Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Институт математики и информационных систем

Факультет компьютерных и физико-математических наук

Кафедра прикладной математики и информатики

**О Т Ч Е Т**

о прохождении учебной практики

на базе кафедры прикладной математики и информатики

ФГБОУ ВО «Вятcкий государственный университет»

(наименование базы практики)

в период с \_08.02.2021\_ по **\_**02.05.2021**\_**

Выполнил

студент гр. ФИб-3302-51-00 Савин Д. А. \_\_\_\_\_\_\_\_\_\_

(шифр) (подпись) (Ф. И. О.) (дата)

Руководитель практики от вуза

к.ф.-м.н, доцент\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ Шалагинова Н.В. \_\_\_\_\_\_\_\_\_\_

(уч. степень, должность) (подпись) (Ф. И. О.) (дата)

Практика защищена

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Киров 2021

**Содержание**

[Введение 3](#_Toc70634595)

[Описание индивидуального задания 4](#_Toc70634596)

[Описание разработанного приложения 5](#_Toc70634597)

[Заключение 11](#_Toc70634598)

[Список литературы 12](#_Toc70634599)

[Приложение 1. Листинг программы 13](#_Toc70634600)

[MainActivity 13](#_Toc70634601)

[AI 15](#_Toc70634602)

[Advice 24](#_Toc70634603)

[AdviceAPI 24](#_Toc70634604)

[AdviceService 24](#_Toc70634605)

[AdviceToString 25](#_Toc70634606)

[DBHelper 25](#_Toc70634607)

[Forecast 26](#_Toc70634608)

[ForecastAPI 26](#_Toc70634609)

[ForecastService 26](#_Toc70634610)

[ForecastToString 27](#_Toc70634611)

[Message 27](#_Toc70634612)

[MessageEntity 28](#_Toc70634613)

[MessageListAdapter 28](#_Toc70634614)

[MessageViewHolder 29](#_Toc70634615)

[Number 29](#_Toc70634616)

[NumberAPI 30](#_Toc70634617)

[NumberService 30](#_Toc70634618)

[NumberToString 30](#_Toc70634619)

[ParsingHtmlService 31](#_Toc70634620)

# Введение

Учебная практика проходила в ФГБОУ ВО «Вятский государственный университет», на базе кафедры фундаментальной информатики и прикладной математики факультета компьютерных и физико-математических наук с 08.02.2021 г. по 02.05.2021 г., количество недель: 12. Установочная конференция состоялась 08.02.2021 г.

Целью учебной практики является закрепление и углубление теоретической подготовки, приобретение практических навыков и компетенций в сфере профессиональной деятельности, в основном путём самостоятельного решения предусмотренного программой индивидуального задания.

Форма проведения учебной практики: компьютерная (разработка мобильного приложения).

В ходе прохождения практики необходимо реализовать мобильное приложение «VoiceAssistent» для операционной системы Android и дополнить его новыми возможностями, используя данные web-ресурсов.

Язык программирования: kotlin.

# Описание индивидуального задания

1. Постановка задачи. Описание требований к функционалу приложения.
2. Описание выбранных сетевых ресурсов (api для получения данных, структуры html-страницы для парсинга).
3. Ссылка на систему контроля версий Git: https://github.com/DaniilSavin/VoiceAssistant

# Описание разработанного приложения

1. Руководство пользователя (описание запросов к голосовому помощнику).

Таблица 1.1. Таблица запросов пользователя и ответов помощзника

|  |  |
| --- | --- |
| Запрос Пользователя | Ответ Помощника/Варианты ответа |
| Привет, Hi, Hello, Здравствуйте | Привет, Здравствуйте |
| Что делаешь? | Отвечаю на вопросы :), Ожидаю. |
| Как дела? | Отлично :), Нормально. |
| Который час?, Время | Выводит текущее время в формате: hh.mm.ss |
| Какой день недели? | Выводит текущий день недели (Понедельник, Вторник и т. д.) |
| Какой сегодня день? | Выводит цифру – номер дня (Понедельник – 1, Вторник – 2 и т. д.) |
| Сколько дней до зачета? | * + - 1. Если есть сохраненная дата, то выводит количество времени (Год/месяц/дней) до этой даты.       2. Если нет сохраненной даты, то спрашивает: "Дата не найдена. Вы желаете добавить? (Y/N)". Если ответ: "y", то выводит: "Введите дату: дд.мм.гггг". Вводим дату по формату, ответ – количество времени (Год/месяц/дней) до этой даты.   Если ответ: "n", то запрос закрывается: "Хорошо. Закрываю вопрос." |
| Сколько дней до dd.mm.yyyy | Выводит количество времени (Год/месяц/дней) до этой даты. |
| Погода в городе Название города | Выводит количество градусов и информацию о погоде. |
| Число (например, 20) в строку | Переводит число из цифр в слово/слова. |
| Праздник вчера\сегодня\завтра\dd/mm/yyyy | Выводит название праздника в заданный день. |
| Совет | Вывод случайно выбранный совет. |
| Курс криптовалюты | Выводит названия криптовалют и стоимость в долларах. |

1. Описание реализации:

Работа с API проводилась с помощью сериализации, использовалась библиотека Retrofit2. Для каждого API было создано 4 класса (для примера API: <https://api.adviceslip.com/>):

Класс сущности API с определением необходимых тегов:

|  |
| --- |
| **class** **Advice** : Serializable {  @JvmField  @SerializedName("slip")  @Expose  var slip: adv? = **null**  inner **class** **adv** {  @JvmField  @SerializedName("advice")  @Expose  var advice: String? = **null**  }  } |

Класс с созданием соединения к сайту с помощью библиотеки Retrofit2:

|  |
| --- |
| object AdviceService {  @JvmStatic  val aPI: AdviceAPI  get() {  val retrofit = Retrofit.Builder()  .baseUrl("https://api.adviceslip.com/")  .addConverterFactory(GsonConverterFactory.create())  .build()  **return** retrofit.create(AdviceAPI::class.java)  }  } |

Класс с описанием конкретного Get запроса:

|  |
| --- |
| **interface** **AdviceAPI** {  @get:GET("/advice")  val advice: Call<Advice?>?  } |

Класс для непосредственного взаимодействия API и возвращения значений в основной код:

|  |
| --- |
| object AdviceToString {  @JvmStatic  fun getAdvice(number: String?, callback: Consumer<String?>) {  val api = aPI  val call = api.advice  val str = arrayOf<String?>("-1")  call!!.enqueue(object : Callback<Advice?> {  override fun onResponse(call: Call<Advice?>, response: Response<Advice?>) {  val result = response.body()  **if** (result != **null**) {  **if** (result.slip!!.advice != **null**) {  str[0] = result.slip!!.advice  callback.accept(str[0])  } **else** {  str[0] = "result.advice=null"  callback.accept(str[0])  }  } **else** {  str[0] = "result = null"  callback.accept(str[0])  }  }  override fun onFailure(call: Call<Advice?>, t: Throwable) {  Log.w("ADVICE", Objects.requireNonNull(t.message))  str[0] = "Не могу перевести failure"  }  })  }  } |

Для парсинга сетевого ресурса была использована библиотека Jsoup, с помощью которой создается соединение с сайтом, и далее с помощью css-запроса возвращаются необходимые элементы.

