**Создание приложения по получения на телефон перечня устройств BlueTooth к которым можно подключиться и подключение к одному из них (часть 1)**

1. Создаем приложение с использование языка Котлин на основе EmptyActivity
2. Прописываем класс Adapter который отвечает за работу блютуз на смартфоне

class MainActivity : AppCompatActivity() {  
  **private var btAdapter: BluetoothAdapter? = null**  
  
 override fun onCreate(savedInstanceState: Bundle?) {

1. Создаем процедуру для инициализации блютуз менеджера и блютуз адаптера затем

private fun init(){

val btManager = getSystemService(Context.*BLUETOOTH\_SERVICE*) as BluetoothManager  
 btAdapter = btManager.*adapter* }

1. Инициализируем функцию прописав ее название в onCreate

onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 **init()**

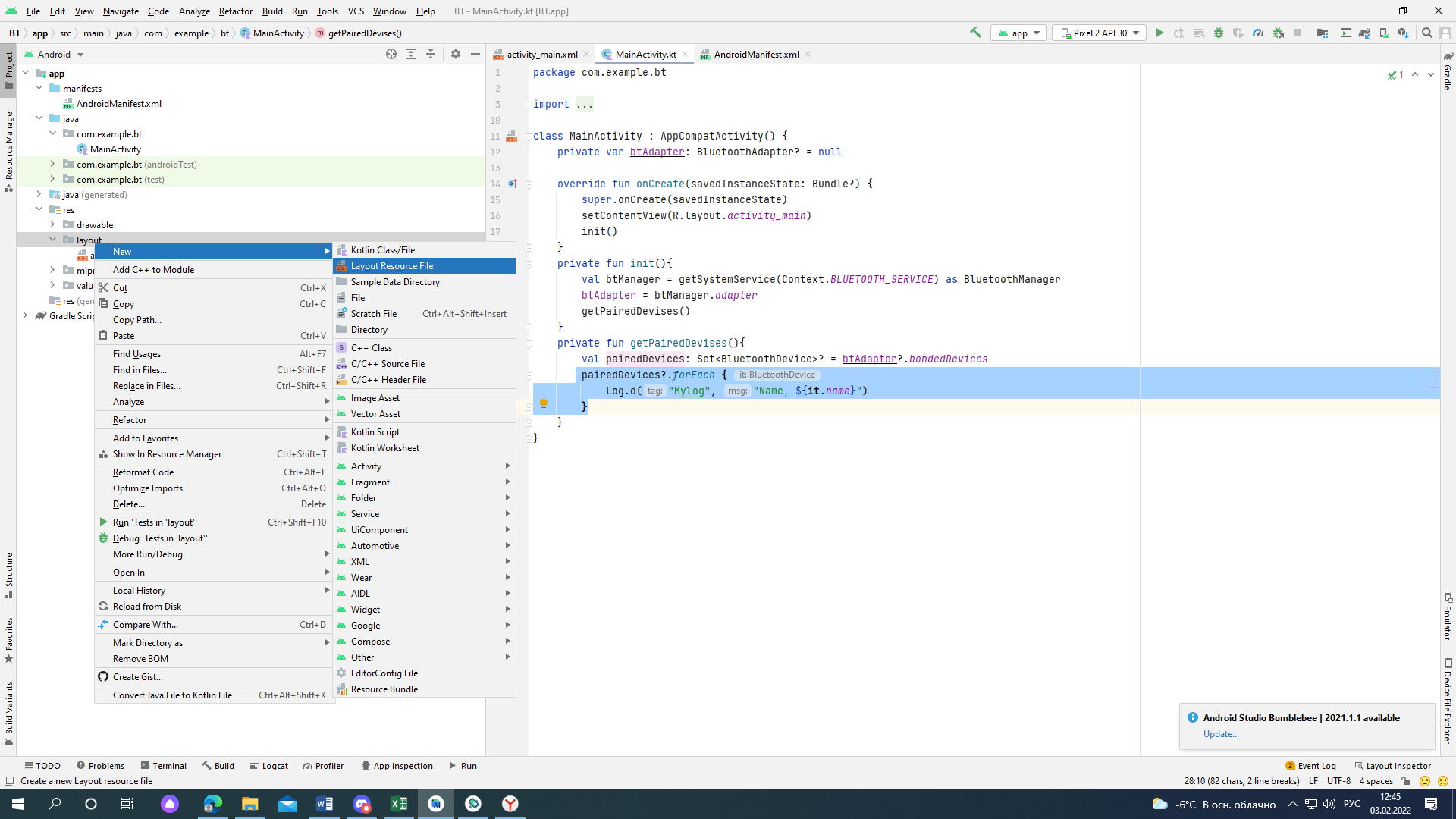
1. Создаем функцию, которая будет выдавать список на устройстве всех подключенных устройств

private fun getPairedDevises(){  
 val pairedDevices: Set<BluetoothDevice>? = btAdapter?.*bondedDevices*}

