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 \_\_\_\_\_\_\_\_\_\_\_ Шатунов Д.И. \_\_\_\_\_\_\_\_\_\_\_\_

(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](#_Toc69900974)

[Описание индивидуального задания 4](#_Toc69900975)

[Описание разработанного приложения 5](#_Toc69900976)

[Заключение 6](#_Toc69900977)

[Список литературы 7](#_Toc69900978)

[Приложение 1. Листинг программы 8](#_Toc69900979)

# 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.

# Описание разработанного приложения

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

|  |  |
| --- | --- |
| Пользователь | Помощник |
| Привет | Привет |
| Как дела? | Неплохо |
| Что делаешь? | Отвечаю на вопросы |
| Дата | Выводит текущую дату |
| Время | Выводит текущее время |
| Количество дней между X и Y | Выводит полное количество дней между днем X и днем Y |
| День недели | Выводит текущий день недели |
| Курс валюты | Выводит курс евро и доллара по отношению к рублю на текущий момент |
| Ковид в стране X | Выводит количество заболевших, умерших и выздоровевших на текущий момент в стране X |
| Фильмы X | Выводит 5 лучших фильмов за год X |

1. Описание реализации:

Работа с api проводилась с помощью сериализации, использовалась библиотека Retrofit2.

Для каждого api было создано 4 класса (для примера api <https://mmediagroup.fr/covid-19> ):

1.Класс сущности api с определением необходимых тегов

**class** Covid : Serializable {  
 @SerializedName(**"All"**)  
 @Expose  
 **var** All: Covid2? = **null** inner **class** Covid2 {  
 @SerializedName(**"confirmed"**)  
 @Expose  
 **var** confirmed: Int? = **null** @SerializedName(**"deaths"**)  
 @Expose  
 **var** deaths: Int? = **null** @SerializedName(**"recovered"**)  
 @Expose  
 **var** recovered: Int? = **null** }  
}

2.Класс с созданием соединения к сайту с помощью библиотеки Retrofit2.

**object** CovidService {  
 **val** apiCovid: CovidApi  
 **get**() {  
 **val** retrofit = Retrofit.Builder()  
 .baseUrl(**"https://covid-api.mmediagroup.fr"**)  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
 **return** retrofit.create(CovidApi::**class**.java)  
 }  
}

3.Класс с описанием конкретного Get запроса

**interface** CovidApi {  
 @GET(**"/v1/cases"**)  
 **fun** getCurrentCountry(@Query(**"country"**) country: String?): Call<Covid?>?  
}

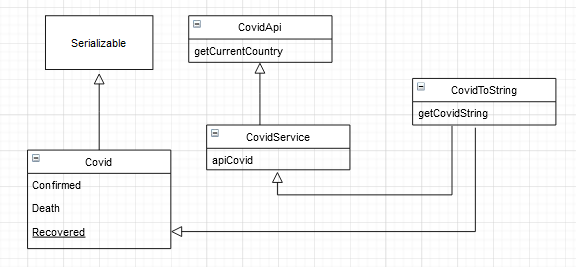
4.Класс для непосредственного взаимодействия api и возвращения значений в основной код.

**object** CovidToString {  
 @RequiresApi(api = Build.VERSION\_CODES.*N*)  
 **var result2**: Covid? = **null  
 var answer** = **""** @RequiresApi(api = Build.VERSION\_CODES.*N*)  
 **fun** getCovidString(country: String?, callback: Consumer<String?>) {  
 **val** api2: CovidApi = CovidService.**apiCovid  
 val** call = api2.getCurrentCountry(country)  
 call!!.enqueue(**object** : Callback<Covid?> {  
 **override fun** onResponse(call: Call<Covid?>, response: Response<Covid?>) {  
 **result2** = response.body()  
 **answer** += **"""  
 Подтвержденные случаи: ${result2**!!.**All**!!.**confirmed**.*toString*()**}  
 Количество смертей: ${result2**!!.**All**!!.**deaths**.*toString*()**}  
 Количество выздоровленных: ${result2**!!.**All**!!.**recovered}  
 """**.*trimIndent*()  
 callback.accept(**answer**)  
 **answer** = **""** }  
  