|  |
| --- |
| **private** **const** val URL2 = "https://ru.investing.com/crypto/"  @get:Throws(IOException::class)  val cryptoCurrencyExchangeRate: String  get() {  val document = Jsoup.connect(URL2).get()  val str = StringBuilder()  val elements = document.select("tr[i]")  **for** (element in elements) {  str.append(element.getElementsByClass("left bold elp name cryptoName first js-currency-name")  .select("a[href]").text())  .append(" ")  .append(element.getElementsByClass("left noWrap elp symb js-currency-symbol").text())  .append(" - ")  .append(element.getElementsByClass("price js-currency-price").select("a[href]").text())  .append("$\n")  }  **return** str.toString()  } |

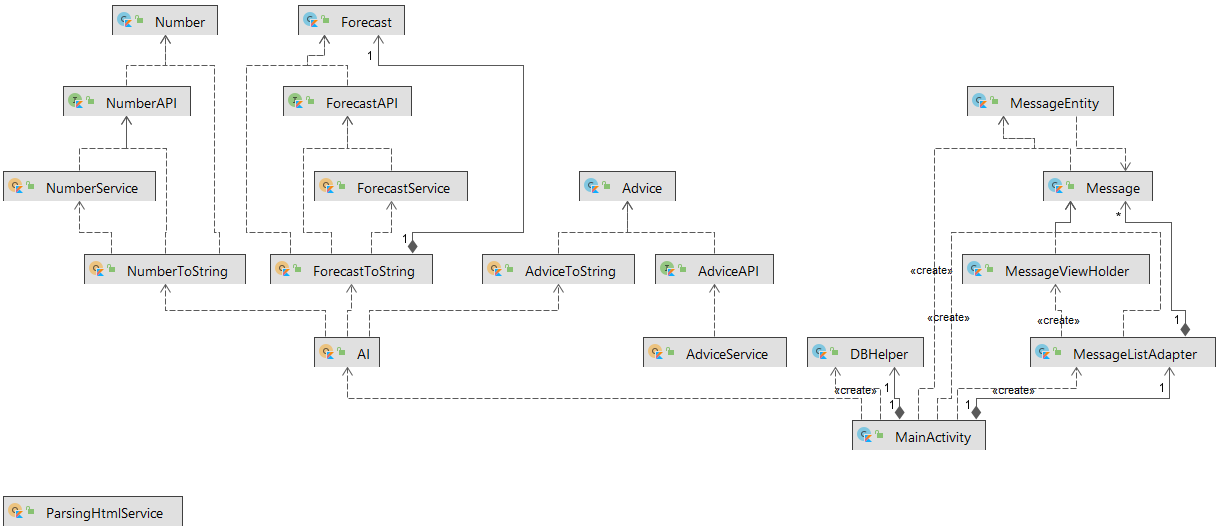


Рисунок 2.1. Диаграмма классов прниложения

Взаимодействие классов проекта отображено на диаграмме классов (рис. 2.1)

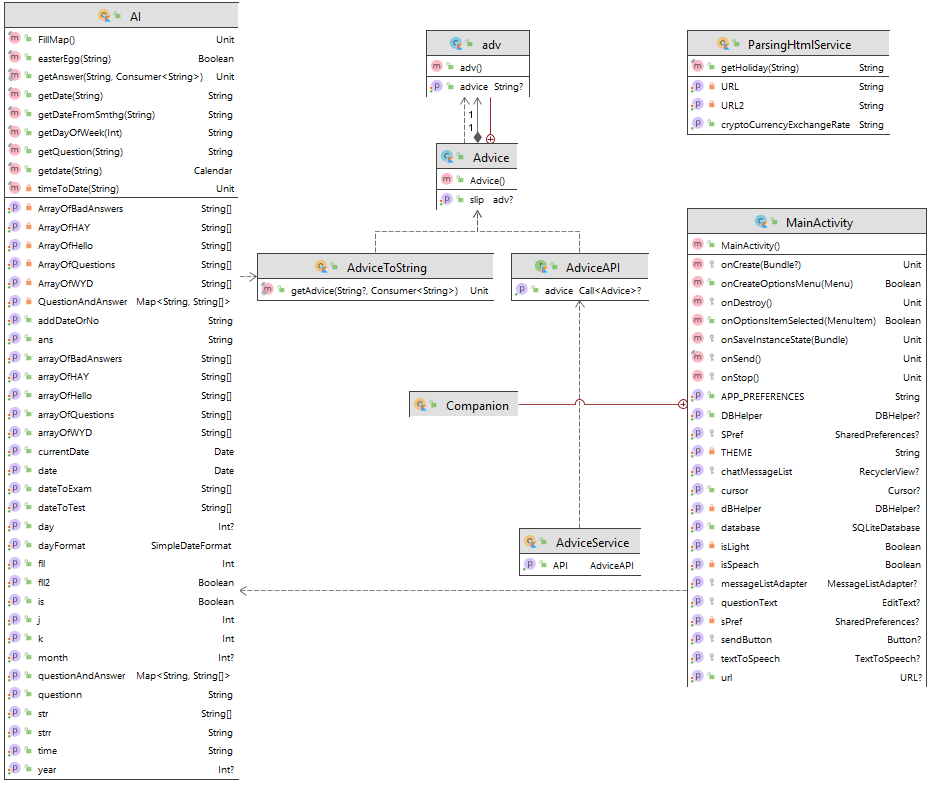


Рисунок 2.2. Диаграмма классов API «Advice»

Класс AI содержит обработку запроса пользователя и в зависимости от запроса, вызывает тот или иной метод, функция AI.getAnswer() вызывается в классе MainActivity.

Пример обработки запроса пользователя в классе AI:

|  |
| --- |
| **if** (questionn.toLowerCase().contains("совет")) {  AdviceToString.getAdvice(" ") { s: String? ->  **if** (s != **null**) {  callback.accept(s)  } **else** callback.accept("Нельзя")  }  **break**  } **else**  **if** (questionn.toLowerCase().contains("курс криптовалюты")) {  object : AsyncTask<String?, Int?, Void?>() {  **protected** override fun doInBackground(vararg params: String?): Void? {  **try** {  answers.add(ParsingHtmlService.cryptoCurrencyExchangeRate)  } **catch** (e: IOException) {  e.printStackTrace()  }  **return** **null**  }  override fun onPostExecute(aVoid: Void?) {  **super**.onPostExecute(aVoid)  callback.accept(java.lang.String.join(", ", answers))  }  }.execute()  **break**  } |

Вызов функции AI.getAnswer() в классе MainActivity:

|  |
| --- |
| @SuppressLint("NewApi", "StaticFieldLeak") *// @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)*  @Throws(ParseException::class, IOException::class)  **protected** fun onSend() {  val text = questionText!!.text.toString()  **if** (!StringUtils.isBlank(text)) {  AI.getAnswer(text) { s: String? ->  messageListAdapter!!.messageList.add(Message(text, **true**))  **if** (isSpeach) {  textToSpeech!!.speak(s, TextToSpeech.QUEUE\_FLUSH, **null**, **null**)  }  messageListAdapter!!.messageList.add(Message(s, **false**))  *// }*  messageListAdapter!!.notifyDataSetChanged()  chatMessageList!!.scrollToPosition(messageListAdapter!!.messageList.size - 1)  questionText!!.setText("")  }  }  } |

Диалог сохраняется при закрытии приложения, при сворачивании приложения, при смене темы и при повороте экрана. Для сохранения диалогов используется функция onSaveInstanceState():

|  |
| --- |
| override fun onSaveInstanceState(outState: Bundle) {  **super**.onSaveInstanceState(outState)  val s = messageListAdapter!!.messageList as ArrayList<Message>  outState.putSerializable("key", s)  } |

Условие, при котором идет восстановление диалога:

|  |
| --- |
| val str: ArrayList<Message>?  **if** (savedInstanceState != **null**) {  str = savedInstanceState.getSerializable("key") as ArrayList<Message>?  **for** (i in str!!.indices) {  messageListAdapter!!.messageList.add(str[i])  }  messageListAdapter!!.notifyDataSetChanged()  } |

Наиболее сложным моментом при переходе с Java на Kotlin оказался поиск информации по обращению к объектам на языке Kotlin.