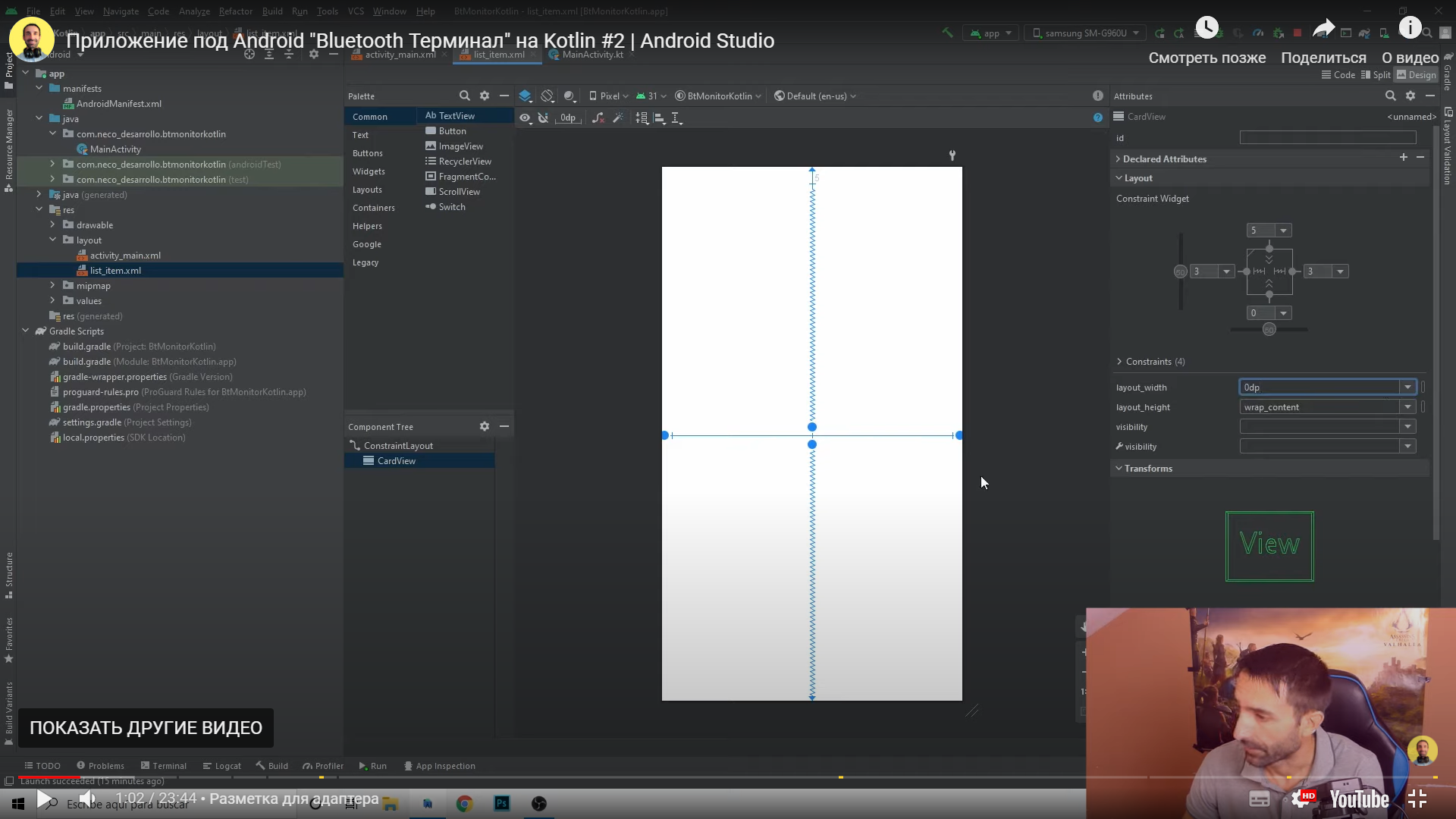
1. Подключаем в файле AndroidManifest разрешения на использования блютуз

<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.neco\_desarrollo.btmonitorkotlin">  
**<uses-permission** **android:name="android.permission.BLUETOOTH\_CONNECT" />  
<uses-permission android:name="android.permission.BLUETOOTH" />** <application