 **override fun** onFailure(call: Call<Covid?>, t: Throwable) {  
 callback.accept(**"Не могу получить данные"**)  
 }  
 })  
 }  
}

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 URL** = **"https://www.film.ru/online/year/"** @Throws(IOException::**class**)  
  
 **fun** getFilm(date: String): String  
 {  
 **val** reg = Regex(**"[\\s+]"**)  
 **var** div: Elements? = **null  
 val** sParts = date.*split*(reg).*toTypedArray*()  
 **val** iParts = *arrayOfNulls*<String>(sParts.**size**)  
 **for** (i **in** sParts.*indices*)  
 iParts[i] = sParts[i]  
 **val** document = Jsoup.connect(**URL** + iParts[1]).get()  
 **val** str: MutableList<String> = ArrayList()  
 **for** (i **in** 1..5) {  
 div = document.select(**"#movies\_list > div:nth-child(1) > a:nth-child($**i**) > strong"**)  
 **if** (div.isEmpty())  
 **break  
 else** str.add(**"""${**div.text()**}."""**.*trimIndent*()  
 )  
 }  
 **return if** (str.**size** != 0) str.toString().*replace*(**"^\\[|\\]$"**.*toRegex*(), **""**).*replace*(**","**, **""**)  
 **else  
 "Нет данных по этому году"** }  
}

Диаграмма классов для api <https://mmediagroup.fr/covid-19>



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

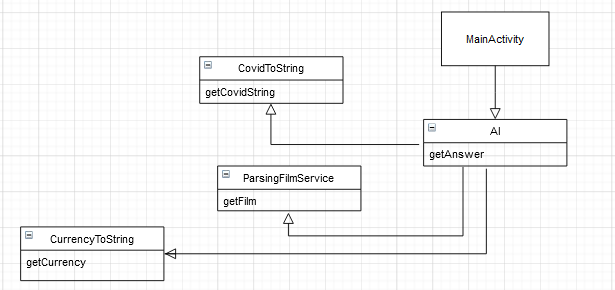
Пример обработки запроса пользователя в классе AI

if (text.*contains*("ковид в стране"))  
{  
 val numPattern = Pattern.compile("ковид в стране (\\p{L}+)", Pattern.*CASE\_INSENSITIVE*)  
 val matcher = numPattern.matcher(text)  
 if (matcher.find())  
 {  
 val country = matcher.group(1)  
 CovidToString.getCovidString(AI.firstUpperCase(country), *Consumer*<String?> **{** s **->** if (s != null) {  
 answers.add(s)  
 }  
 callback.accept(java.lang.String.join(", ", answers))  
 **}**)  
 }  
}

Вызов функции getAnswer в классе MainActivity

protected open fun onSend() {  
 val text = questionText!!.text.toString()  
 AI.getAnswer(text, Consumer<String> { s ->  
 Log.i("WEATHER", "MA: $s")  
 textToSpeech!!.speak(s, TextToSpeech.QUEUE\_FLUSH, null, null)  
 messageListAdapter!!.messageList.add(Message(text, true))  
 messageListAdapter!!.messageList.add(Message(s, false))  
 messageListAdapter!!.notifyDataSetChanged()  
 chatMessageList!!.scrollToPosition(messageListAdapter!!.messageList.size - 1)  
 questionText!!.setText("")  
 })  
}

Диаграмма взаимодействия класса AI с другими классами



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)  
}

Для восстановления диалогов используется условие при котором savedInstanceState != null

**var** 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 языка Java на kotlin оказалась работа с синтаксисом языка.

Тестирование приложения проводилось на мобильном устройстве с процессором snapdragon 835, операционной системой android 10 и ОЗУ 6 гб.

# 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 10 версии. Использовался язык программирования kotlin, среда разработки Android Studio. Листинг разработанного приложения приведен в Приложении 1.