1. Тестирование приложения.

Тестирование проводилось на мобильном устройстве с данными характеристиками:

Процессор –­ Qualcomm Snapdragon 660

Количество ОЗУ – 4 ГБ

Версия Android – 10 QKQ1.190910.002

# Заключение

Учебная практика способствовала закреплению и углублению теоретической подготовки, приобретению практических навыков и компетенций в сфере профессиональной деятельности путём реализации всех этапов разработки мобильного приложения.

В ходе практики было реализовано приложение «VoiceAssistent» для операционной системы Android, Версии 10 QKQ1.190910.002. Использовался язык программирования kotlin, среда разработки Android Studio. Листинг разработанного приложения приведен в Приложении 1.

Было выбрано API web-ресурса <https://api.adviceslip.com/>

При переносе приложения с языка java на язык kotlin наиболее сложным показалось разбираться в новом синтаксисе, потому что синтаксис kotlin достаточно сильно отличается от синтаксиса языка java.

# Список литературы

1. Филлипс Б. Android. Программирование для профессионалов. – Питре, 2019. – 688 с.
2. Android Studio [Электронный ресурс]: URL: <https://developer.android.com/studio/index.html> (дата обращения 01.02.2018)

# Приложение 1. Листинг программы

# MainActivity

package com.example.voiceassistent  
  
import android.annotation.SuppressLint  
import android.content.ContentValues  
import android.content.Context  
import android.content.SharedPreferences  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.graphics.Bitmap  
import android.os.Bundle  
import android.speech.tts.TextToSpeech  
import android.view.Menu  
import android.view.MenuItem  
import android.view.View  
import android.widget.Button  
import android.widget.EditText  
import androidx.appcompat.app.AppCompatActivity  
import androidx.appcompat.app.AppCompatDelegate  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import org.apache.commons.lang3.StringUtils  
import java.io.IOException  
import java.net.URL  
import java.text.ParseException  
import java.util.\*  
import com.example.voiceassistent.DBHelper  
import com.example.voiceassistent.DBHelper.Companion.*FIELD\_DATE*import com.example.voiceassistent.DBHelper.Companion.*FIELD\_MESSAGE*import com.example.voiceassistent.DBHelper.Companion.*FIELD\_SEND*import com.example.voiceassistent.DBHelper.Companion.*TABLE\_MESSAGES*import com.example.voiceassistent.MessageEntity  
import com.example.voiceassistent.Message  
import com.example.voiceassistent.MessageListAdapter  
  
import java.util.function.Consumer  
  
class MainActivity : AppCompatActivity() {  
 protected var sendButton: Button? = null  
 protected var questionText: EditText? = null  
 protected var textToSpeech: TextToSpeech? = null  
 protected var chatMessageList: RecyclerView? = null  
 protected var messageListAdapter: MessageListAdapter? = null  
 protected var sPref: SharedPreferences? = null  
 var url: URL? = null  
 private var isLight = true  
 private val isSpeach = false  
 private val THEME = "THEME"  
 var dBHelper: DBHelper? = null  
 lateinit var database: SQLiteDatabase  
 var cursor: Cursor? = null  
  
 *// @RequiresApi(api = Build.VERSION\_CODES.JELLY\_BEAN\_MR1)* override fun onCreate(savedInstanceState: Bundle?) {  
 sPref = getSharedPreferences(  
 APP\_PREFERENCES,  
 Context.*MODE\_PRIVATE* )  
 isLight = sPref?.getBoolean(THEME, true)!!  
 if (!isLight) *delegate*.*localNightMode* = AppCompatDelegate.*MODE\_NIGHT\_YES* super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 sendButton = findViewById(R.id.*sendButton*)  
 questionText = findViewById(R.id.*questionField*)  
 chatMessageList = findViewById(R.id.*chatMessageList*)  
 messageListAdapter = MessageListAdapter()  
  
 chatMessageList?.*layoutManager* = LinearLayoutManager(this)  
 chatMessageList?.*adapter* = messageListAdapter  
  
 dBHelper = DBHelper(this)  
 database = dBHelper!!.*writableDatabase* val str: ArrayList<Message>?  
 if (savedInstanceState != null) {  
 str = savedInstanceState.getSerializable("key") as ArrayList<Message>?  
 for (i in str!!.*indices*) {  
 messageListAdapter!!.messageList.add(str[i])  
 }  
 messageListAdapter!!.notifyDataSetChanged()  
 }  
 if (savedInstanceState == null) {  
 val cursor = database.query(DBHelper.TABLE\_MESSAGES, null, null, null, null, null, null)  
 if (cursor.moveToFirst()) {  
 val messageIndex = cursor.getColumnIndex(DBHelper.FIELD\_MESSAGE)  
 val dateIndex = cursor.getColumnIndex(DBHelper.FIELD\_DATE)  
 val sendIndex = cursor.getColumnIndex(DBHelper.FIELD\_SEND)  
 do {  
 val entity = MessageEntity(cursor.getString(messageIndex),  
 cursor.getString(dateIndex), cursor.getInt(sendIndex))  
 val message = Message(entity)  
 if (message.date == null || message.text == null || message.isSend == null) {  
 *//* } else {  
 messageListAdapter!!.messageList.add(message)  
 }  
 } while (cursor.moveToNext())  
 }  
 cursor.close()  
 }  
 textToSpeech = TextToSpeech(*applicationContext*) **{** i **->** if (isSpeach) {  
 if (i != TextToSpeech.*ERROR*) {  
 textToSpeech!!.*language* = Locale("ru")  
 }  
 }  
 **}** sendButton?.setOnClickListener(View.OnClickListener **{** try {  
 onSend()  
 } catch (e: ParseException) {  
 e.printStackTrace()  
 } catch (e: IOException) {  
 e.printStackTrace()  
 }  
 **}**)  
 }  
  
 override fun onSaveInstanceState(outState: Bundle) {  
 super.onSaveInstanceState(outState)  
 val s = messageListAdapter!!.messageList as ArrayList<Message>  
 outState.putSerializable("key", s)  
 }  
  
 override fun onCreateOptionsMenu(menu: Menu): Boolean {  
 *menuInflater*.inflate(R.menu.*menu\_main*, menu)  
 return super.onCreateOptionsMenu(menu)  
 }  
  
 override fun onDestroy() {  
 if (textToSpeech != null) {  
 textToSpeech!!.stop()  
 textToSpeech!!.shutdown()  
 }  
 database!!.close()  
 dBHelper!!.close()  
 super.onDestroy()  
 }  
  
 override fun onStop() {  
 val editor = sPref!!.edit()  
 editor.putBoolean(THEME, isLight)  
 editor.apply()  
 database!!.delete(DBHelper.TABLE\_MESSAGES, null, null)  
 for (i in messageListAdapter!!.messageList.*indices*) {  
 val entity = MessageEntity(messageListAdapter!!.messageList[i])  
 val contentValues = ContentValues()  
 contentValues.put(DBHelper.FIELD\_MESSAGE, entity.text)  
 contentValues.put(DBHelper.FIELD\_SEND, entity.isSend)  
 contentValues.put(DBHelper.FIELD\_DATE, entity.date)  
 database!!.insert(DBHelper.TABLE\_MESSAGES, null, contentValues)  
 }  
 super.onStop()  
 }  
  
