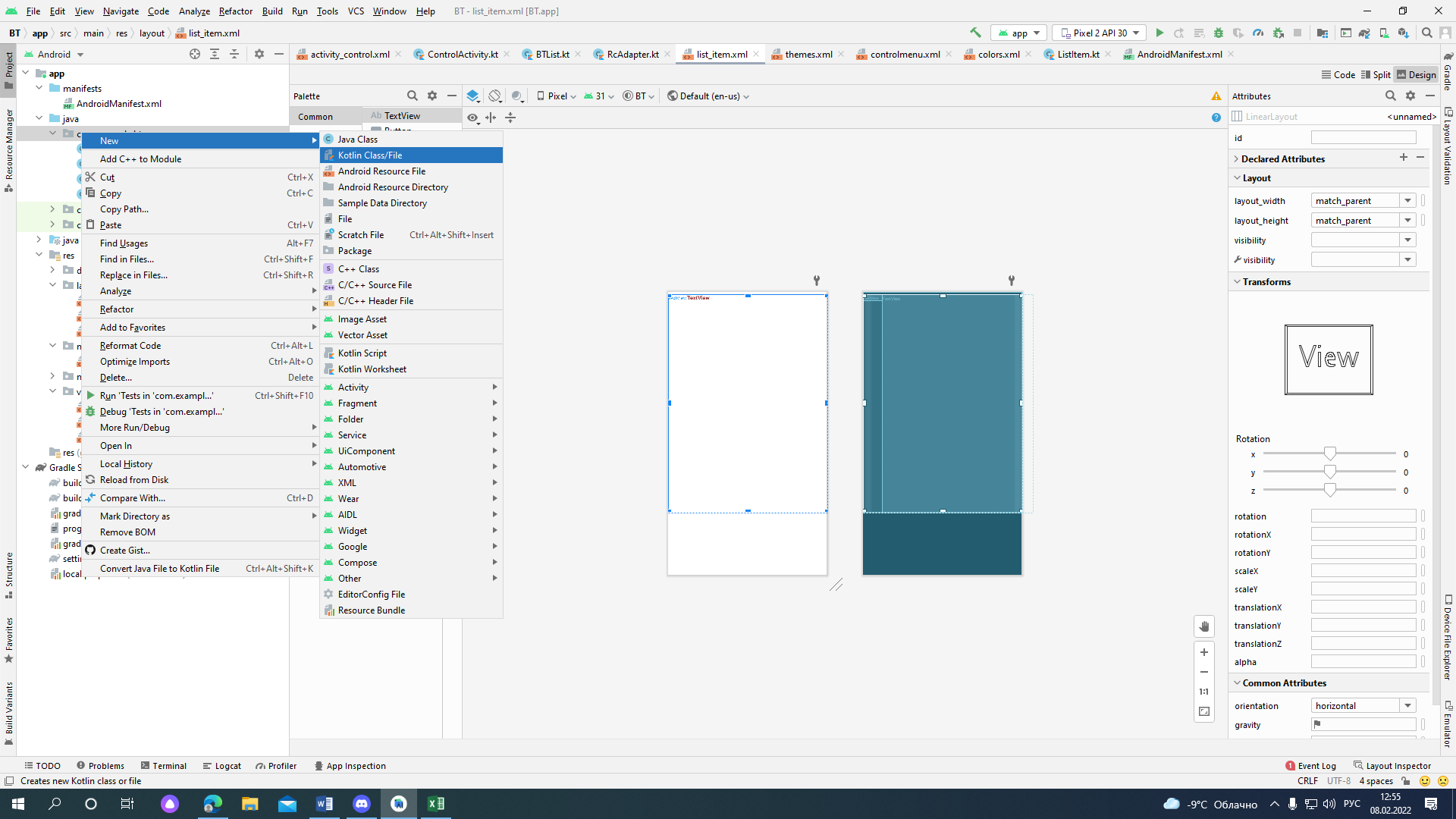
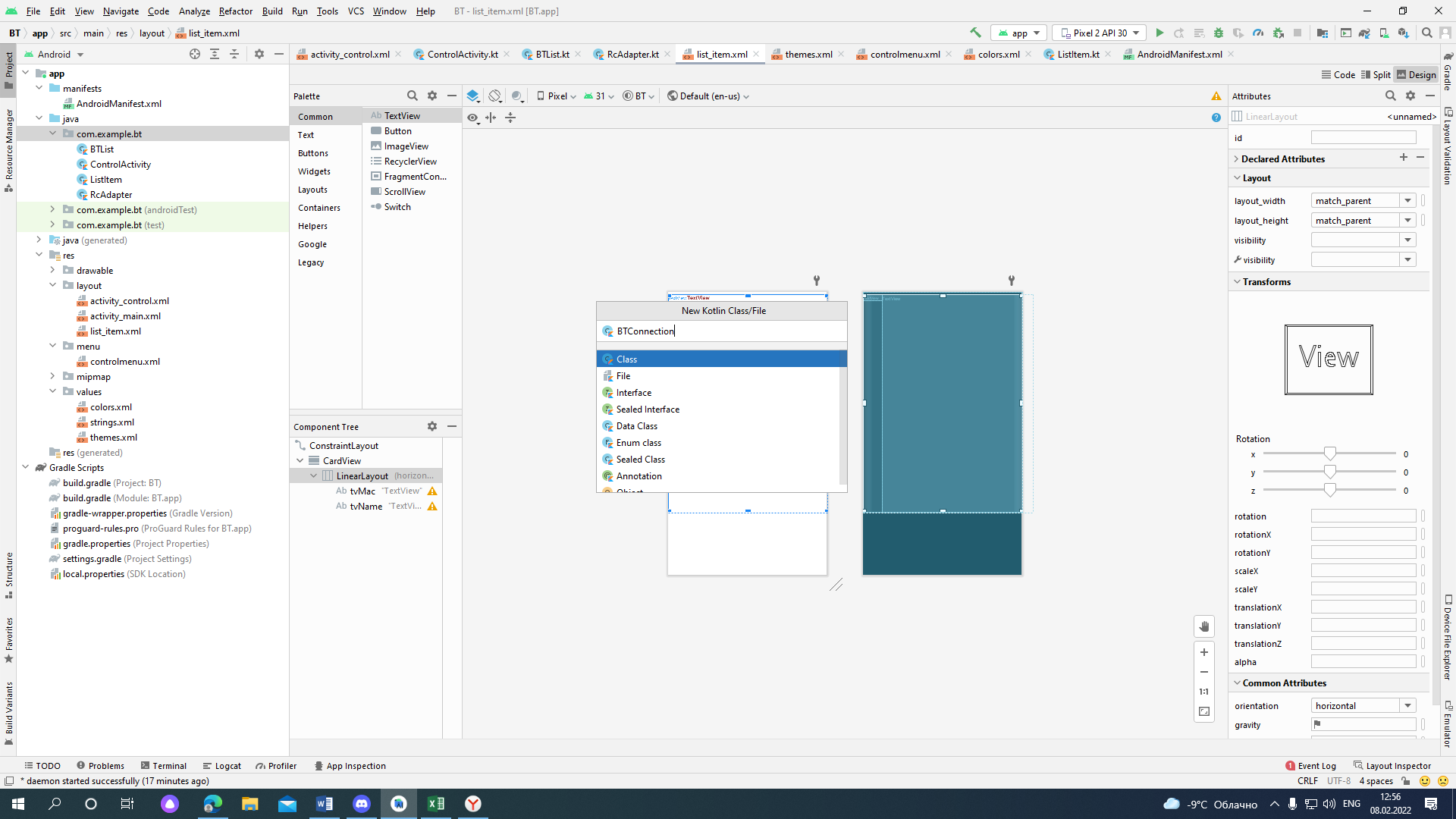
**Создание приложения по получения на телефон перечня устройств BlueTooth к которым можно подключиться и подключение к одному из них (часть 3)**

