Министерство образования Новосибирской области
ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ИНТЕРАКТИВНЫХ ЗАМЕТОК

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03. Разработка мобильных приложений

НАТКиГ.211700.043.000ПЗ

Разработал: Рябой В.П.

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наши дни человек воспринимает колоссальное количество информации, которая занимает значительное место в памяти. При этом приходится выбирать из плотного потока данных именно то, что необходимо запомнить. В связи с большим потоком, часто важная информация теряется.

Заметки – это то, что поможет не потерять важную информацию и не забыть о чем-либо.Также приложение должно присылать уведомление, если пользователь не просто записал информацию, а установил дату напоминания к заметке.

Целью курсового проекта является разработка приложения для записи, хранения и отображения важной информации.

Для достижения цели поставлены следующие задачи:

1. исследовать предметную область;
2. спроектировать макет приложения;
3. спроектировать базу данных;
4. разработать приложение по макету;
5. протестировать полученный продукт.

Объектом исследования является хранение информации.

Предметом исследования является электронный способ хранения информации.

Практическая значимость курсового проекта состоит в возможности решения вопросов хранения и фильтрации информации. Разработанная программа способствует сохранению важной для человека информации.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Описание предметной области

На сегодняшний день существует множество способов хранить информацию. Человек использует бумагу чтобы записать что-то важное. Цифровую информацию чаще всего хранят на жестких дисках, Flash-носителях или оптических дисках. Дополнительно, существует масса облачных хранилищ, таких как GoogleDrive.

Но каждый из вышеперечисленных способов имеет свои недостатки. Не всегда под рукой имеется листок бумаги и ручка или карандаш, чтобы что-то записать. Жесткий диск или Flash-носитель нужно специально носить с собой, а также найти персональный компьютер, к которому и подключать их, чтобы что-то сохранить.

Облачное хранилище данных более универсально, по сравнению с материальным носителем, так как оно может быть установлено на смартфоне. Но оно хранит файлы, в которых находится информация, а не саму информацию, что нужна пользователю.

Приложение для создания интерактивных заметок — это программный продукт, позволяющий создавать заметки на мобильном устройстве, с возможностью установки даты оповещения. Под возможностью установки даты оповещения имеется в виду функция, позволяющая установить время, появления Push-уведомленияс текстом, который хранится в заметке.

Приложения такого формата помогают избавиться от поиска ручки, ровной поверхности, чтобы сделать какую-либо запись, а также пролистывания страниц в поисках нужной даты. Смартфон имеется практически у каждого, и он всегда под рукой. У бумажных органайзеров имеется одно преимущество, перед их программной версией — это скорость набора информации, но по сравнению с возможностями электронных устройств это преимущество незначительно.

1.2 Образ клиента

Данное приложение нацелено на различных пользователей, которые сталкиваются с большим количеством информации. Ими могут быть людистарше 14 лет, различные «Big Data Analyst'ы» (Аналитик больших данных), инженеры, разработчики программных продуктов и другие. Каждый, кто работает с большим потоком данных, может забыть что-то, и это приложение поможет справиться с этой проблемой.

1.3 Сценарии

Денис работает специалистом по глубинному машинному обучению, и он женат на Елене. Недавно, руководство поставило ему задачу по улучшению составленного ранее алгоритма, по которому машина «думает», на это ему давался срок в 1 месяц. Елена периодически просит мужа купить продукты, но Денис часто про них забывает, так как занят работой. Чтобы не забывать про просьбы жены, специалисту стоит установить приложение, решающее его проблему.

Майкл часто ходит в компьютерные салоны, чтобы собрать свой ПК. Он записывает на листок, какие комплектующие ему необходимы. Один раз, когда Майкл пришел в магазин, он не нашел записки, и чтобы подобного не повторялось, ему стоит установить приложение, которое решит его проблему.

Евгений работает программистом. Перед сном у него часто появляются идеи, которые он хотел бы реализовать, и чтобы не забыть про них, Евгению стоило бы установить соответствующее приложение.

1.4 Сбор и анализ прототипов

Рассмотрим несколько приложений, решающих подобные задачи. В качестве примеров выбраны системное приложение «Заметки» (RedmiNote 7), системное приложение «Календарь» RedmiNote 7. Чтобы лучше понимать, что это за приложения, рассмотрим каждое из них.