Было выбрано API web-ресурса <https://mmediagroup.fr/covid-19>, <https://fixer.io/>

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)

3. Сайт Александра Климова [Электронный ресурс]: URL: <http://developer.alexanderklimov.ru/android/> (дата обращения 05.04.2021)

# Приложение 1. Листинг программы

# MainActivity

package com.example.kotlinapp

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.os.Build

import android.os.Bundle

import android.speech.tts.TextToSpeech

import android.speech.tts.TextToSpeech.OnInitListener

import android.util.Log

import android.view.Menu

import android.view.MenuItem

import android.view.View

import android.widget.Button

import android.widget.EditText

import androidx.annotation.RequiresApi

import androidx.appcompat.app.AppCompatActivity

import androidx.appcompat.app.AppCompatDelegate

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.example.kotlinapp.Database.DBHelper

import com.example.kotlinapp.Database.DBHelper.Companion.FIELD\_DATE

import com.example.kotlinapp.Database.DBHelper.Companion.FIELD\_MESSAGE

import com.example.kotlinapp.Database.DBHelper.Companion.FIELD\_SEND

import com.example.kotlinapp.Database.DBHelper.Companion.TABLE\_MESSAGES

import com.example.kotlinapp.Database.MessageEntity

import com.example.kotlinapp.MessageService.Message

import com.example.kotlinapp.MessageService.MessageListAdapter

import java.io.IOException

import java.text.ParseException

import java.util.\*

import java.util.function.Consumer

open 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

private var isLight = true

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 == false) 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

var str: ArrayList<Message>

if (savedInstanceState != null) {

str = (savedInstanceState.getSerializable("key") as ArrayList<Message>?)!!

for (i in str!!.indices) {

if (i % 2 == 0)

messageListAdapter!!.messageList.add(str[i])

if (i % 2 != 0)

messageListAdapter!!.messageList.add(str[i])

}

messageListAdapter!!.notifyDataSetChanged()

}

if (savedInstanceState == null) {

val cursor = database.query(TABLE\_MESSAGES, null, null, null, null, null, null)

if (cursor.moveToFirst()) {

val messageIndex = cursor.getColumnIndex(FIELD\_MESSAGE)

val dateIndex = cursor.getColumnIndex(FIELD\_DATE)

val sendIndex = cursor.getColumnIndex(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,

OnInitListener { i ->

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(TABLE\_MESSAGES, null, null)

for (i in 0 until messageListAdapter!!.messageList.size) {

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(TABLE\_MESSAGES, null, contentValues)

}

super.onStop()

}

@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\_dialog -> {

messageListAdapter!!.messageList.clear();

}

else -> {

}

}

return super.onOptionsItemSelected(item)

}

@SuppressLint("NewApi")

@RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)

@Throws(

ParseException::class,

IOException::class

)

protected open fun onSend() {

val text = questionText!!.text.toString()

AI.getAnswer(text, Consumer<String> { s ->

Log.i("WEATHER", "MA: $s")

textToSpeech!!.speak(s, TextToSpeech.QUEUE\_FLUSH, null, null)

messageListAdapter!!.messageList.add(Message(text, true))

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.kotlinapp

import android.annotation.SuppressLint

import android.os.AsyncTask

import android.os.Build

import android.util.Log

import androidx.annotation.RequiresApi

import com.example.kotlinapp.CovidAPI.CovidToString

import com.example.kotlinapp.CurrencyAPI.CurrencyToString

import com.example.kotlinapp.NumberConversionAPI.NumberToString

import com.example.kotlinapp.ParsingService.ParsingFilmService

import com.example.kotlinapp.ParsingService.ParsingHtmlService

import com.example.kotlinapp.WeatherAPI.ForecastToString

import java.io.IOException

import java.text.ParseException

import java.text.SimpleDateFormat

import java.time.LocalDateTime

import java.time.format.DateTimeFormatter

import java.util.\*

import java.util.function.Consumer

import java.util.regex.Pattern

object AI {

var hashMap: MutableMap<String, String> =

HashMap()

var dayFormat = SimpleDateFormat("dd/MM/YYYY")

var currDate: String = dayFormat.format(Date())

var timeFormat = SimpleDateFormat("HH:mm:ss")

var currTime = timeFormat.format(Date())

var dayFormat2 = SimpleDateFormat("EEEE")