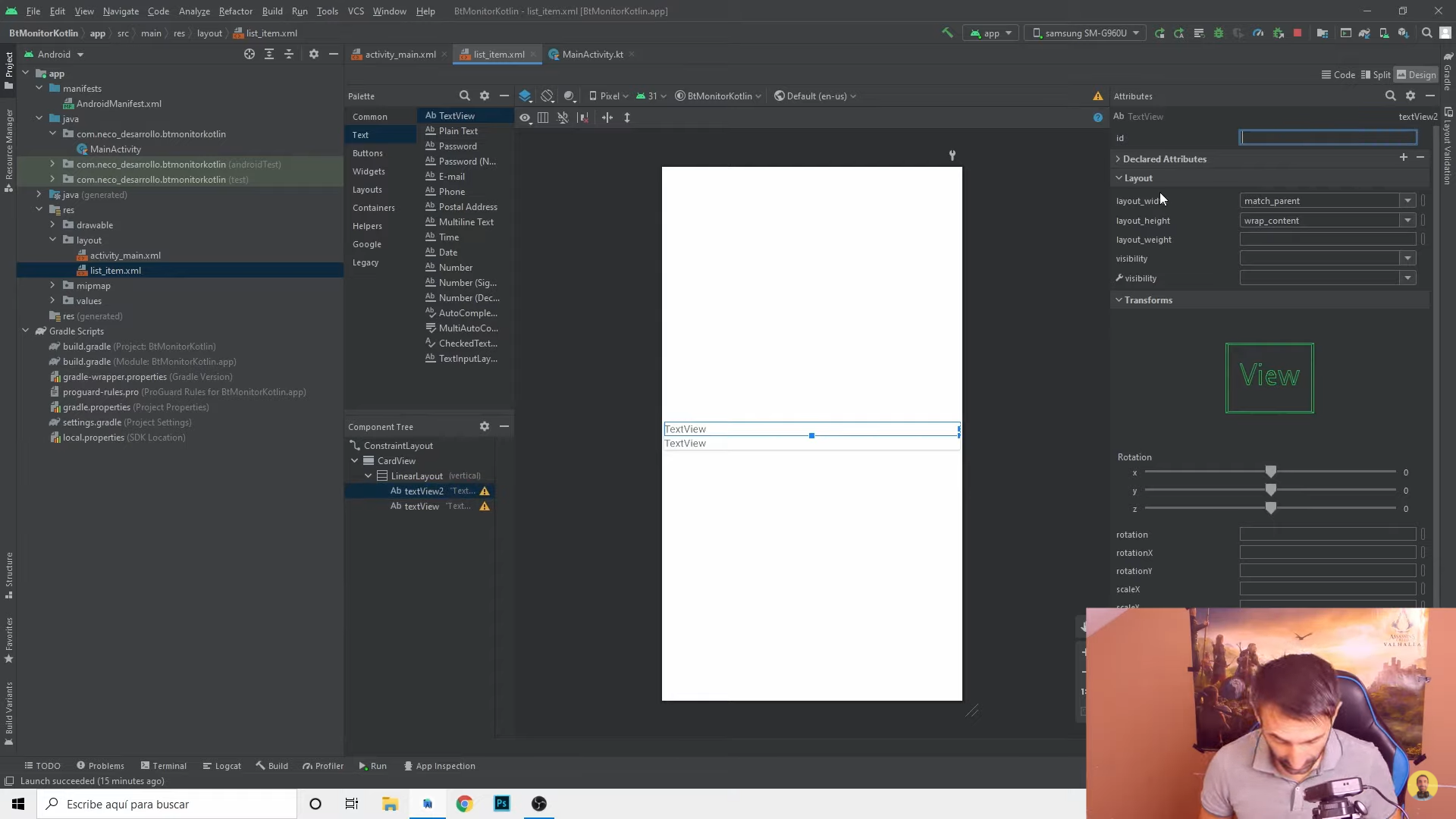
1. Создаем отображение списка устройств в приложении. Добавляем новый ресурс в layout с именем list\_item