Системное приложение «Заметки» (RedmiNote 7) является встроенным в систему смартфона XiaomiRedmiNote 7. Дизайн выполнен в минималистичном стиле. Предполагается несколько стандартных размеров шрифтов, а также сортировка по дате изменения или по дате создания. В самих заметках имеется возможность изменения цвета фона, создания напоминания, скрытия, удаления или перемещения заметки в другую папку. Помимо текста, в заметку можно добавить звуковой файл или графический. Также, кроме самих заметок, в приложении имеются задачи, которые являются более простым видом заметки и могут быть «выполнены» или оставаться активными. Данное приложение не может использоваться на других устройствах, имеются задачи, в которых нет необходимости. Приложение поддерживает несколько типов данных, но это как достоинство, так и недостаток. Не всегда есть возможность прослушать аудиофайл. На рисунках 1–4 представлен интерфейс приложения «Заметки».

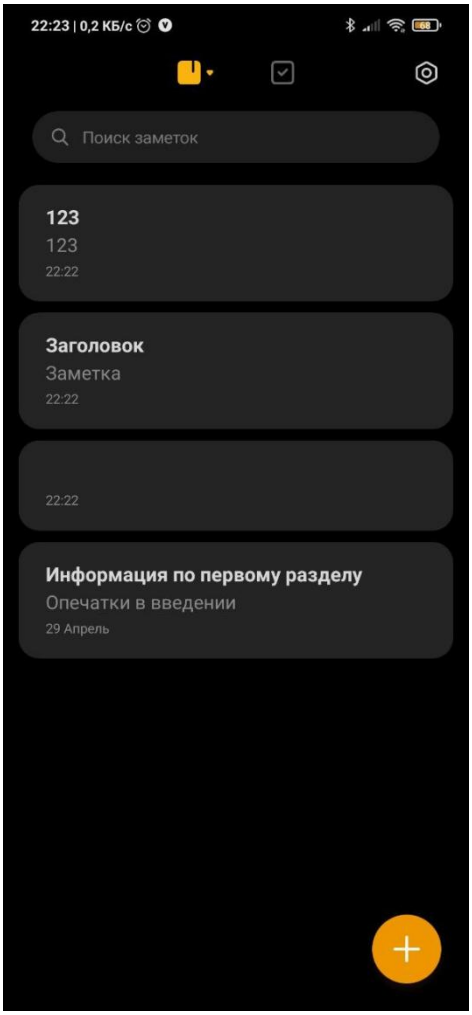

Рисунок 1 — Главный экран

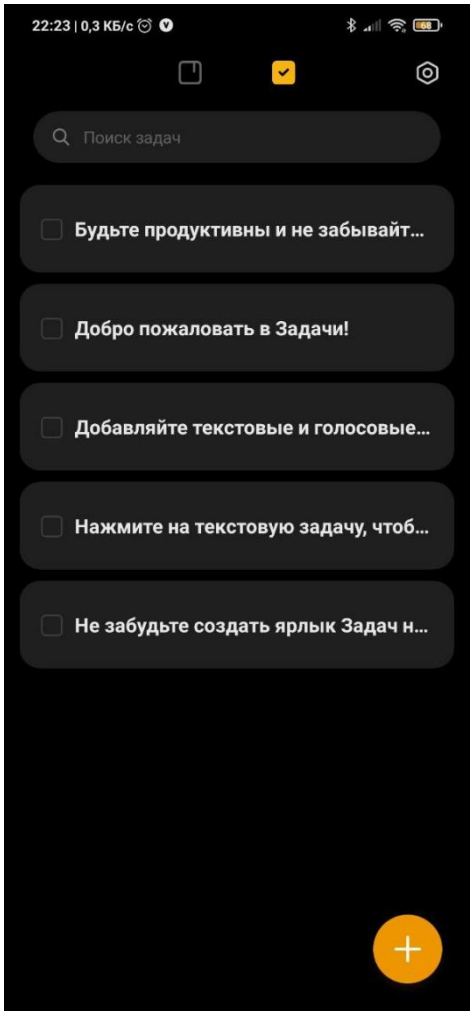