/\*fun DictionaryFill(hashMap: MutableMap<\*, \*>) {

hashMap["привет"] = "Привет!"

hashMap["как дела"] = "Неплохо"

hashMap["чем занимаешься"] = "Отвечаю на вопросы"

}\*/

@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 firstUpperCase(word: String?): String? {

return if (word == null || word.isEmpty()) "" else word.substring(0, 1)

.toUpperCase() + word.substring(1) //или return word;

}

@SuppressLint("NewApi")

@RequiresApi(api = Build.VERSION\_CODES.N)

@Throws(ParseException::class, IOException::class)

fun getAnswer(text: String, callback: Consumer<String>) {

var text = text

//AI.DictionaryFill(hashMap)

val result: String? = null

val answers = ArrayList<String>()

val entries = ArrayList<Map.Entry<\*, \*>>(hashMap.entries)

text = text.toLowerCase()

/\* for (HashMap.Entry entry : entries)

{\*/if (text.contains("сколько времени") || text.contains("время")) {

timeFormat = SimpleDateFormat("HH:mm:ss")

currTime = timeFormat.format(Date())

answers.add("Сейчас - " + currTime)

callback.accept(java.lang.String.join(", ", answers /\*hashMap.get(entry.getKey()))\*/))

}

if (text.contains("какой сегодня день") || text.contains("дата")) {

dayFormat = SimpleDateFormat("dd/MM/YYYY")

currDate = dayFormat.format(Date())

answers.add("Сегодня - " + currDate)

callback.accept(java.lang.String.join(", ", answers /\*hashMap.get(entry.getKey()))\*/))

}

if (text.contains("какой день недели") || text.contains("день недели")) {

dayFormat2 = SimpleDateFormat("EEEE")

val calendar: Calendar = GregorianCalendar(TimeZone.getTimeZone(currDate))

answers.add("Сегодня - " + dayFormat2.format(calendar.time))

callback.accept(java.lang.String.join(", ", answers /\*hashMap.get(entry.getKey()))\*/))

}

if (text.contains("сколько дней до") || text.contains("дней до") || text.contains("сколько до")) {

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

var date2: Date? = null

val pattern = Pattern.compile(regex)

val matcher = pattern.matcher(text)

date2 = SimpleDateFormat("dd/MM/yyyy").parse(currDate)

if (matcher.find()) {

date1 = SimpleDateFormat("dd/MM/yyyy").parse(matcher.group())

} else answers.add("Дата не найдена")

try {

} catch (e: Exception) {

e.printStackTrace()

}

val difference = date1!!.time - date2.time

val days = difference / (24 \* 60 \* 60 \* 1000)

currDate = dayFormat.format(Date())

nextDate = dayFormat.format(date1)

answers.add("Дней от " + currDate + " до " + nextDate + " = " + days)

callback.accept(java.lang.String.join(", ", answers /\*hashMap.get(entry.getKey()))\*/))

}

if (text.contains("переведи число") || text.contains("переведи")) {

val numPattern = Pattern.compile("переведи число ([0-9]+)", Pattern.CASE\_INSENSITIVE)

val matcher = numPattern.matcher(text)

if (matcher.find()) {

val number = matcher.group(1)

NumberToString.getNumString(Integer.valueOf(number), Consumer<String?> { s ->

if (s != null) {

answers.add(s)

}

Log.i("WEATHER", "AI: $s")

callback.accept(java.lang.String.join(", ", answers))

})

}

}

if (text.contains("погода в городе")) {

val cityPattern = Pattern.compile("погода в городе (\\p{L}+)", Pattern.CASE\_INSENSITIVE)

val matcher = cityPattern.matcher(text)