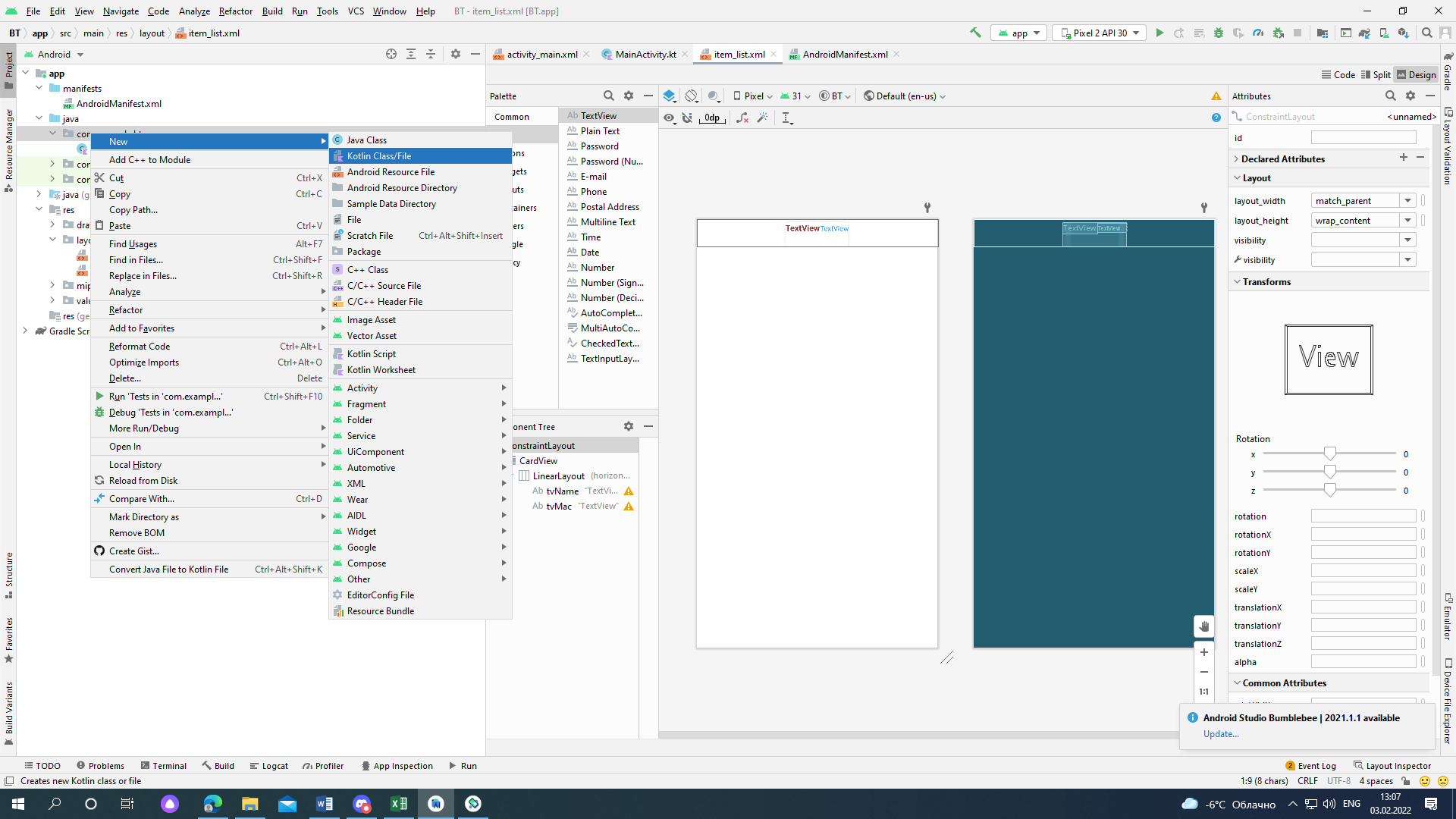
1. Переносим компонент CardView и устанавливаем у него следующие параметры