В этой части мы создаем два класса, для подключения

1. Создаем класс BTConnection





class BTConnection (private val adapter: BluetoothAdapter) {  
 lateinit var cThread: ConnectThread

1. И создадим в нем функцию подключения, в эту функцию мы будем передавать mac адрес для подключения, в качестве переменной используем адаптер, который мы создавали ранее, он находит устройство

fun connect(mac: String) {  
 if (adapter.*isEnabled* && mac.*isNotEmpty*()) {  
 val device = adapter.getRemoteDevice(mac)  
 device.*let* **{** cThread = ConnectThread(**it**)  
 cThread.start()  
 **}** }  
 }

1. Создаем класс второстепенного потока ConnectThread типа Thread
2. Определяем в функцию в нем run для запуска потока, при этом обязательно создаем исключение, чтобы видеть если не будет подключение. Всю информацию будем видеть в Logcat
3. override fun run() {  
    try {  
    Log.d("MyLog","Connecting...")  
    mSocket?.connect()  
    Log.d("MyLog","Connected")  
    }catch (i: IOException){//исключение  
    Log.d("MyLog","Can not connect to device")  
    closeConnection()  
    }  
    }
4. и пропишем в самом классе переменные и инициализируем данный класс

val uuid = "00001101-0000-1000-8000-00805F9B34FB"  
var mSocket: BluetoothSocket? = null  
  
init {  
 try {  
 mSocket = device.createRfcommSocketToServiceRecord(UUID.fromString(uuid))  
 }catch (i: IOException){  
  
 }

1. и добавим функцию закрытия соединения

fun closeConnection(){  
 try {  
 mSocket?.close()  
 }catch (i: IOException){  
  
 }  
 }

1. и добавим переменную устройства в класс

class ConnectThread(**private val device: BluetoothDevice**) : Thread() {

1. добавляем в класс class ControlActivity

lateinit var btConnection: BTConnection

1. и создаем функцию для инициализации в данном классе, передаем в нее адаптер из BTConnection

private fun init(){  
 val btManager = getSystemService(Context.*BLUETOOTH\_SERVICE*) as BluetoothManager  
 val btAdapter = btManager.*adapter* btConnection = BTConnection(btAdapter)  
}

1. и допишем запуск этой функции по событию onCreate

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityControlBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 onBtListResult()  
 **init()  
}**

1. создаем глобальную переменную в этом классе, которая будет в себе хранить mac адрес

class ControlActivity : AppCompatActivity() {  
  
private lateinit var binding: ActivityControlBinding  
private lateinit var actListLauncher: ActivityResultLauncher<Intent>  
lateinit var btConnection: BTConnection  
**private var listItem: ListItem? = null**

1. и в функции fun onBtListResult() изменим код для получения адреса

было

if(**it**.*resultCode* == *RESULT\_OK*){  
Log.d("MyLog","Name: ${(**it**.*data*?.getSerializableExtra(BTList.DEVICE\_KEY) as ListItem).name}")

Стало

if(**it**.*resultCode* == *RESULT\_OK*){  
listItem = **it**.*data*?.getSerializableExtra(BTList.DEVICE\_KEY) as ListItem

1. и в функции fun onOptionsItemSelected запустим, проверив что значение списка не нулевое

override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 if(item.*itemId* == R.id.*id\_list*){  
 actListLauncher.launch(Intent(this, BTList::class.*java*))  
 } else if(item.*itemId* == R.id.*id\_connect*){  
 **listItem.*let* {  
 btConnection.connect(it?.mac!!)** **}** }  
 return super.onOptionsItemSelected(item)  
}

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