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 /\*hashMap.get(entry.getKey()))\*/

)

)

})

//return "Не знаю я, какая там погода у вас в городе "+cityName;

}

}

if (text.contains("ковид в стране"))

{

val numPattern = Pattern.compile("ковид в стране (\\p{L}+)", Pattern.CASE\_INSENSITIVE)

val matcher = numPattern.matcher(text)

if (matcher.find())

{

val country = matcher.group(1)

CovidToString.getCovidString(AI.firstUpperCase(country), Consumer<String?> { s ->

if (s != null) {

answers.add(s)

}

callback.accept(java.lang.String.join(", ", answers))

})

}

}

if (text.contains("курс валюты"))

{

CurrencyToString.getCurrencyString(Consumer<String?> { s->

if (s!= null) {

answers.add(s)

}

callback.accept(java.lang.String.join(", ", answers))

})

}

if (text.contains("фильмы")) {

val finalText = text

object : AsyncTask<String?, Int?, Void?>() {

override fun doInBackground(vararg params: String?): Void? {

try {

answers.add(ParsingFilmService.getFilm(finalText))

} catch (e: IOException) {

e.printStackTrace()

}

return null

}

override fun onPostExecute(aVoid: Void?) {

super.onPostExecute(aVoid)

callback.accept(java.lang.String.join(", ", answers))

}

}.execute(\*finalText.split(",").toTypedArray())

}

if (text.contains("праздник")) {

val finalText = text

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())

/\*Observable.fromCallable(new Callable <Object>() {

@Override

public Object call() {

try {

answers.add(ParsingHtmlService.getHoliday(date));

return answers;

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

})

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe((result2) -> {

callback.accept(String.join(", ", answers));

});

Observable.fromCallable( Callable <Object> {

@Override

Object call() {

try {

answers.add(ParsingHtmlService.getHoliday(date));

return answers;

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

})\*/

}

}

}

**ParsingFilmService**

package com.example.kotlinapp.ParsingService

import org.jsoup.Jsoup

import org.jsoup.select.Elements

import java.io.IOException

import java.util.\*

object ParsingFilmService {

private const val URL = "https://www.film.ru/online/year/"

@Throws(IOException::class)

fun getFilm(date: String): String

{

val reg = Regex("[\\s+]")

var div: Elements? = null

val sParts = date.split(reg).toTypedArray()

val iParts = arrayOfNulls<String>(sParts.size)

for (i in sParts.indices)

iParts[i] = sParts[i]

val document = Jsoup.connect(URL + iParts[1]).get()

val str: MutableList<String> = ArrayList()

for (i in 1..5) {

div = document.select("#movies\_list > div:nth-child(1) > a:nth-child($i) > strong")

if (div.isEmpty())

break

else

str.add("""${div.text()}.""".trimIndent()

)

}

return if (str.size != 0) str.toString().replace("^\\[|\\]$".toRegex(), "").replace(",", "")

else

"Нет данных по этому году"

}

}

**MessageService**

package com.example.kotlinapp.MessageService

import com.example.kotlinapp.Database.MessageEntity

import java.util.\*

class Message {

public var text: String? = null

public var date: Date? = null

public var isSend: Boolean

public var str: ArrayList<String>? = null

public var message: Message? = 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()

}

/\* public Message(Message message, boolean isSend) {

this.message = message;

this.isSend = isSend;

this.date = new Date();

}\*/

constructor(entity: MessageEntity) {

text = entity.text

if (entity.date == null) date = null

else date = Date(entity.date)

isSend = entity.isSend !== 0

}

}

**MessageListAdapter**

package com.example.kotlinapp.MessageService

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import androidx.recyclerview.widget.RecyclerView

import com.example.kotlinapp.R

import java.util.\*

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.kotlinapp.MessageService

import android.view.View

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.example.kotlinapp.R

import java.text.DateFormat

import java.text.SimpleDateFormat

open class MessageViewHolder(itemView: View) :

RecyclerView.ViewHolder(itemView)