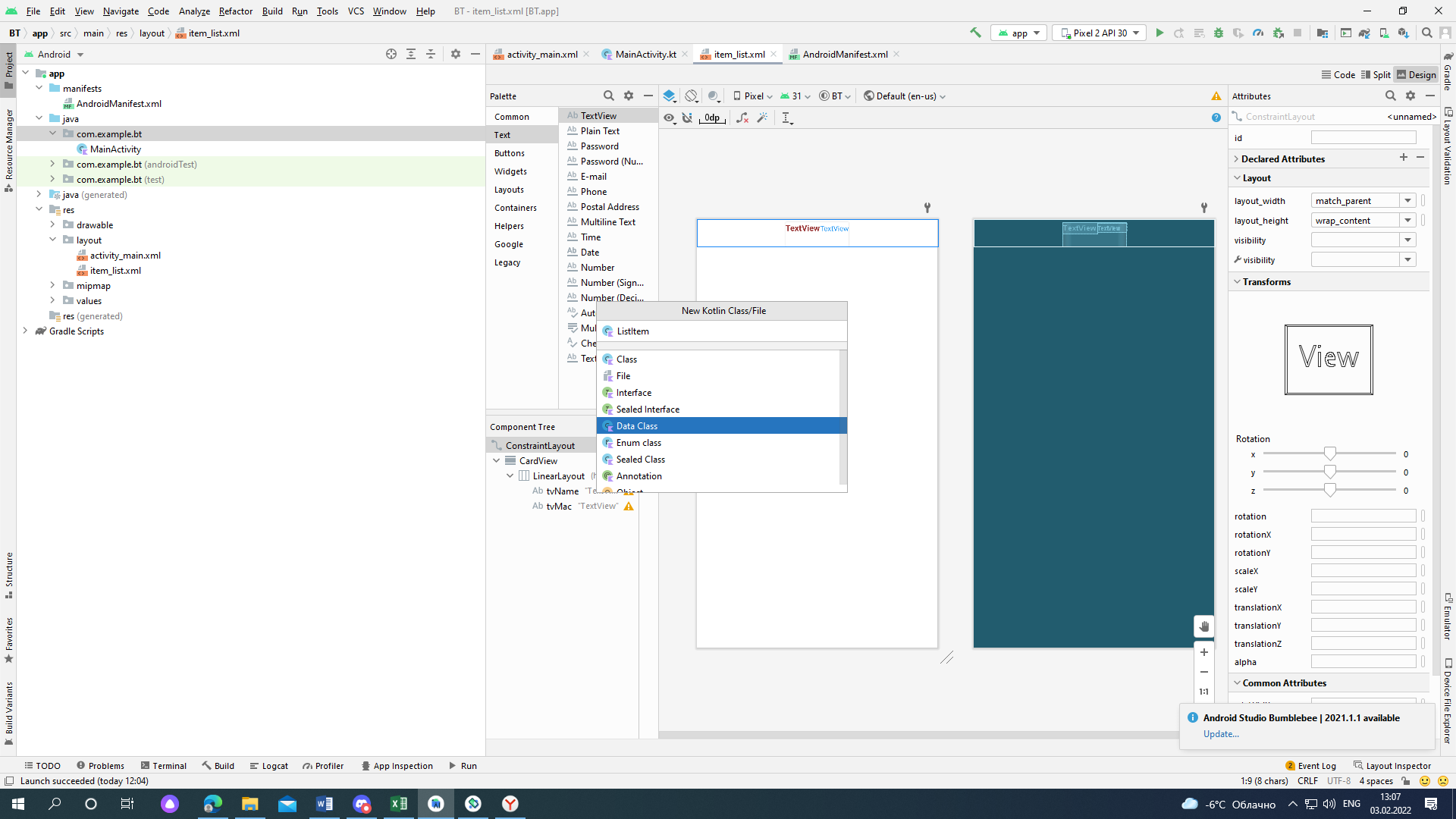


1. Переносим компоненты LinerLayout c pading = 5dp, TextView две штуки с id tvName и tvMac



1. И у ConstraintLayout Делаем layout\_height=wrapcontent
2. Создаем новый класс ListItem