 @SuppressLint("NonConstantResourceId") *// @RequiresApi(api = Build.VERSION\_CODES.JELLY\_BEAN\_MR1)* override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 when (item.*itemId*) {  
 R.id.*day\_settings* -> {  
 *delegate*.*localNightMode* = AppCompatDelegate.*MODE\_NIGHT\_NO* isLight = true  
 }  
 R.id.*night\_settings* -> {  
 *delegate*.*localNightMode* = AppCompatDelegate.*MODE\_NIGHT\_YES* isLight = false  
 }  
 R.id.*clear\_chat* -> {  
 messageListAdapter!!.messageList.clear()  
 super.recreate()  
 }  
 else -> {  
 }  
 }  
 return super.onOptionsItemSelected(item)  
 }  
  
  
 @SuppressLint("NewApi", "StaticFieldLeak") *// @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)* @Throws(ParseException::class, IOException::class)  
 protected fun onSend() {  
 val text = questionText!!.*text*.toString()  
 if (!StringUtils.isBlank(text)) {  
 AI.getAnswer(text) **{** s: String? **->** messageListAdapter!!.messageList.add(Message(text, true))  
 if (isSpeach) {  
 textToSpeech!!.speak(s, TextToSpeech.*QUEUE\_FLUSH*, null, null)  
 }  
 messageListAdapter!!.messageList.add(Message(s, false))  
  
 *// }* messageListAdapter!!.notifyDataSetChanged()  
 chatMessageList!!.scrollToPosition(messageListAdapter!!.messageList.size - 1)  
 questionText!!.setText("")  
 **}** }  
 }  
  
 companion object {  
 const val APP\_PREFERENCES = "mysettings"  
 }  
}

# AI

package com.example.voiceassistent  
  
import android.annotation.SuppressLint  
import android.os.AsyncTask  
import android.util.Log  
import java.io.IOException  
import java.text.DateFormat  
import java.text.ParseException  
import java.text.SimpleDateFormat  
import java.time.LocalDateTime  
import java.time.format.DateTimeFormatter  
import java.util.\*  
import java.util.concurrent.TimeUnit  
import java.util.regex.Pattern  
import java.util.function.Consumer  
  
object AI {  
 var time = "!"  
 var ans = ""  
 var `is` = false  
 var date = Date()  
 var currentDate = Date()  
 var dayFormat = SimpleDateFormat("dd/MM/YYYY")  
 var dateToTest = *arrayOf*(  
 "-1",  
 "-1",  
 "-1")  
 var dateToExam = *arrayOf*(  
 "-1",  
 "-1",  
 "-1")  
 var ArrayOfQuestions = *arrayOf*(  
 "совет",  
 "курс криптовалюты",  
 "в строку",  
 "погода в городе",  
 "привет",  
 "как дела",  
 "что делаешь",  
 "чем занимаешься",  
 "hello",  
 "hi",  
 "здравствуйте",  
 "который час",  
 "какой день недели",  
 "какой сегодня день",  
 "сколько дней до"  
 )  
 var ArrayOfBadAnswers = *arrayOf*(  
 "Ой, что?",  
 "Я не уверен..",  
 "Я не знаю.",  
 "Не понял вас..")  
 var ArrayOfHello = *arrayOf*(  
 "Привет",  
 "Здравствуйте")  
 var ArrayOfWYD = *arrayOf*(  
 "Отвечаю на вопросы :)",  
 "Ожидаю.")  
 var ArrayOfHAY = *arrayOf*(  
 "Отлично :)",  
 "Нормально.")  
 var QuestionAndAnswer: MutableMap<String, Array<String>> = HashMap()  
 fun FillMap() {  
 QuestionAndAnswer["Я еще не знаю этого :("] = ArrayOfBadAnswers  
 QuestionAndAnswer["привет"] = ArrayOfHello  
 QuestionAndAnswer["hello"] = ArrayOfHello  
 QuestionAndAnswer["hi"] = ArrayOfHello  
 QuestionAndAnswer["здравствуйте"] = ArrayOfHello  
 QuestionAndAnswer["что делаешь"] = ArrayOfWYD  
 QuestionAndAnswer["как дела"] = ArrayOfHAY  
 }  
  
 var year: Int? = null  
 var month: Int? = null  
 var day: Int? = null  
 var questionn = ""  
  
 *//public static Calendar t2;* @JvmStatic  
 @Throws(ParseException::class)  
 fun getAnswer(question: String, callback: Consumer<String?>) {  
  
 *// Форматирование времени как "день.месяц.год"* var question = question  
 val dateFormat: DateFormat = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())  
 val dateText = dateFormat.format(currentDate)  
 *// Форматирование времени как "часы:минуты:секунды"* val timeFormat: DateFormat = SimpleDateFormat("HH:mm:ss", Locale.getDefault())  
 val timeText = timeFormat.format(currentDate)  
 val c = Calendar.getInstance()  
 c.*firstDayOfWeek* = 2  
 c.*time* = currentDate  
 val dayOfWeek = c[Calendar.*DAY\_OF\_WEEK*]  
 FillMap()  
 var x = false  
 val answers = ArrayList<String>()  
 val answer: Array<String>  
 *//String ans="";* questionn = question.*toLowerCase*(Locale.*ROOT*)  
 while (true) {  
 if ((questionn == "y" || questionn == "n") && fll == 3) {  
 ans = getDateFromSmthg(questionn)  
 if (questionn == "y") {  
 fll = 4  
 } else {  
 fll = 0  
 fll2 = false  
 k = 0  
 *//questionn="exit";* j = 0  
 }  
 *//return ans;* callback.accept(ans)  
 x = true  
 break  
 }  
 if (fll == 4) {  
 *//return getDateFromSmthg(questionn);* timeToDate("сколько дней до " + questionn)  
 for (i in dateToTest.*indices*) {  
 if (dateToTest[i] == "-1") {  
 dateToTest[i] = questionn  
 break  
 }  
 }  
 questionn = "exit"  
 if (time == "0") {  
 callback.accept("Этот день уже наступил")  
 break  
 } else {  
 val n = time.*toInt*()  
 var day = ""  
 day = if (n % 10 == 1) {  
 " день"  
 } else {  
 if (n % 10 > 1 && n % 10 < 5) {  
 " дня"  
 } else {  
 " дней"  
 }  
 }  
 callback.accept("До этой даты " + time + day)  
 fll = 0  
 fll2 = false  
 k = 0  
 j = 0  
 break  
 }  
 }  
 fll = 0  
 fll2 = false  
 k = 0  
 j = 0  
 break  
 }  
 questionn = getQuestion(questionn)  
 var r:String = "[-+]?d+"  
  
