**Лабораторная работа 3**

**Изменение интерфейса чата**

Для отображения диалога лучше использовать список. RecyclerView – это список, который можно скролить вверх и вниз. Он состоит из ячеек различного вида.

Чтобы использовать RecyclerView нужно:

1. Создать разметку, на которую поместить RecyclerView.
2. Создать разметку для ячейки RecyclerView.
3. Описать модель (класс) для ячейки.
4. Описать класс ViewHolder, загружающий данные из модели в ячейку.
5. Описать класс Adapter, который свяжет все данные в одно целое.

Начните с создания ячеек. Оформите макет согласно рисунку 1. Описание создания ниже по тексту.

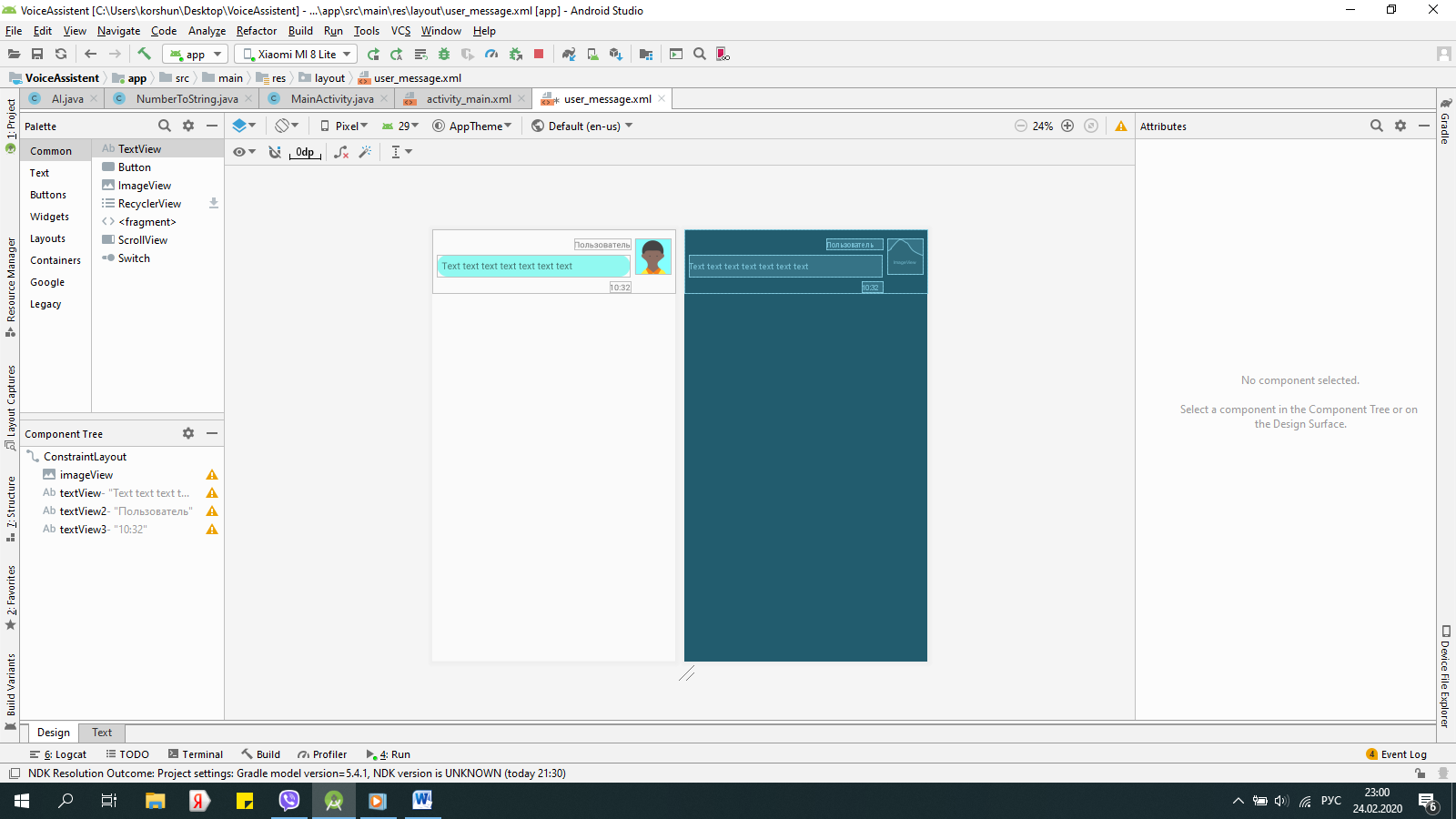


Рисунок 1. Разметка user\_message.xml

Сначала создайте цветные заготовки для фраз.