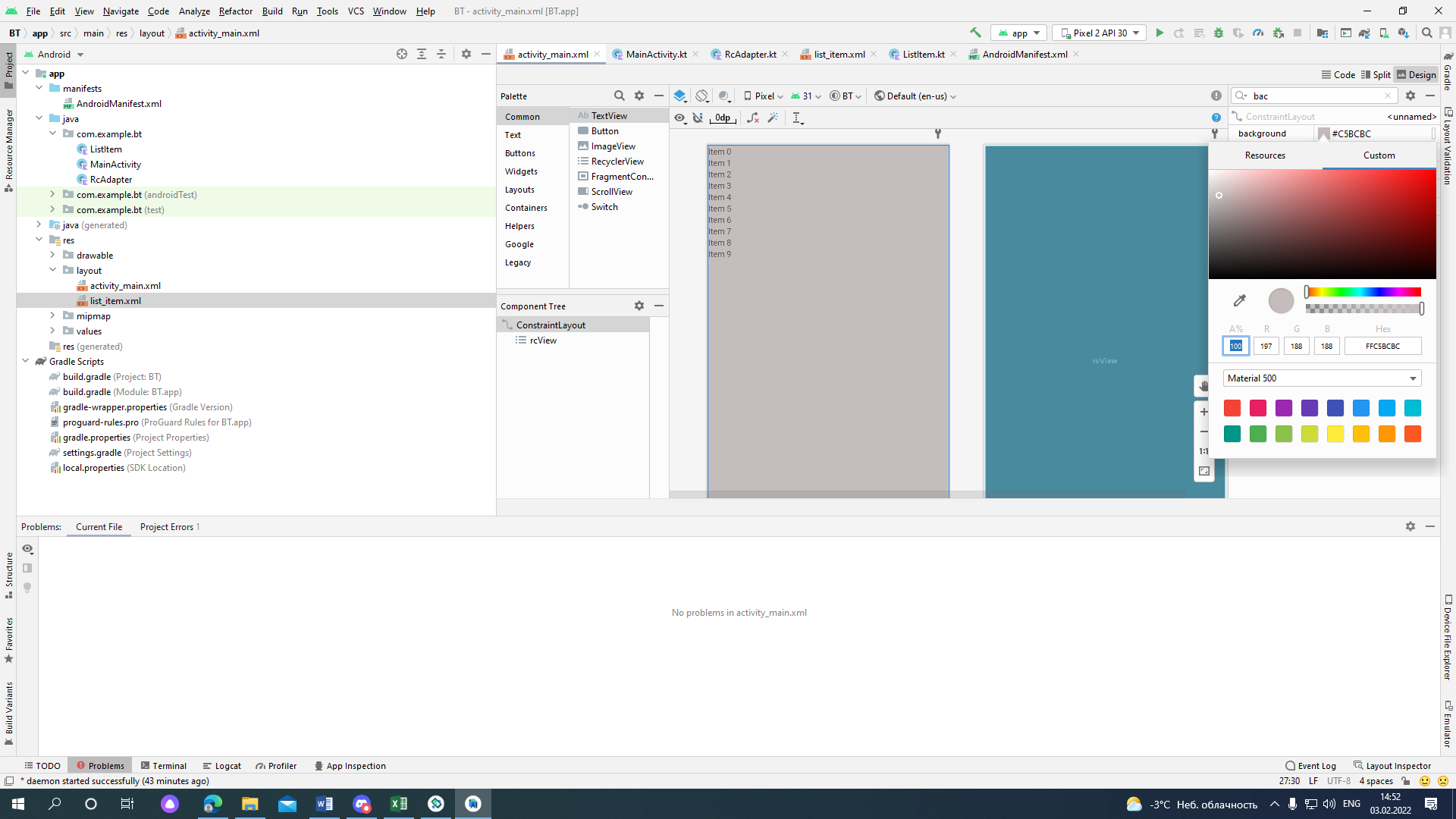




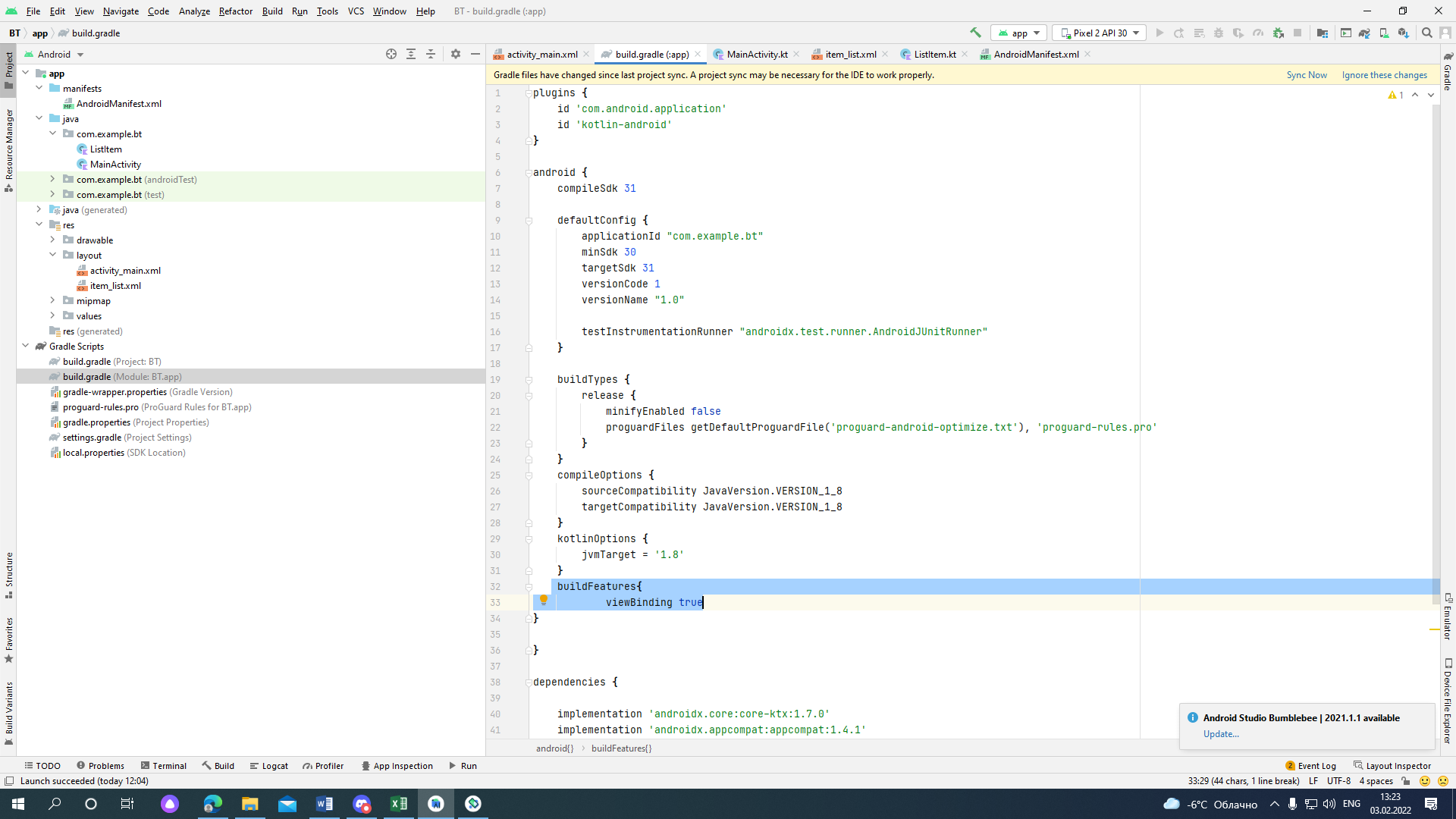
data class ListItem(  
 var name: String,  
 var mac: String  
)