 if (questionn.*matches*(r.*toRegex*())) {  
 if (time == "0") {  
 callback.accept("Этот день уже наступил")  
 *//break;* } else {  
 val n = time.*toInt*()  
 var day = ""  
 day = if (n % 10 == 1) {  
 " день"  
 } else {  
 if (n % 10 > 1 && n % 10 < 5) {  
 " дня"  
 } else {  
 " дней"  
 }  
 }  
 callback.accept("До этой даты " + time + day)  
 x = true  
 questionn = "exit"  
 }  
 }  
 val rand = Random()  
 if (!x) {  
 while (true)  
 {  
 if (!`is`) {  
 if (QuestionAndAnswer.containsKey(questionn)) {  
 answer = Objects.requireNonNull(QuestionAndAnswer[questionn])!!  
 ans = answer[rand.nextInt(answer.size)]  
 callback.accept(ans)  
 break  
 } else {  
 if (questionn.*contains*("который час") || question.*contains*("время")) {  
 currentDate = Date();  
 ans = timeFormat.format(currentDate)  
 callback.accept(ans)  
 break  
 } else {  
 if (questionn.*contains*("какой день недели")) {  
 ans = getDayOfWeek(dayOfWeek)  
 callback.accept(ans)  
 break  
 } else {  
 if (questionn.*contains*("какой сегодня день")) {  
 ans = Integer.toString(dayOfWeek - 1)  
 callback.accept(ans)  
 break  
 } else {  
 if (questionn.*contains*("сколько дней до")) {  
 if (!questionn.*contains*("зачет") && !questionn.*contains*("зачёт")) {  
 timeToDate(questionn)  
 }  
 if (time == "0") {  
 callback.accept("Этот день уже наступил")  
 break  
 } else {  
 val n = time.*toInt*()  
 var day = ""  
 day = if (n % 10 == 1) {  
 " день"  
 } else {  
 if (n % 10 > 1 && n % 10 < 5) {  
 " дня"  
 } else {  
 " дней"  
 }  
 }  
 callback.accept("До этой даты " + time + day)  
 break  
 }  
 } else {  
 if (questionn.*contains*("Дата не найдена. Вы желаете добавить? (Y/N)")) {  
 ans = questionn  
 fll = 3  
 callback.accept(ans)  
 *// questionn="";* break  
 } else {  
 if (questionn.*contains*("addOrNo")) {  
 ans = addDateOrNo  
 if (fll != 0) {  
 fll = 4  
 }  
 callback.accept(ans)  
  
 *// addDateOrNo=question.toLowerCase();  
 // questionn="";* break  
 } else {  
 if (questionn.*contains*("exit")) {  
 question = ""  
 questionn = ""  
 fll = 0  
 fll2 = false  
 k = 0  
 break  
 }  
 *// if (questionn.contains("погода в городе"))  
 run* **{** while (true) {  
 try {  
 val cityPattern = Pattern.compile(  
 "погода в городе (\\p{L}+)",  
 Pattern.*CASE\_INSENSITIVE* )  
 val numberPattern = Pattern.compile(  
 "(\\p{Digit}+) в строку",  
 Pattern.*CASE\_INSENSITIVE* )  
 var matcher = cityPattern.matcher(questionn)  
 if (matcher.find()) {  
 val cityName = matcher.group(1)  
 ForecastToString.getForecast(  
 cityName,  
 *Consumer*<String?> **{** s **->** if (s != null)  
 answers.add(s)  
 Log.i("WEATHER", "AI: $s")  
 callback.accept(  
 java.lang.String.join(  
 ", ",  
 answers  
 )  
 )  
 **}**)  
 break  
 }  
 matcher = numberPattern.matcher(questionn)  
 if (matcher.find()) {  
 val number = matcher.group(1)  
 NumberToString.getNumber(  
 number,  
 *Consumer*<String?> **{** s **->** if (s != null) {  
 answers.add(s)  
 }  
 Log.i("WEATHER", "AI: $s")  
 callback.accept(  
 java.lang.String.join(  
 ", ",  
 answers  
 )  
 )  
 **}**)  
 break  
 }  
 } catch (e: Exception) {  
 callback.accept("Не получается узнать :(")  
 break  
 }  
 if (questionn.*toLowerCase*().*contains*("праздник")) {  
 val finalText = question  
 val date = getDate(finalText)  
 object : AsyncTask<String?, Int?, Void?>() {  
 override fun doInBackground(vararg params: String?): Void? {  
 try {  
 answers.add(ParsingHtmlService.getHoliday(date))  
 } catch (e: IOException) {  
 e.printStackTrace()  
 }  
 return null  
 }  
  
 override fun onPostExecute(aVoid: Void?) {  
 super.onPostExecute(aVoid)  
 callback.accept(  
 java.lang.String.join(  
 ", ",  
 answers  
 )  
 )  
 }  
  
 }.execute(\*date.*split*(",").*toTypedArray*())  
 break  
 } else  
 if (questionn.*toLowerCase*().*contains*("совет")) {  
 AdviceToString.getAdvice(" ") **{** s: String? **->** if (s != null) {  
 callback.accept(s)  
 } else callback.accept("Нельзя")  
 **}** break  
 } else  
 if (questionn.*toLowerCase*()  
 .*contains*("курс криптовалюты")  
 ) {  
 object : AsyncTask<String?, Int?, Void?>() {  
 protected override fun doInBackground(vararg params: String?): Void? {  
 try {  
 answers.add(ParsingHtmlService.cryptoCurrencyExchangeRate)  
 } catch (e: IOException) {  
 e.printStackTrace()  
 }  
 return null  
 }  
  
 override fun onPostExecute(aVoid: Void?) {  
 super.onPostExecute(aVoid)  
 callback.accept(  
 java.lang.String.join(  
 ", ",  
 answers  
 )  
 )  
 }  
 }.execute()  
 break  
 } else  
 if (questionn.*toLowerCase*().*contains*("создатель")) {  
 callback.accept("SAVIN DANIIL")  
 break  
 } else {  
 callback.accept("Вопрос понял. Думаю...")  
 question = ""  
 questionn = ""  
 break  
 }  
 }  
 *// break* **}** }  
 }  
 }  
 }  
 }  
 }  
 }  
 } else {  
 callback.accept(ans)  
 break  
 }  
 break  
 }  
 }  
 }  
  
 private fun timeToDate(date: String) {  
 var date = date  
 val newDate: Array<String>  
 var temp = date.*replace*("сколько дней до ", "")  
 temp = temp.*replace*(" ", "")  
 temp = temp.*replace*("?", "")  
 newDate = temp.*split*("\\.").*toTypedArray*()  
 val calendar1: Calendar = GregorianCalendar()  
 val calendar2: Calendar = GregorianCalendar(newDate[2].*toInt*(), newDate[1].*toInt*() - 1, newDate[0].*toInt*())  
 val temp1 = calendar1.*timeInMillis* val temp2 = calendar2.*timeInMillis* val timeLeft: Long  
 if (calendar1.before(calendar2)) {  
 timeLeft = Math.abs(temp2 - temp1)  
 TimeUnit.*MILLISECONDS*.toDays(timeLeft)  
 time = (TimeUnit.*MILLISECONDS*.toDays(timeLeft) + 1).toString()  
 } else time = "0"  
 fll = 0  
 fll2 = false  
 j = 0  
 date = ""  
 }  
  