{

private var messageText: TextView

private var messageDate: TextView

fun bind(message: Message) {

messageText.text = message.text

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)

}

}

**DBHelper**

package com.example.kotlinapp.Database

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"

}

}

**MessageEntity**

package com.example.kotlinapp.Database

import com.example.kotlinapp.MessageService.Message

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

}

}

**Currency**

package com.example.kotlinapp.CurrencyAPI

import com.google.gson.annotations.Expose

import com.google.gson.annotations.SerializedName

import java.io.Serializable

class Currency : Serializable {

@SerializedName("rates")

@Expose

var rates: Currency2? = null

inner class Currency2 {

@SerializedName("RUB")

@Expose

var RUB: Double? = null

@SerializedName("USD")

@Expose

var USD: Double? = null

}

}

**CurrencyApi**

package com.example.kotlinapp.CurrencyAPI

import retrofit2.Call

import retrofit2.http.GET

import retrofit2.http.Query

interface CurrencyApi {

@GET("latest?access\_key=c9247f8509de4c72ec5ff1eb57b3ff68")

fun getCurrentCurrency(@Query("symbols") temp: String?): Call<Currency?>?

}

**CurrencyService**

package com.example.kotlinapp.CurrencyAPI

import retrofit2.Retrofit

import retrofit2.converter.gson.GsonConverterFactory

object CurrencyService {

val apiCurrency: CurrencyApi

get() {

val retrofit = Retrofit.Builder()

.baseUrl("http://data.fixer.io/api/")

.addConverterFactory(GsonConverterFactory.create())

.build()

return retrofit.create(CurrencyApi::class.java)

}

}

**CurrencyToString**

package com.example.kotlinapp.CurrencyAPI

import android.os.Build

import androidx.annotation.RequiresApi

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

import java.util.function.Consumer

object CurrencyToString {

@RequiresApi(api = Build.VERSION\_CODES.N)

var result2: Currency? = null

var answer = ""

@RequiresApi(api = Build.VERSION\_CODES.N)

fun getCurrencyString(callback: Consumer<String?>) {

val api2: CurrencyApi = CurrencyService.apiCurrency

val call = api2.getCurrentCurrency("USD,RUB")

call!!.enqueue(object : Callback<Currency?> {

override fun onResponse(call: Call<Currency?>, response: Response<Currency?>) {

result2 = response.body()

val result2znakaRUB = java.lang.String.format("%.2f", result2!!.rates!!.RUB)

val temp = result2!!.rates!!.RUB!! / result2!!.rates!!.USD!!

val result2znakaUSD = String.format("%.2f", temp)

answer += "Евро: $result2znakaRUB руб.\n"

answer += "Доллар: $result2znakaUSD руб."

//answer += "";

callback.accept(answer)

answer = ""

}

override fun onFailure(call: Call<Currency?>, t: Throwable) {

callback.accept("Не могу получить данные")

}

})

}

}

**Covid**

package com.example.kotlinapp.CovidAPI

import com.google.gson.annotations.Expose

import com.google.gson.annotations.SerializedName

import java.io.Serializable

class Covid : Serializable {

@SerializedName("All")

@Expose

var All: Covid2? = null

inner class Covid2 {

@SerializedName("confirmed")

@Expose

var confirmed: Int? = null

@SerializedName("deaths")

@Expose

var deaths: Int? = null

@SerializedName("recovered")

@Expose

var recovered: Int? = null

}

}

**CovidApi**

package com.example.kotlinapp.CovidAPI

import com.google.gson.annotations.Expose

import com.google.gson.annotations.SerializedName

import java.io.Serializable

class Covid : Serializable {

@SerializedName("All")

@Expose

var All: Covid2? = null

inner class Covid2 {

@SerializedName("confirmed")

@Expose

var confirmed: Int? = null

@SerializedName("deaths")

@Expose

var deaths: Int? = null

@SerializedName("recovered")