мы описали данные которые будем использовать

1. Создаем сам список: в ActivityMain удаляем textView и добавляем RecyclerView ( id=rcView)прикрепляем его с отступами по 0, устанавливаем по высоте и ширине math\_constraint. Иделаем цвет фона серый, используя свойство background у СonstraintLayout



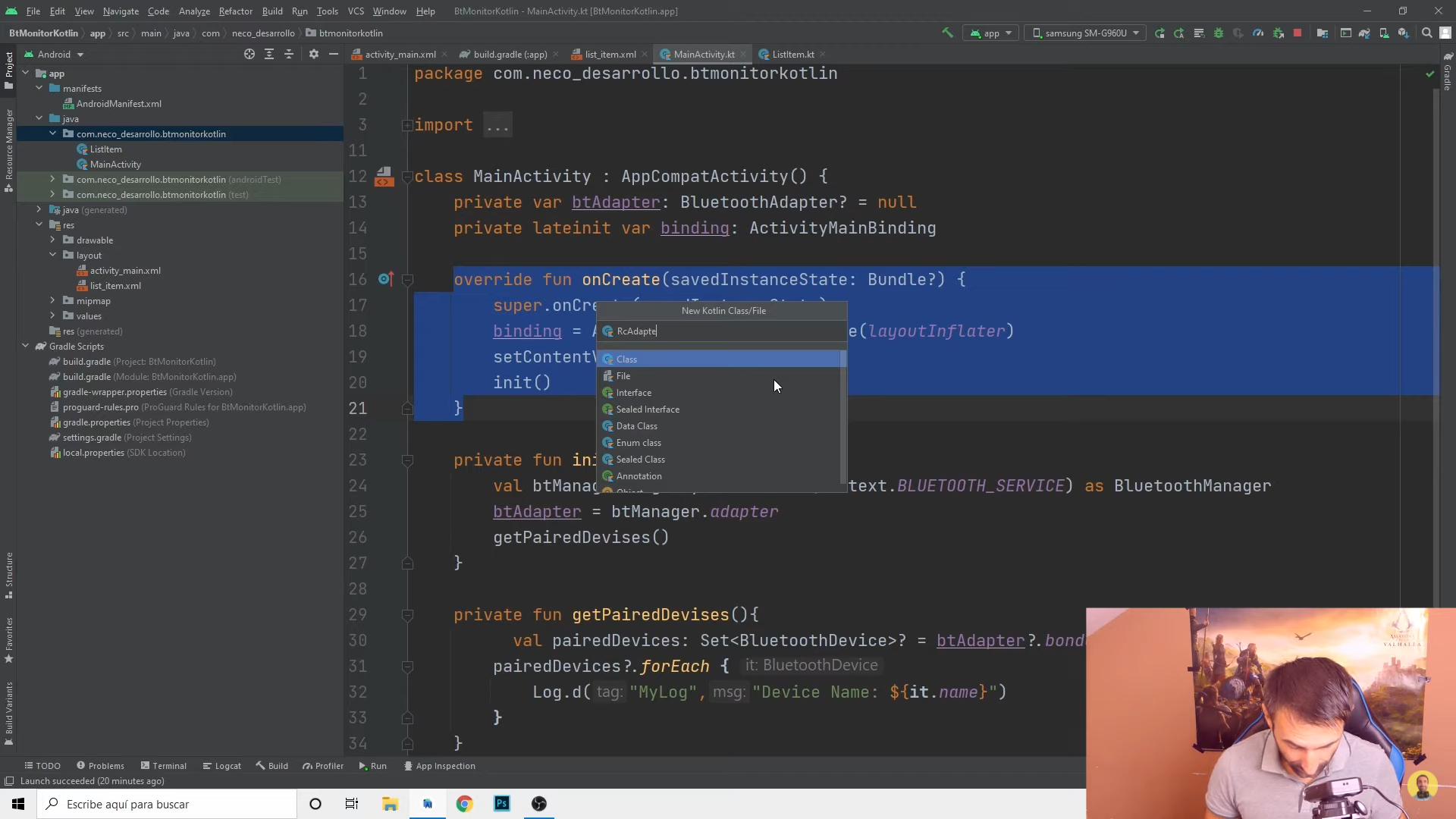
1. В buildgradle(Module) добавим Viewbunding
2. Этот класс содержит все привязки от свойств макета (например, пользовательской переменной) до представлений макета и знает, как назначать значения для выражений привязки.