В контекстном меню папки res/drawable выберите New -> File. Создайте файл с именем user\_bubble.xml. В него перенесите следующий код:

*<?***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="#91F9F1"**></**solid**>  
 <**corners android:radius="20dp"**></**corners**>  
  
</**shape**>

Данный код задает прямоугольник голубого цвета со скругленными краями радиуса 20dp. Прямоугольнику обязательно нужно задать размеры. В данном случае ширина и длина имеют значения match\_parent.

По аналогии создайте второй файл с именем assistant\_bubble.xml. Он должен отличаться цветом фона.

Создайте новый layout-файл. В контекстном меню папки res/layout выберите New -> XML -> Layout XML File. Задайте ему имя user\_message, значение поля Root Tag – ConstraintLayout (введите полное имя, иначе будут ошибки). Задайте максимальное значение ширины и высоты.

Разместите в правом верхнем уголу ImageView. Размеры картинки: 60dp на 60dp, отступы от краев (правого и верхнего) – по 8dp.

Слева от картинки расположите TextView. Размеры: ширина – match\_constraint, высота – wrap\_content. Отступы по 8 dp с трех сторон. Свойству background присвойте значение созданного прямоугольника user\_bubble. Установите внутренние отступы – padding в значение 4dp.

Добавьте еще два TextView для отображения имени и времени. Выровняйте их в соответствии с правым краем TextView сообщения.

Измените значение высоты ConstraintLayout на wrap\_content.

Создайте еще один layout-файл с именем assistant\_message.xml и оформите согласно рисунку 2.

