package com.example.pr\_13

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import android.widget.Toast

import androidx.recyclerview.widget.DividerItemDecoration

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import androidx.room.\*

import com.google.gson.annotations.SerializedName

import kotlinx.coroutines.DelicateCoroutinesApi

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

import retrofit2.Retrofit

import retrofit2.converter.gson.GsonConverterFactory

import retrofit2.http.GET

import retrofit2.http.Path

import java.util.\*

@Entity

data class AppealsRoom(

@ColumnInfo(name="Appeal String")

val appealString: String,

@ColumnInfo(name="Appeal Symbol")

val appealSymbol: String

)

{

@PrimaryKey(autoGenerate = true)

var id: Int = 0

}

@Dao

interface AppealsDao {

@Query("SELECT \* FROM AppealsRoom")

fun getEverything(): List<AppealsRoom>

@Insert

fun insertAll(vararg appeals: AppealsRoom)

@Delete

fun delete(appealRoom: AppealsRoom)

@Query("DELETE FROM AppealsRoom")

fun dropAll()

}

@Database(entities = [AppealsRoom::class], version = 1)

abstract class DB: RoomDatabase() {

abstract fun AppealsDao(): AppealsDao

}

data class Appeals(

@SerializedName("localString")

var appealLocalName: String? = "",

@SerializedName("localsymbol")

var appealLocalDate: String? = ""

)

interface APIPublicAppeals {

@GET("v3/publicAppeals/{year}/{secondField}")

fun getAppeals(@Path("firstfield") firstfield: string, @Path("secondField") secondField: Char): Call<MutableList<Appeals>>

}

class AppealsAdapter(private val AppealsList: List<Appeals>): RecyclerView.Adapter<AppealsAdapter.AppealsViewHolder>() {

fun getResult() = AppealsList

class AppealsViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {

//name = string

//date = symbol

fun fill(string: String, symbol: String) {

itemView.findViewById<TextView>(R.id.appealString).text = string

itemView.findViewById<TextView>(R.id.appealSymbol).text = symbol

}

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): AppealsViewHolder {

val itemView = LayoutInflater.from(parent.context).inflate(R.layout.appeal, parent, false)

return AppealsViewHolder(itemView)

}

override fun onBindViewHolder(holder: AppealsViewHolder, position: Int) {

val appeal = AppealsList[position]

holder.fill(appeal.appealLocalString!!, appeal.appealLocalSymbol!!)

}

override fun getItemCount(): Int = AppealsList.size

}

@DelicateCoroutinesApi

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val retrofit = Retrofit.Builder().baseUrl("https://10.0.2.2:10001/?input=privet").

addConverterFactory(GsonConverterFactory.create()).build()

val publicAppeals = retrofit.create(APIPublicAppeals::class.java)

val displayInfo = findViewById<Button>(R.id.displayInfo)

val editFirstField = findViewById<EditText>(R.id.editFirstField)

val editSecondField = findViewById<EditText>(R.id.editSecondField)

val layoutManager = LinearLayoutManager(this)

val AppealsRecyclerView = findViewById<RecyclerView>(R.id.AppealsList)

AppealsRecyclerView.layoutManager = layoutManager

AppealsRecyclerView.addItemDecoration(DividerItemDecoration(this, DividerItemDecoration.VERTICAL))

val db = Room.databaseBuilder(applicationContext, DB::class.java, "appeal").build()

val AppealsDao = db.AppealsDao()

GlobalScope.launch(Dispatchers.IO) {

val adapter = AppealsAdapter(

AppealsDao.getEverything().map { Appeals().apply { appealLocalString = it.appealString; appealLocalSymbol = it.appealSymbol } })

launch(Dispatchers.Main) {

AppealsRecyclerView.adapter = adapter

}

}

displayInfo.setOnClickListener {

val firstfield = editFirstFiled.text.toString()

val secondfield = editSecondField.text.toString()

if (editFirstField.text.toString() != "") {

if (Regex("""[A-Z]{20}""").matches(firstField)) {

if (secondField != "") {

if (Regex("""[A-Z]||[!]||[?]{1}""").matches(secondField)) { \\проуглить

publicAppeals.getAppeals(firstField, secondField).enqueue(object : Callback<MutableList<Appeals>> {

override fun onResponse(

call: Call<MutableList<Appeals>>,

response: Response<MutableList<Appeals>>

)

{

val result = response.body()

if (result != null) {

GlobalScope.launch(Dispatchers.IO) {

AppealsDao.dropAll()

AppealsDao.insertAll(

\*result.toList().map {

AppealsRoom(

it.appealLocalString!!,

it.appealLocalSymbol!!

)

}.toTypedArray()

)

}

val adapter = AppealsAdapter(result)

AppealsRecyclerView.adapter = adapter

}

}

override fun onFailure(

call: Call<MutableList<Appeals>>,

t: Throwable

) {

Toast.makeText(this@MainActivity, getString(R.string.retrofitError), Toast.LENGTH\_LONG).show()

}

})

} else Toast.makeText(this, getString(R.string.IncorrectSecondField), Toast.LENGTH\_LONG).show()

} else Toast.makeText(this, getString(R.string.EmptySecondField), Toast.LENGTH\_LONG).show()

} else Toast.makeText(this, getString(R.string.IncorrectFirstField), Toast.LENGTH\_LONG).show()

} else {

//if (editFirstField.text != null) {

// Toast.makeText(this, getString(R.string.IncorrectFirstField), Toast.LENGTH\_LONG).show()

//} else {

Toast.makeText(this, getString(R.string.EmptyFirstField), Toast.LENGTH\_LONG).show()

}

}

}

}