И его активизируем в MainActivity

class MainActivity : AppCompatActivity() {  
 private var btAdapter: BluetoothAdapter? = null  
 **private lateinit var binding: ActivityMainBinding**  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 **binding = ActivityMainBinding.inflate(*layoutInflater*)** setContentView(binding.root)  
 init()  
}

1. Создаем класс адаптера



Класс будет брать данные и заполнять список class RcAdapter

В качестве данных он будет использовать данные списка ListItem и эти данные будет хранить класс ViewHolder, но его еще нет и его надо создать тут же и назовем его ItemHolder

class RcAdapter : ListAdapter<ListItem, RcAdapter.ItemHolder>(ItemComparator()) {  
  
 class ItemHolder(view: View) : RecyclerView.ViewHolder(view){  
 val binding = ListItemBinding.bind(view)  
fun setData(item: ListItem) = *with*(binding)**{//функция будет заполнять наши элем.-ы** tvName.*text* = item.name  
 tvMac.*text* = item.mac  
 **}** companion object{//запускаем предыдущий класс и заполняем  
 fun create(parent: ViewGroup): ItemHolder{  
 return ItemHolder(LayoutInflater.from(parent.*context*).inflate(R.layout.*list\_item*, parent, false))  
 }  
 }  
 }  
  
 class ItemComparator : DiffUtil.ItemCallback<ListItem>(){//класс который будет //сравнивать элементы при записи, нужно ли их обновлять  
 override fun areItemsTheSame(oldItem: ListItem, newItem: ListItem): Boolean {  
 return oldItem.mac == newItem.mac  
 }  
  
 override fun areContentsTheSame(oldItem: ListItem, newItem: ListItem): Boolean {  
 return oldItem == newItem  
 }  
 }  
//функция для создания списка  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ItemHolder {  
 return ItemHolder.create(parent)  
 }  
//заполняется позиция списка  
 override fun onBindViewHolder(holder: ItemHolder, position: Int) {  
holder.setData(getItem(position), listener)

}  
}

1. Прописываем наш адаптер в class MainActivity

private lateinit var adapter: RcAdapter

и инициализируем его в функции init

adapter = RcAdapter()

и инициализируем адаптер

binding.rcView.layoutManager = LinearLayoutManager(this)  
binding.rcView.adapter = adapter

1. создаем временный список для передачи данных в функции getPairedDevises

private fun getPairedDevises(){  
 val pairedDevices: Set<BluetoothDevice>? = btAdapter?.*bondedDevices***val tempList = ArrayList<ListItem>()  
 pairedDevices?.*forEach* {//передаем элементы в список  
 tempList.add(ListItem(it.*name*, it.*address*))//передаем их в адаптер** **}**

**Проверка возможна только если подключено реальное устройство у которого есть уже список подключенных устройств**