 fun getQuestion(text: String): String {  
 var i = 0  
 val fl = false  
 var question = text  
 if (!easterEgg(text)) {  
 while (i < ArrayOfQuestions.size) {  
 if (text.*contains*("сколько дней до")) {  
 if (text.*contains*("зачет") || text.*contains*("экзамен") || text.*contains*("зачёт")) {  
 question = getDateFromSmthg(text)  
 break  
 }  
 } else {  
 if (text.*contains*(ArrayOfQuestions[i]) */\*ArrayOfQuestions[i].contains(text)\*/* && (text.length >= ArrayOfQuestions[i].length || text.length == ArrayOfQuestions[i].length - 1)) {  
 question = if (!text.*contains*("погода в городе") && !text.*contains*("в строку")) {  
 ArrayOfQuestions[i]  
 } else {  
 text  
 }  
 break  
 }  
 }  
 i++  
 }  
 } else {  
 return ans  
 }  
 return question  
 }  
  
 var fll = 0  
 var addDateOrNo = ""  
 lateinit var str: Array<String>  
 var strr = ""  
 var j = 0  
 var k = 0  
 var fll2 = false  
 fun getDateFromSmthg(date: String): String {  
 if (fll == 0) {  
 str = date.*split*(" ").*toTypedArray*()  
 strr = str[str.size - 1]  
 }  
 if (strr.*contains*("зачет") || strr.*contains*("зачёт")) {  
 if (fll == 0) {  
 while (j < dateToTest.size) {  
 if (dateToTest[j] != "-1") {  
 *//Calendar tt= getdate(dateToTest[j]);  
 // ans=getDifference(tt);* timeToDate(dateToTest[j])  
 addDateOrNo = time  
 fll = 0  
 fll2 = false  
 k = 0  
 *//fll2=true; fll=4;* break  
 } else {  
 if (dateToTest[j] == "-1" && j == dateToTest.size - 1) {  
 fll = 1  
 }  
 }  
 j++  
 }  
 }  
 if (fll == 1) {  
 addDateOrNo = "Дата не найдена. Вы желаете добавить? (Y/N)"  
 }  
 if (fll == 3) {  
 if (date == "y") {  
 addDateOrNo = "Введите дату: дд.мм.гггг"  
 } else {  
 if (date == "n") {  
 addDateOrNo = "Хорошо. Закрываю вопрос."  
 questionn = "exit"  
 } else {  
 addDateOrNo = "Error"  
 }  
 *//questionn = "addOrNo";* fll = 0  
 }  
 }  
 if (fll == 4) {  
 if (!fll2) {  
 if (j != 0) {  
 dateToTest[j - 1] = date  
 } else {  
 dateToTest[j] = date  
 }  
 }  
 val tt = getdate(date)  
 *//addDateOrNo = getDifference(tt)* fll = 0  
 fll2 = false  
 j = 0  
 }  
 }  
 if (strr.*contains*("экзамен")) {  
 if (fll == 0) {  
 while (k < dateToExam.size) {  
 if (dateToExam[k] != "-1") {  
 *//addDateOrNo = getDate(dateToExam[k]).toString();  
 //Calendar tt= getdate(dateToExam[k]);  
 // ans=getDifference(tt);* fll = 0  
 fll2 = false  
 k = 0  
 *//fll2=true; fll=4;* break  
 } else {  
 if (dateToExam[k] == "-1" && k == dateToExam.size - 1) {  
 fll = 1  
 }  
 }  
 k++  
 }  
 }  
 if (fll == 1) {  
 addDateOrNo = "Дата не найдена. Вы желаете добавить? (Y/N)"  
 }  
 if (fll == 3) {  
 if (date == "y") {  
 addDateOrNo = "Введите дату через пробелы: дд мм гггг"  
 questionn = "addOrNo"  
 } else {  
 if (date == "n") {  
 addDateOrNo = "Хорошо. Закрываю вопрос."  
 } else {  
 addDateOrNo = "Error"  
 }  
 questionn = "addOrNo"  
 fll = 0  
 }  
 }  
 if (fll == 4) {  
 if (!fll2) {  
 if (k != 0) {  
 dateToExam[k - 1] = date  
 } else {  
 dateToExam[k] = date  
 }  
 }  
 val tt = getdate(date)  
 *//addDateOrNo = getDifference(tt)* fll = 0  
 fll2 = false  
 k = 0  
 }  
 }  
  
 *// Calendar t2 = Calendar.getInstance();  
 // t2.set(year, month - 1, day);* return addDateOrNo  
 }  
  
 @SuppressLint("SimpleDateFormat") *// @RequiresApi(api = Build.VERSION\_CODES.O)* @Throws(ParseException::class)  
 fun getDate(text: String): String {  
 val ldt: LocalDateTime  
 val format1 = DateTimeFormatter.ofPattern("dd/MM/yyyy", Locale.*ENGLISH*)  
 return if (text.*contains*("вчера")) {  
 ldt = LocalDateTime.now().minusDays(1)  
 format1.format(ldt)  
 } else {  
 if (text.*contains*("завтра")) {  
 ldt = LocalDateTime.now().plusDays(1)  
 format1.format(ldt)  
 } else {  
 if (text.*contains*("сегодня")) dayFormat.format(Date()) else {  
 val nextDate: String  
 dayFormat = SimpleDateFormat("dd/MM/yyyy")  
 val regex = "(0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.](19|20)\\d\\d"  
 var date1: Date? = null  
 val pattern = Pattern.compile(regex)  
 val matcher = pattern.matcher(text)  
 if (matcher.find()) date1 = SimpleDateFormat("dd/MM/yyyy").parse(matcher.group())  
 nextDate = dayFormat.format(date1)  
 nextDate  
 }  
 }  
 }  
 }  
  
 fun getdate(date1: String): Calendar {  
 val str = date1.*split*(" ").*toTypedArray*()  
 year = str[str.size - 1].*toInt*()  
 month = str[str.size - 2].*toInt*()  
 day = str[str.size - 3].*toInt*()  
 val t2 = Calendar.getInstance()  
 t2[year!!, month!! - 1] = day!!  
 return t2  
 }  
  
 fun getDayOfWeek(dayOfWeek: Int): String {  
 return if (dayOfWeek == 1) {  
 ans = "Воскресенье"  
 ans  
 } else {  
 if (dayOfWeek == 2) {  
 ans = "Понедельник"  
 ans  
 } else {  
 if (dayOfWeek == 3) {  
 ans = "Вторник"  
 ans  
 } else {  
 if (dayOfWeek == 4) {  
 ans = "Среда"  
 ans  
 } else {  
 if (dayOfWeek == 5) {  
 ans = "Четверг"  
 ans  
 } else {  
 if (dayOfWeek == 6) {  
 ans = "Пятница"  
 ans  
 } else {  
 if (dayOfWeek == 7) {  
 ans = "Суббота"  
 ans  
 } else {  
 ans = "Error 412"  
 ans  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
  