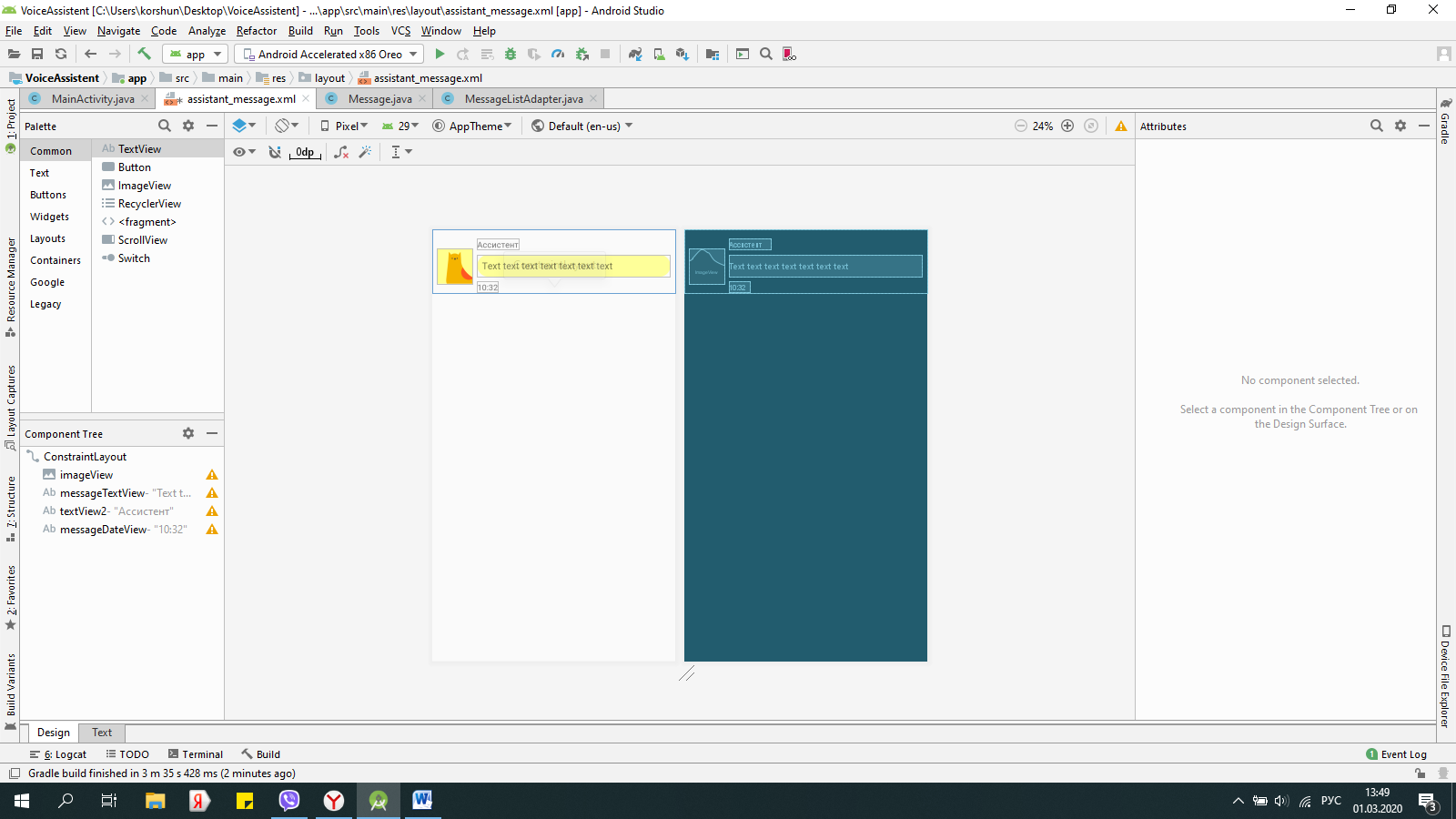


Рисунок 2. Разметка assistant\_message.xml

Дайте виджетам следующие названия:

* цветное текстовое поле для отображения фраз диалога назовите **messageTextView**;
* текстовое поле для вывода времени назовите **messageDateView**.

Убедитесь, что переименование выполнено в обоих файлах.

Ячейки для RecyclerView созданы.

С каждой ячейкой должна быть связана модель. В данном случае модель будет содержать текст и дату сообщения. В программе роль модели выполняет класс. Создайте класс Message со следующими полями:

**public** String **text**;  
**public** Date **date**;  
**public** Boolean **isSend**;

Логическое поле необходимо, чтобы различать сообщения: посланное пользователем от принятого им.

Сгенерируйте автоматически конструктор с двумя параметрами. Для этого внутри класса вызовите контекстное меню, а в нем пункт Generate… (Alt+Insert). В списке выберите конструктор, затем выделите поля text и isSend и нажмите OK. Поле date будет заполняться автоматически текущей датой:

**this**.**date** = **new** Date();

Модель описана.

Далее создайте класс MessageViewHolder, который будет отвечать за отображение ячеек. В данном случае он будет отображать текст и дату из модели в ячейке таблицы. Сделайте его наследником класса RecyclerView.ViewHolder.

Класс RecyclerView не включен в проект по умолчанию, поэтому его придется добавить. Самое простое и в тоже время самое разумное, что нужно сделать, это, в контекстном меню выбрать Add dependency on … Сборщик проекта gradle сам «подтянет» нужную зависимость.

Добавьте в класс поля для отображения сообщения и даты:

**protected** TextView **messageText**;  
**protected** TextView **messageDate**;

Класс подчеркнут красным цветом. Чтобы узнать, чего ему не хватает, нажмите Alt+Enter. Он просит создать конструктор. Создайте его автоматически

**public** MessageViewHolder(@NonNull View itemView) {  
 **super**(itemView);  
}

Добавьте в конструктор связь между полем класса и компонентом в разметке:

**messageText** = itemView.findViewById(R.id.***messageTextView***);  
**messageDate** = itemView.findViewById(R.id.***messageDateView***);

Опишите в классе метод bind, который «положит» данные из модели в соответствующие текстовые поля.