Рисунок 2— Экран Задачи

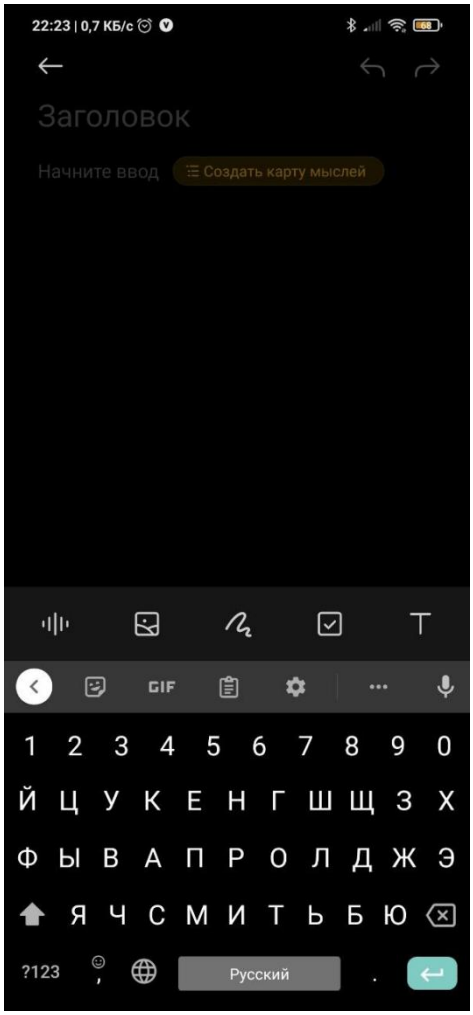

Рисунок 3 — Добавление заметки

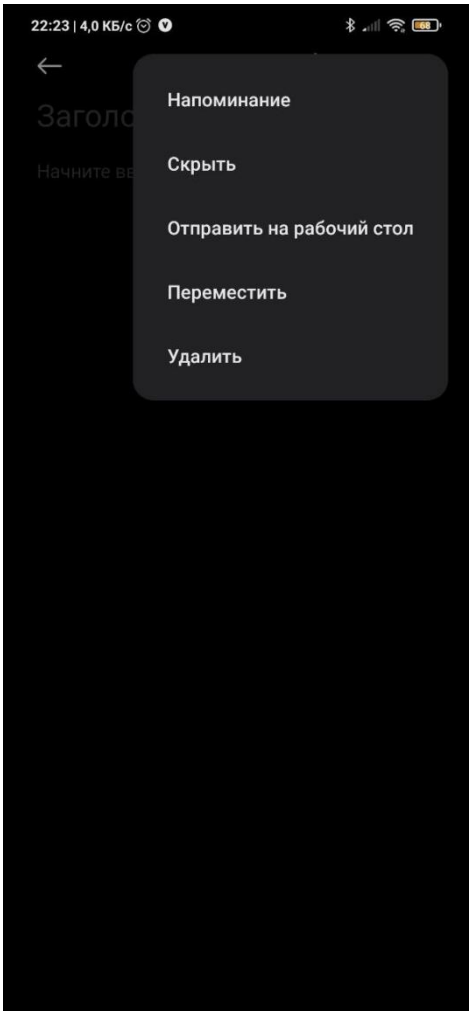

Рисунок 4 — Меню

Системное приложение «Календарь» Xiaomi Redmi Note 7 также является встроенным систему смартфона Xiaomi Redmi Note 7. Дизайн приложения схож с дизайном системного приложения «Заметки» Xiaomi Redmi Note 7. Кроме основных функций календаря, имеется возможность планирования мероприятий, т.е. создание «заметки» на выбранный день с кратким описанием. Также есть возможность создавать заметки, но они не являются ключевым моментом, а потому плохо реализованы. Если на устройстве имеется несколько Google-аккаунтов, то все мероприятия, созданные на других устройствах с одного аккаунта, будут видны и на том смартфоне, где авторизован этот же аккаунт. Данное приложение не нацелено на обычные заметки, а потому просто хранить информацию,

которую нужно было где-то записать, не удобно. На рисунках 5–7, что представлены ниже, показан дизайн приложения «Календарь»