   
}

# Advice

package com.example.voiceassistent  
  
import com.google.gson.annotations.Expose  
import com.google.gson.annotations.SerializedName  
import java.io.Serializable  
  
class Advice : Serializable {  
 @JvmField  
 @SerializedName("slip")  
 @Expose  
 var slip: adv? = null  
  
 inner class adv {  
 @JvmField  
 @SerializedName("advice")  
 @Expose  
 var advice: String? = null  
 }  
}

# AdviceAPI

package com.example.voiceassistent  
  
import retrofit2.Call  
import retrofit2.http.GET  
  
interface AdviceAPI {  
 @get:GET("/advice")  
 val advice: Call<Advice?>?  
}

# AdviceService

package com.example.voiceassistent  
  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
  
object AdviceService {  
 @JvmStatic  
 val aPI: AdviceAPI  
 get() {  
 val retrofit = Retrofit.Builder()  
 .baseUrl("https://api.adviceslip.com/")  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
 return retrofit.create(AdviceAPI::class.*java*)  
 }  
}

# AdviceToString

package com.example.voiceassistent  
  
import android.util.Log  
import androidx.core.util.Consumer  
import com.example.voiceassistent.AdviceService.*aPI*import retrofit2.Call  
import retrofit2.Callback  
import retrofit2.Response  
import java.util.\*  
  
object AdviceToString {  
 @JvmStatic  
 fun getAdvice(number: String?, callback: Consumer<String?>) {  
 val api = *aPI* val call = api.advice  
 val str = *arrayOf*<String?>("-1")  
 call!!.enqueue(object : Callback<Advice?> {  
 override fun onResponse(call: Call<Advice?>, response: Response<Advice?>) {  
 val result = response.body()  
 if (result != null) {  
 if (result.slip!!.advice != null) {  
 str[0] = result.slip!!.advice  
 callback.accept(str[0])  
 } else {  
 str[0] = "result.advice=null"  
 callback.accept(str[0])  
 }  
 *//callback.accept(String.valueOf(newStr));* } else {  
 str[0] = "result = null"  
 callback.accept(str[0])  
 }  
 }  
  
 override fun onFailure(call: Call<Advice?>, t: Throwable) {  
 Log.w("ADVICE", Objects.requireNonNull(t.message))  
 str[0] = "Не могу перевести failure"  
 }  
 })  
 }  
}

# DBHelper

package com.example.voiceassistent  
  
import android.content.Context  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
  
class DBHelper(context: Context?) :  
 SQLiteOpenHelper(  
 context,  
 DATABASE\_NAME,  
 null,  
 DATABASE\_VERSION  
 ) {  
 override fun onCreate(db: SQLiteDatabase) {  
 db.execSQL(  
 "create table " + TABLE\_MESSAGES + "(" +  
 FIELD\_ID + " integer primary key," +  
 FIELD\_MESSAGE + " text," +  
 FIELD\_SEND + " integer," +  
 FIELD\_DATE + " text" + ")"  
 )  
 }  
  
 override fun onUpgrade(  
 db: SQLiteDatabase,  
 oldVersion: Int,  
 newVersion: Int  
 ) {  
 db.execSQL("drop table if exists $TABLE\_MESSAGES")  
 onCreate(db)  
 }  
  
 companion object {  
 const val DATABASE\_VERSION = 1  
 const val DATABASE\_NAME = "messageDb"  
 const val TABLE\_MESSAGES = "messages"  
 const val FIELD\_ID = "id"  
 const val FIELD\_MESSAGE = "message"  
 const val FIELD\_SEND = "send"  
 const val FIELD\_DATE = "date"  
 }  
}

# Forecast

package com.example.voiceassistent  
  
import com.google.gson.annotations.Expose  
import com.google.gson.annotations.SerializedName  
import java.io.Serializable  
  
class Forecast : Serializable {  
 @SerializedName("current")  
 @Expose  
 var current: Weather? = null  
  
 inner class Weather {  
  
 @SerializedName("temperature")  
 @Expose  
 var temperature: Int? = null  
  
 @SerializedName("weather\_descriptions")  
 @Expose  
 var weather\_descriptions: List<String>? = null  
 }  
}

# ForecastAPI

package com.example.voiceassistent  
  
import retrofit2.http.GET  
import retrofit2.http.Query  
  
interface ForecastAPI {  
 @GET("/current?access\_key=f853c8a905f351f18c211114ddac44f1")  
 open fun getCurrentWeather(@Query("query") city: String?): retrofit2.Call<Forecast?>?  
}  
  
*//f853c8a905f351f18c211114ddac44f1*

# ForecastService

package com.example.voiceassistent  
  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
  
  
object ForecastService {  
 val getApi: ForecastAPI  
 get() {  
 val retrofit = Retrofit.Builder()  
 .baseUrl("http://api.weatherstack.com")  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
 return retrofit.create(ForecastAPI::class.*java*)  
 }  
}

# ForecastToString

package com.example.voiceassistent  
  
import android.os.Build  
import android.util.Log  
import androidx.annotation.RequiresApi  
import retrofit2.Call  
import retrofit2.Callback  
import retrofit2.Response  
import java.util.function.Consumer  
  
object ForecastToString {  
 var result: Forecast? = null  
 var answer: String? = ""  
  
 fun getForecast(city: String?, callback: Consumer<String?>) {  
 val api: ForecastAPI = ForecastService.getApi  
 val call: Call<Forecast?>? = api.getCurrentWeather(city)  
 call!!.enqueue(object : Callback<Forecast?> {  
 override fun onResponse(  
 call: Call<Forecast?>?,  
 response: Response<Forecast?>?  
 ) {  
 result = response!!.body()  
 if (result != null) {  
 answer =  
 "сейчас где-то " + result!!.current!!.temperature.*toString*() + degrees(  
 result!!.current!!.temperature!!  
 ).*toString*() + " и " + result!!.current!!.weather\_descriptions!![0]  
 Log.i("WEATHER", "ForecastToString: $answer")  
 callback!!.accept(answer!!)  
 } else callback!!.accept("Не могу узнать погоду")  
 Log.i("WEATHER", "ForecastToString: " + "Не могу узнать погоду")  
 }  
  
 override fun onFailure(  
 call: Call<Forecast?>?,  
 t: Throwable?  
 ) {  
 Log.w("WEATHER", t!!.message!!)  
 }  
 })  
 }  
  
 private fun degrees(x: Int): String? {  
 var x = x  
 x = Math.abs(x)  
 return if (x % 10 == 1 && x != 11 && x % 100 != 11) " градус"  
 else  
 if (x % 10 == 2 && x != 12 && x % 100 != 12 || x % 10 == 3 && x != 13 && x % 100 != 13  
 || x % 10 == 4 && x != 14 && x % 100 != 14  
 ) " градуса"  
 else " градусов"  
 }  
}

# Message

package com.example.voiceassistent  
  
import android.graphics.Bitmap  
import java.util.\*  
  
class Message {  
 @JvmField  
 var text: String? = null  
 @JvmField  
 var date: Date? = null  
 @JvmField  
 var isSend: Boolean  
 var str: ArrayList<String>? = null  
 var message: Message? = null  
 var image: Bitmap? = null  
  
 constructor(text: String?, isSend: Boolean) {  
 this.text = text  
 this.isSend = isSend  
 date = Date()  
 }  
  
 constructor(str: ArrayList<String>?, isSend: Boolean) {  
 this.str = str  
 this.isSend = isSend  
 date = Date()  
 }  
  
 constructor(entity: MessageEntity) {  
 text = entity.text  
 if (entity.date == null) date = null else date = Date(entity.date)  
 isSend = entity.isSend != 0  
 }  
  
 constructor(image: Bitmap?, isSend: Boolean) {  
 this.image = image  
 this.isSend = isSend  
 date = Date()  
 }  
}