@Expose

var recovered: Int? = null

}

}

**CovidService**

package com.example.kotlinapp.CovidAPI

import retrofit2.Retrofit

import retrofit2.converter.gson.GsonConverterFactory

object CovidService {

val apiCovid: CovidApi

get() {

val retrofit = Retrofit.Builder()

.baseUrl("https://covid-api.mmediagroup.fr")

.addConverterFactory(GsonConverterFactory.create())

.build()

return retrofit.create(CovidApi::class.java)

}

}

**CovidToString**

package com.example.kotlinapp.CovidAPI

import android.os.Build

import androidx.annotation.RequiresApi

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

import java.util.function.Consumer

object CovidToString {

@RequiresApi(api = Build.VERSION\_CODES.N)

var result2: Covid? = null

var answer = ""

@RequiresApi(api = Build.VERSION\_CODES.N)

fun getCovidString(country: String?, callback: Consumer<String?>) {

val api2: CovidApi = CovidService.apiCovid

val call = api2.getCurrentCountry(country)

call!!.enqueue(object : Callback<Covid?> {

override fun onResponse(call: Call<Covid?>, response: Response<Covid?>) {

result2 = response.body()

answer += """

Подтвержденные случаи: ${result2!!.All!!.confirmed.toString()}

Количество смертей: ${result2!!.All!!.deaths.toString()}

Количество выздоровленных: ${result2!!.All!!.recovered}

""".trimIndent()

callback.accept(answer)

answer = ""

}

override fun onFailure(call: Call<Covid?>, t: Throwable) {

callback.accept("Не могу получить данные")

}

})

}

}

**Файлы ресурсов**

**Assistant\_bubble**

<?xml version="1" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:shape="rectangle">

<solid android:color="@color/assistantBubble"></solid>

<corners android:radius="20dp"></corners>

</shape>

**User\_bubble**

<?xml version="1" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:shape="rectangle">

<solid android:color="@color/userBubble" ></solid>

<corners android:radius="20dp"></corners>

</shape>

**Activity\_main**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/chatWindow"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:textAlignment="viewStart"

android:textDirection="inherit"

tools:context=".MainActivity">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/chatMessageList"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

app:layout\_constraintBottom\_toTopOf="@+id/linearLayout"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.0"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<LinearLayout

android:id="@+id/linearLayout"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

android:orientation="horizontal"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent">

<EditText

android:id="@+id/questionField"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginEnd="8dp"

android:layout\_weight="1"

android:ems="10"

android:hint="@string/InputText"

android:inputType="textPersonName" />

<Button

android:id="@+id/sendButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/sendText" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Assistant\_message**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:padding="4dp"

android:paddingStart="4dp"

android:paddingLeft="4dp"

android:paddingTop="4dp"

android:paddingEnd="4dp"

android:paddingRight="4dp"

android:paddingBottom="4dp">

<ImageView

android:id="@+id/imageView2"

android:layout\_width="60dp"

android:layout\_height="60dp"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginBottom="8dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:srcCompat="@drawable/laik2" />

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:text="Ассистент"

app:layout\_constraintStart\_toEndOf="@+id/imageView2"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/messageTextView"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

android:background="@drawable/assistant\_bubble"

android:text="TextView"

app:layout\_constraintBottom\_toTopOf="@+id/messageDateView"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.0"

app:layout\_constraintStart\_toEndOf="@+id/imageView2"

app:layout\_constraintTop\_toBottomOf="@+id/textView"

app:layout\_constraintVertical\_bias="1.0" />

<TextView

android:id="@+id/messageDateView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginBottom="8dp"

android:text="10:10"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toEndOf="@+id/imageView2" />

</androidx.constraintlayout.widget.ConstraintLayout>

**User\_message**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingStart="4dp"

android:paddingLeft="4dp"

android:paddingTop="4dp"

android:paddingEnd="4dp"

android:paddingRight="4dp"

android:paddingBottom="4dp">

<TextView

android:id="@+id/messageTextView"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

android:background="@drawable/user\_bubble"

android:text="TextView"

app:layout\_constraintBottom\_toTopOf="@+id/messageDateView"

app:layout\_constraintEnd\_toStartOf="@+id/imageView"

app:layout\_constraintHorizontal\_bias="1.0"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/userName" />

