount(): Int {  return 1  }   override fun getItemId(position: Int): Long {  return position.toLong()  }   override fun hasStableIds(): Boolean {  return true  } } |

1. ListViewAppWidget

|  |
| --- |
| package com.example.widgetmvvm  import android.app.PendingIntent import android.appwidget.AppWidgetManager import android.appwidget.AppWidgetProvider import android.content.Context import android.content.Intent import android.widget.RemoteViews   */\*\*  \* Implementation of App Widget functionality.  \*/* class ListViewAppWidget : AppWidgetProvider() {  override fun onUpdate(  context: Context, appWidgetManager: AppWidgetManager,  appWidgetIds: IntArray  ) {  super.onUpdate(context, appWidgetManager, appWidgetIds)  for (i in appWidgetIds) {  updateWidget(context, appWidgetManager, i)  }  }   fun updateWidget(  context: Context, appWidgetManager: AppWidgetManager,  appWidgetId: Int  ) {  val rv = RemoteViews(  context.*packageName*,  R.layout.*list\_view\_app\_widget* )  setUpdateTV(rv, context, appWidgetId)  setList(rv, context, appWidgetId)  setListClick(rv, context, appWidgetId)  appWidgetManager.updateAppWidget(appWidgetId, rv)  appWidgetManager.notifyAppWidgetViewDataChanged(appWidgetId,  R.id.*list\_item*);  }   fun setUpdateTV(rv: RemoteViews, context: Context?, appWidgetId: Int) {  val updIntent = Intent(context, ListViewAppWidget::class.*java*)  updIntent.*action* = AppWidgetManager.*ACTION\_APPWIDGET\_UPDATE* updIntent.putExtra(AppWidgetManager.*EXTRA\_APPWIDGET\_IDS*, *intArrayOf*(appWidgetId))  val updPIntent = PendingIntent.getBroadcast(  context,  appWidgetId, updIntent, 0  )  rv.setOnClickPendingIntent(R.id.*refresh*, updPIntent)  }   fun setList(rv: RemoteViews, context: Context?, appWidgetId: Int) {  val adapter = Intent(context, MyService::class.*java*)  adapter.putExtra(AppWidgetManager.*EXTRA\_APPWIDGET\_ID*, appWidgetId)  rv.setRemoteAdapter(R.id.*list\_item*, adapter)  }   fun setListClick(rv: RemoteViews?, context: Context?, appWidgetId: Int) {} } |

1. MyService

|  |
| --- |
| package com.example.widgetmvvm  import android.content.Intent import android.widget.RemoteViewsService import com.example.widgetmvvm.adapter.ListViewAdapterWidget  class MyService : RemoteViewsService() {  override fun onGetViewFactory(intent: Intent): RemoteViewsFactory {  return ListViewAdapterWidget(applicationContext, intent)  } } |

1. LoginResultCallBacks

|  |
| --- |
| package com.example.widgetmvvm  interface LoginResultCallBacks {  fun onSuccess(message: String)  fun onError(message: String) } |

1. AndroidManifest

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.widgetmvvm">   <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/Theme.WidgetMVVM">   <service  android:name="MyService"  android:permission="android.permission.BIND\_REMOTEVIEWS"/>  <receiver android:name=".ListViewAppWidget">  <intent-filter>  <action android:name="android.appwidget.action.APPWIDGET\_UPDATE" />  </intent-filter>   <meta-data  android:name="android.appwidget.provider"  android:resource="@xml/list\_view\_app\_widget\_info" />  </receiver>   <activity android:name=".AddTask" />  <activity android:name=".MainActivity">  <intent-filter>  <action android:name="android.intent.action.MAIN" />   <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

Вывод: научился создавать мобильное приложение с widget в который выводится список по MVVM модели.