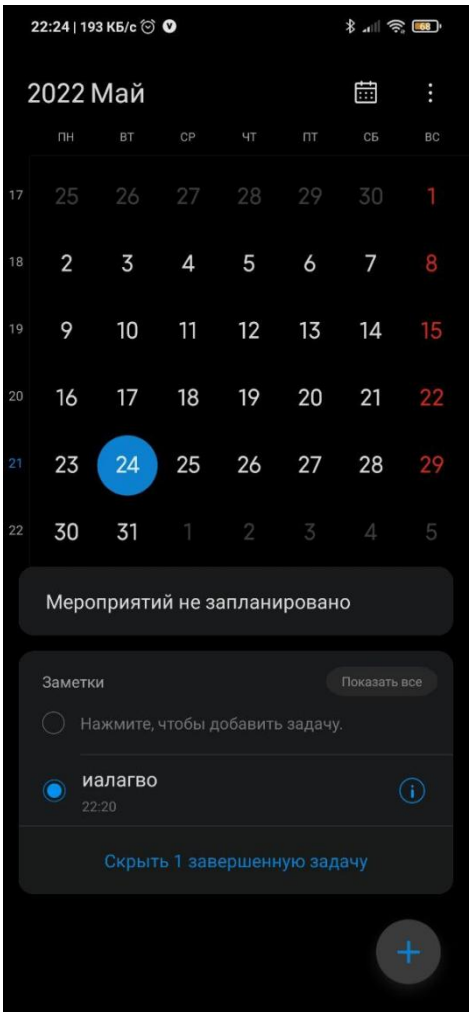

Рисунок 5 — Главный экран

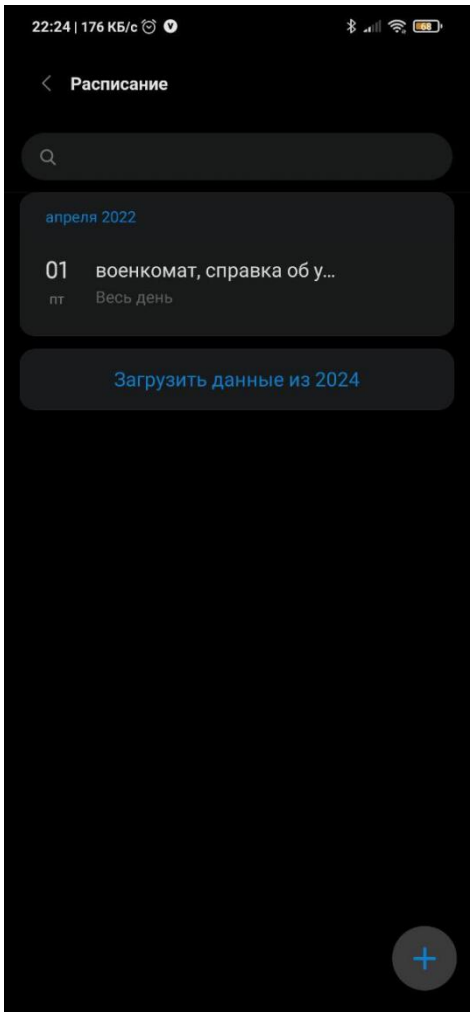

Рисунок 6 — Расписание

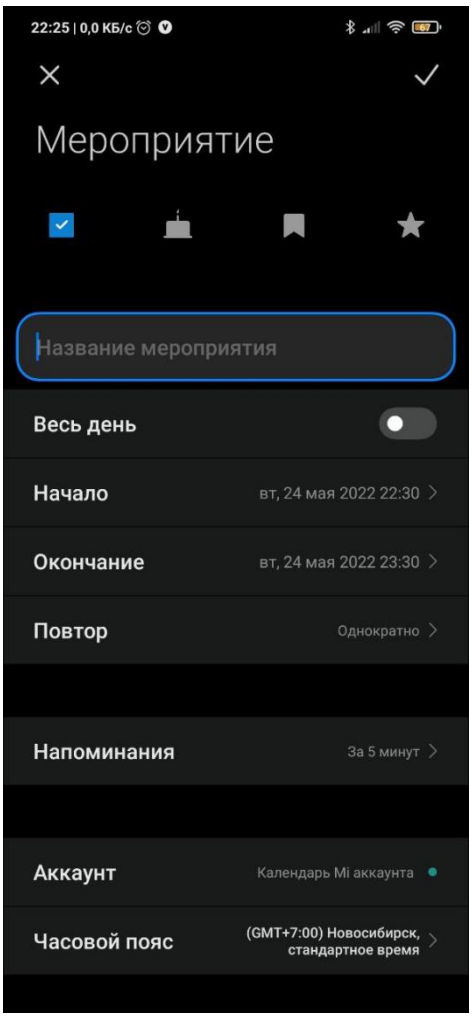

Рисунок 7 — Создание мероприятия

В таблице 1 представлено сравнение данных приложений по следующим критериям: возможность установки на другие устройства, необходимость иметь подключение к интернету, перегруженность интерфейса, сложность в освоении, поддержка русского языка, сложность добавления и работы с данными.

Таблица 1 – Сравнение прототипов

Название приложения	Системное приложение «Заметки»	Системное приложение «Календарь»
Возможность установки на другие устройства	Нет	Нет
Подключение к интернету	Не нужно	Не нужно
Перегруженность интерфейса	Перегружен	Перегружен
Сложность в освоении	Легко	Легко
Поддержка русского языка	Есть	Есть
Сложность добавления и работы с данными	Легко	Сложно

Рассмотрев несколько приложений, выполняющих похожие задачи, было решено написать приложение, которое решало бы задачу хранения данных, но имела более удобный и простой для работы дизайн.

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 UI/UX дизайн проекта

2.1.1 Логотип и цветовая схема

Для проектирования приложения был выбран сервис для разработки различных дизайнов и интерфейсов «Figma».

Для мобильного приложения был разработан логотип, представленный на рисунке 8.