**public void** bind(Message message) {  
 messageText.setText(message.text);  
 DateFormat fmt = **new** SimpleDateFormat(**""**);  
 messageDate.setText(fmt.format(message.date));  
}

Обратите внимание, для вывода даты в нужном виде используется класс DateFormat. Вместо пустой строки в конструкторе задайте формат даты/времени сообщения.

Класс ViewHolder описан.

Последнее, что нужно написать – это класс Adapter, который соединит в единое целое Layout, Model и ViewHolder.

Создайте класс MessageListAdapter. Сделайте его наследником класса RecyclerView.Adapter и опишите внутри поле для хранения сообщений:

**public** List<Message> **messageList** = **new** ArrayList<>();

Класс снова подчеркнут. В нем нужно перегрузить 3 метода: onCreateViewHolder, onBindViewHolder и getItemCount. Сделайте это.

Метод onBindViewHolder вызывается каждый раз, когда появляется новое сообщение. Он имеет два параметра:

**public void** onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, **int** position)

Реализуйте в этом методе следующую логику. Возьмите из списка **messageList** сообщение под номером position и передайте его в метод bind параметра holder. Сам параметр holder сначала приведите к классу MessageViewHolder.

Метод onCreateViewHolder принимает 2 аргумента:

**public** RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 **return null**;  
}

Параметр parent это элемент, внутри которого будет отображаться RecyclerView, параметр viewType это тип представления (элемента списка). В данном проекте 2 тип представления.

Опишите в классе две константы для хранения типов:

**private static final int *ASSISTANT\_TYPE***=0;  
**private static final int *USER\_TYPE***=1;

Переменная viewType будет принимать одно из этих значений. В методе onCreateViewHolder напишите обработку пришедших сообщений:

**if** (viewType == ***USER\_TYPE***) {  
 *//создание сообщения от пользователя*  
}  
**else** {  
 *//создание сообщения от ассистента*  
}

Создать сообщение означает создать view и положить в него все данные. Объявите перед условием переменную:

View view;

В случае пользовательского сообщения view нужно загружать из файла user\_message.xml:

view = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.***user\_message***,parent,**false**);

Аналогичные строки напишите для сообщения ассистента.

Метод должен вернуть

**new** MessageViewHolder(view);

Последний метод в классе адаптера getItemCount реализуйте самостоятельно, его назначение очевидно.

Так же понадобится метод для выяснения типа сообщения в зависимости от номера index:

**public int** getItemViewType(**int** index){  
 Message message = **messageList**.get(index);  
 **if** (message.**isSend**){  
 **return *USER\_TYPE***;  
 }  
 **else return *ASSISTANT\_TYPE***;  
}

Адаптер описан.

Перейдите к файлу activity\_main.xml и замените в нем **TextView** на **RecyclerView**. «Притащите» его к краям как на рисунке 3. Дайте название **chatMessageList**.