# MessageEntity

package com.example.voiceassistent  
  
class MessageEntity {  
 var text: String?  
 var date: String  
 var isSend: Int  
  
 constructor(text: String?, date: String, isSend: Int) {  
 this.text = text  
 this.date = date  
 this.isSend = isSend  
 }  
  
 constructor(message: Message) {  
 text = message.text  
 date = message.date.*toString*()  
 isSend = if (!message.isSend) 0 else 1  
 }  
}

# MessageListAdapter

package com.example.voiceassistent  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.recyclerview.widget.RecyclerView  
import java.util.\*  
import com.example.voiceassistent.R  
  
  
class MessageListAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>() {  
 var messageList: ArrayList<Message> = ArrayList()  
  
  
 override fun onCreateViewHolder(  
 parent: ViewGroup,  
 viewType: Int  
 ): RecyclerView.ViewHolder {  
 val view: View  
 return if (viewType == USER\_TYPE) {  
 view = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*user\_message*, parent, false)  
 MessageViewHolder(view)  
 } else {  
 view = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*assistant\_message*, parent, false)  
 MessageViewHolder(view)  
 }  
 }  
  
 override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {  
 MessageViewHolder(holder.itemView).bind(messageList[position])  
 }  
 override fun getItemCount(): Int {  
 return messageList.size  
  
 }  
  
 override fun getItemViewType(index: Int): Int {  
 var message = messageList[index]  
 return if (message.isSend) {  
 USER\_TYPE  
 } else ASSISTANT\_TYPE  
 }  
  
 companion object {  
 private const val ASSISTANT\_TYPE = 0  
 private const val USER\_TYPE = 1  
 }  
  
}

# MessageViewHolder

package com.example.voiceassistent  
  
import android.annotation.SuppressLint  
import android.view.View  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView.ViewHolder  
import java.text.DateFormat  
import java.text.SimpleDateFormat  
  
class MessageViewHolder(itemView: View) : ViewHolder(itemView) {  
 protected var messageText: TextView  
 protected var messageDate: TextView  
 fun bind(message: Message) {  
 messageText.*text* = message.text  
 @SuppressLint("SimpleDateFormat") val fmt: DateFormat = SimpleDateFormat("HH:mm")  
 messageDate.*text* = fmt.format(message.date)  
 }  
  
 init {  
 messageText = itemView.findViewById(R.id.*messageTextView*)  
 messageDate = itemView.findViewById(R.id.*messageDateView*)  
 }  
}

# Number

package com.example.voiceassistent  
  
import com.google.gson.annotations.SerializedName  
import com.google.gson.annotations.Expose  
import java.io.Serializable  
  
class Number : Serializable {  
 @SerializedName("str")  
 @Expose  
 var number: String? = null  
}

# NumberAPI

package com.example.voiceassistent  
  
import retrofit2.Call  
import retrofit2.http.GET  
import retrofit2.http.Query  
  
interface NumberAPI {  
 @GET("/json/convert/num2str")  
 fun getNumber(@Query("num") number: String?): Call<Number?>?  
}

# NumberService

package com.example.voiceassistent  
  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
  
object NumberService {  
 val api: NumberAPI  
 get() {  
 val retrofit = Retrofit.Builder()  
 .baseUrl("https://htmlweb.ru") *//Базовая часть адреса* .addConverterFactory(GsonConverterFactory.create()) *//Конвертер, необходимый для преобразования JSON'а в объекты* .build()  
 return retrofit.create(NumberAPI::class.*java*) *//Создание объекта, при помощи которого будут выполняться запросы* }  
}

# NumberToString

package com.example.voiceassistent  
  
import android.util.Log  
import java.util.function.Consumer  
import retrofit2.Call  
import retrofit2.Callback  
import retrofit2.Response  
import java.util.\*  
  
object NumberToString {  
 fun getNumber(number: String?, callback: Consumer<String?>) {  
 val api: NumberAPI = NumberService.api  
 val call: Call<Number?>? = api.getNumber(number)  
 call!!.enqueue(object : Callback<Number?> {  
 override fun onResponse(call: Call<Number?>, response: Response<Number?>) {  
 val result = response.body()  
 if (result != null) {  
 if (result.number != null) {  
 val str = result.number  
 val newStr = StringBuilder()  
 val strr = str!!.*split*(" ").*toTypedArray*()  
 var i = 0  
 while (i < strr.size) {  
 if (!strr[i].*contains*("руб")) {  
 newStr.append(strr[i]).append(" ")  
 } else {  
 if (strr[i].*contains*("руб")) {  
 i = str.length - 1  
 }  
 }  
 i++  
 }  
 callback.accept(newStr.toString())  
 } else callback.accept("Не могу перевести result.str")  
 } else callback.accept("Не могу перевести result")  
 }  
  
 override fun onFailure(call: Call<Number?>, t: Throwable) {  
 Log.w("NUMBER", Objects.requireNonNull(t.message))  
 callback.accept("Не могу перевести failure")  
 }  
 })  
 }  
}

# ParsingHtmlService

package com.example.voiceassistent  
  
import org.jsoup.Jsoup  
import java.io.IOException  
import java.util.\*  
  
object ParsingHtmlService {  
 private const val URL = "http://mirkosmosa.ru/holiday/2021"  
 private const val URL2 = "https://ru.investing.com/crypto/"  
 @Throws(IOException::class)  
 fun getHoliday(date: String): String {  
 val reg = Regex("[/.\\s+]")  
 val sParts = date.*split*(reg).*toTypedArray*()  
 val iParts = IntArray(sParts.size)  
 for (i in sParts.*indices*) {  
 iParts[i] = sParts[i].*toInt*()  
 }  
 val document = Jsoup.connect(URL).get()  
 val div = document.select("#holiday\_calend > div:nth-child(" + iParts[1] + ")>div>div:nth-child("  
 + iParts[0] + ")" + " > div.month\_cel\_date + div.month\_cel > ul.holiday\_month\_day\_holiday > li > a[href]")  
 val str: MutableList<String> = ArrayList()  
 for (e in div) {  
 str.add(e.text())  
 }  
 return if (str.size != 0) {  
 str.toString().*replace*("\\[|\\]".*toRegex*(), "").*replace*(",", "\n")  
 } else {  
 "Праздника нет"  
 }  
 }  
  
 @get:Throws(IOException::class)  
 val cryptoCurrencyExchangeRate: String  
 get() {  
 val document = Jsoup.connect(URL2).get()  
 val str = StringBuilder()  
 val elements = document.select("tr[i]")  
 for (element in elements) {  
 str.append(element.getElementsByClass("left bold elp name cryptoName first js-currency-name")  
 .select("a[href]").text())  
 .append(" ")  
 .append(element.getElementsByClass("left noWrap elp symb js-currency-symbol").text())  
 .append(" - ")  
 .append(element.getElementsByClass("price js-currency-price").select("a[href]").text())  
 .append("$\n")  
 }  
 return str.toString()  
 }  
}