Рисунок 8 — Логотип

Определена цветовая схема приложения для светлой темы, которая представлена на рисунке 9. Она привязана к системной теме приложения и при её изменении меняется тема самого приложения

Рисунок 9 — Цветовая схема приложения

2.1.2 Загрузочный экран

Загрузочный экран содержит в себе логотип приложения — логотип и название (рисунок 10). Экран необходим для скрытия различных процессов работы.

Рисунок 10 — Загрузочный экран

2.1.3 Основной экран

Основной экран содержит список заметок, созданных пользователем (рисунок 11). Для поиска конкретной заметки необходимо ввести заголовок в поле «Поиск заметок» — приложение отсортирует список и выведет только схожие с поиском результаты. Нажав на заметку, откроется её содержимое, где можно добавить или изменить его (рисунок 12).

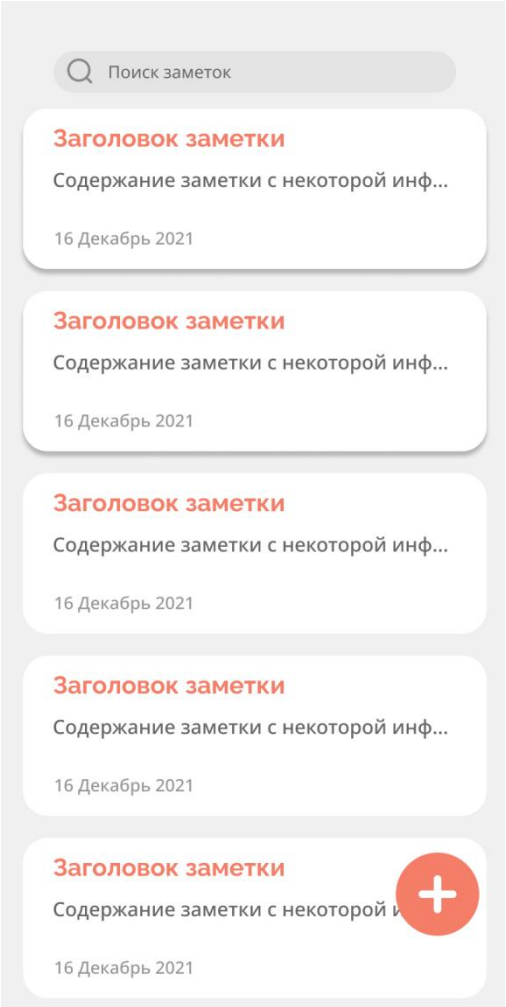

Рисунок 11 — Список заметок

2.1.4 Экран заметки

Экран заметки, представленный на рисунке 12, содержит кнопку возврата к списку заметок, а также кнопку «Меню». В меню имеется 2 функции: удаление заметки и создание напоминания (рисунок 13). При нажатии на удаление, появляется диалоговое в котором предлагается удалить заметку из базы данных (рисунок 14). При создании напоминания появляется диалоговое окно, в котором нужно выбрать дату и время появления Push-уведомления.