<TextView

android:id="@+id/userName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:text="Пользователь"

app:layout\_constraintEnd\_toStartOf="@+id/imageView"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/messageDateView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

android:text="10:10"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toStartOf="@+id/imageView" />

<ImageView

android:id="@+id/imageView"

android:layout\_width="60dp"

android:layout\_height="60dp"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginBottom="8dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:srcCompat="@drawable/leon\_60x60" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Menu\_main**

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:orderInCategory="100"

app:showAsAction="never"

android:title="Дневная тема"

android:id = "@+id/day\_settings"/>

<item

android:orderInCategory="100"

app:showAsAction="never"

android:title="Ночная тема"

android:id = "@+id/night\_settings"/>

<item

android:orderInCategory="100"

android:id="@+id/clear\_dialog"

app:showAsAction="never"

android:title="@string/clear\_dialog\_item" />

</menu>

**Colors**

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="purple\_200">#FFBB86FC</color>

<color name="purple\_500">#FF6200EE</color>

<color name="purple\_700">#FF3700B3</color>

<color name="teal\_200">#FF03DAC5</color>

<color name="teal\_700">#FF018786</color>

<color name="black">#FF000000</color>

<color name="white">#FFFFFFFF</color>

<color name="assistantBubble">#F1FFAA</color>

<color name="userBubble">#C6FFF7</color>

</resources>

**Strings**

<resources>

<string name="app\_name">VoiceAssistant</string>

<string name="InputText">Задайте вопрос</string>

<string name="sendText">Отправить</string>

<string name = "answer" >Вопрос понял. Думаю..."</string>

<string name="day\_settings">Дневная тема</string>

<string name="night\_settings">Ночная тема</string>

<string name="clear\_dialog\_item">Очистить диалог</string>

</resources>

**Themes**

<resources xmlns:tools="http://schemas.android.com/tools">

<!-- Base application theme. -->

<style name="Theme.Kotlinapp" parent="Theme.AppCompat.Light">

<!-- Primary brand color. -->

<item name="colorPrimary">@color/purple\_500</item>

<item name="colorPrimaryVariant">@color/purple\_700</item>

<item name="colorOnPrimary">@color/white</item>

<!-- Secondary brand color. -->

<item name="colorSecondary">@color/teal\_200</item>

<item name="colorSecondaryVariant">@color/teal\_700</item>

<item name="colorOnSecondary">@color/black</item>

<!-- Status bar color. -->

<item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>

<!-- Customize your theme here. -->

</style>

</resources>

**Colors\_night**

<?xml version="1.0" encoding="utf-8"?>

<resources><color name="assistantBubble">#808080</color>

<color name="userBubble">#0000ff</color></resources>

**Themes\_night**

<resources xmlns:tools="http://schemas.android.com/tools">

<!-- Base application theme. -->

<style name="Theme.Kotlinapp" parent="Theme.AppCompat.DayNight">

<!-- Primary brand color. -->

<item name="colorPrimary">@color/purple\_200</item>

<item name="colorPrimaryVariant">@color/purple\_700</item>

<item name="colorOnPrimary">@color/black</item>

<!-- Secondary brand color. -->

<item name="colorSecondary">@color/teal\_200</item>

<item name="colorSecondaryVariant">@color/teal\_200</item>

<item name="colorOnSecondary">@color/black</item>

<!-- Status bar color. -->

<item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>

<!-- Customize your theme here. -->

</style>

</resources>

**Strings\_en**

<resources>

<string name="app\_name">VoiceAssistant</string>

<string name="InputText">Ask me a question</string>

<string name="sendText">Send</string>

<string name = "answer" >Think about that question..."</string>

<string name="day\_settings">Daytime theme</string>

<string name="night\_settings">Night theme</string>

<string name="clear\_dialog\_item">Clear Dialog</string>

</resources>