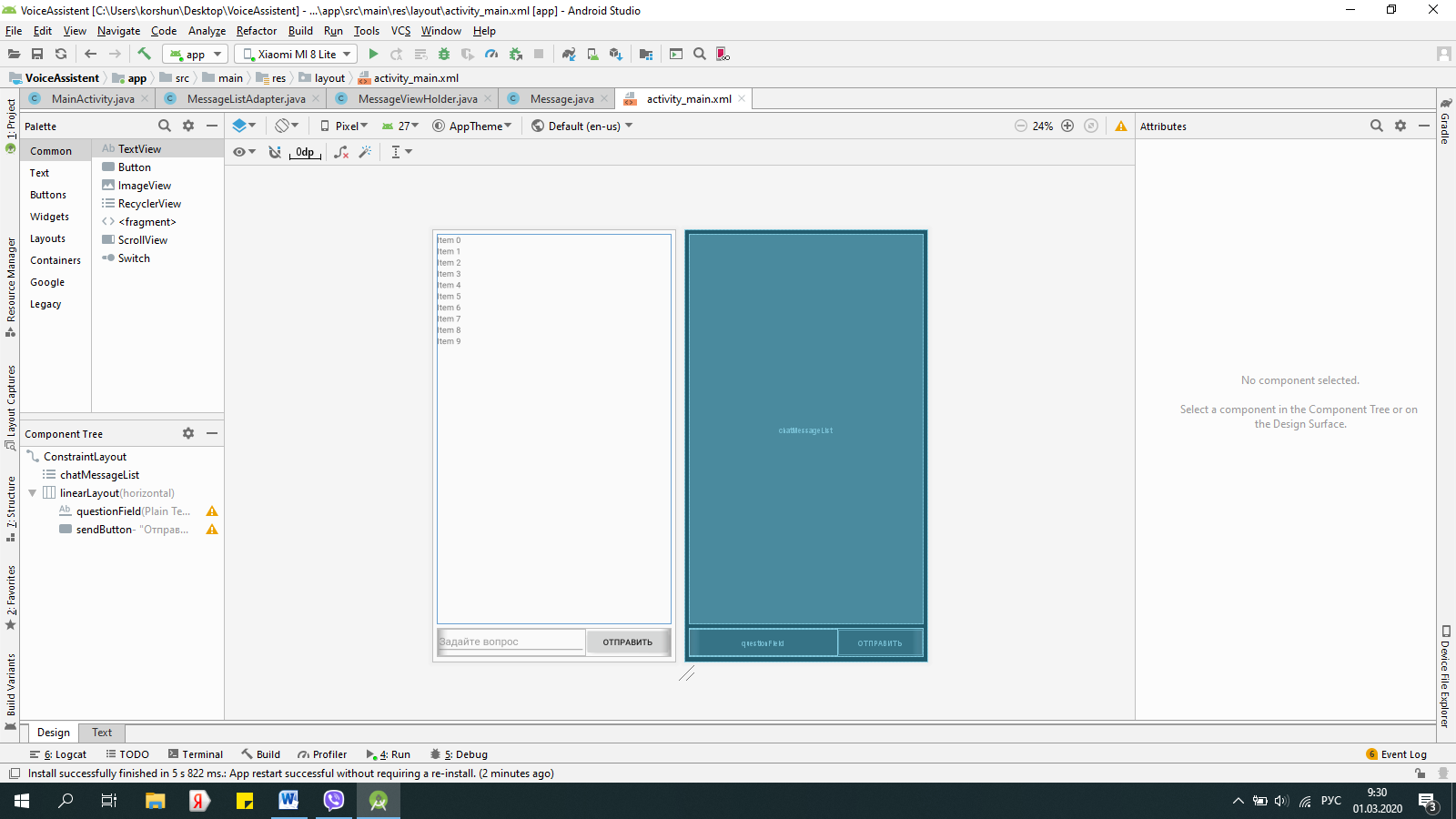


Рисунок 3. RecyclerView в activity\_main.xml

В классе MainActivity нужно исправить появившиеся ошибки. Вместо переменной **chatWindow** опишите новую переменную:

**protected** RecyclerView **chatMessageList**;

Инициализируйте ее в методе onCreate.

Добавьте в класс поле:

**protected** MessageListAdapter **messageListAdapter**;

Создайте экземпляр класса в методе onCreate

**messageListAdapter** = **new** MessageListAdapter();

Инициализируйте chatMessageList и задайте ему адаптер:

**chatMessageList**.setLayoutManager(**new** LinearLayoutManager(**this**));  
**chatMessageList**.setAdapter(**messageListAdapter**);

Обратите внимание, в качестве разметки списка выбран LinearLayout.

Исправьте ошибки в методе onSend. Вместо вывода сообщения пользователя напишите:

**messageListAdapter**.**messageList**.add(**new** Message(text, **true**));

Значение true показывает, что это сообщение пользователя. Аналогичным образом замените вывод сообщения от помощника.

Необходимо уведомить адаптер, что произошло изменение содержимого списка. Вставьте в код следующую строку:

**messageListAdapter**.notifyDataSetChanged();

После оповещения списка логично показать последнее добавленное сообщение:

**chatMessageList**.scrollToPosition(**messageListAdapter**.**messageList**.size()-1);

Таким образом, список будет прокручиваться к новому сообщению.

Запуск! Запустите приложение и убедитесь в его роботоспособности.