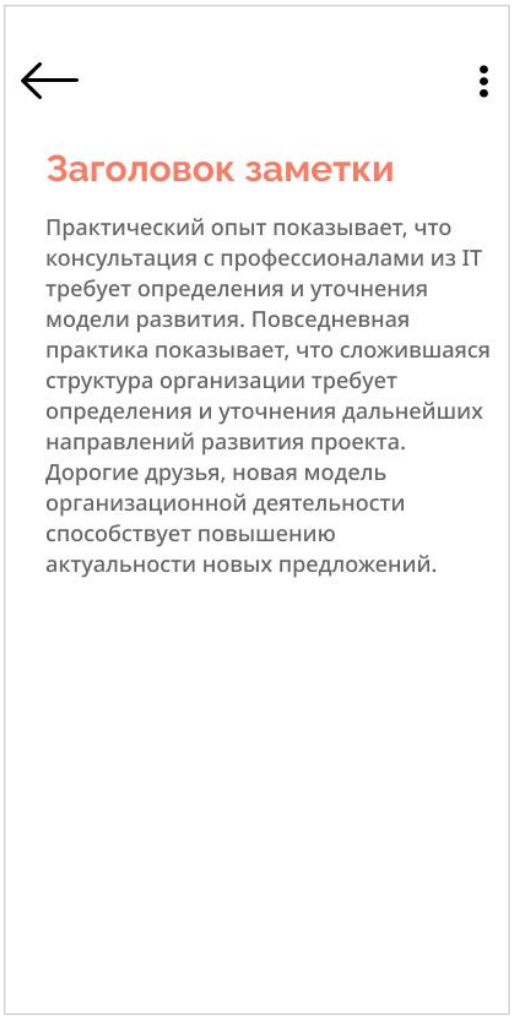

Рисунок 12 — Экран заметки

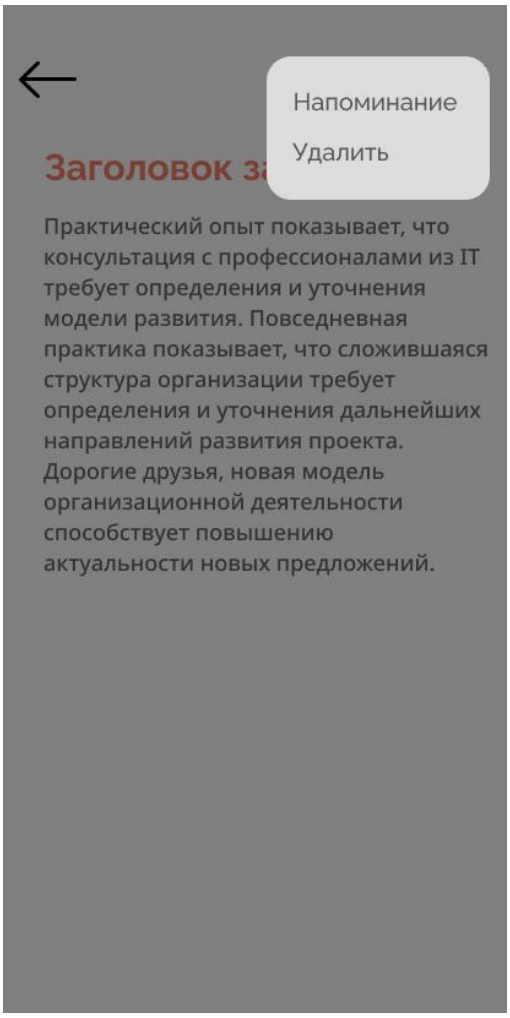

Рисунок 13 — Меню

2.1.5 Диалоговое окно «Удаление заметки»

Диалоговое окно «Удаление заметки» (рисунок 5) предлагает пользователю удалить выбранную заметку или же отменить операцию.

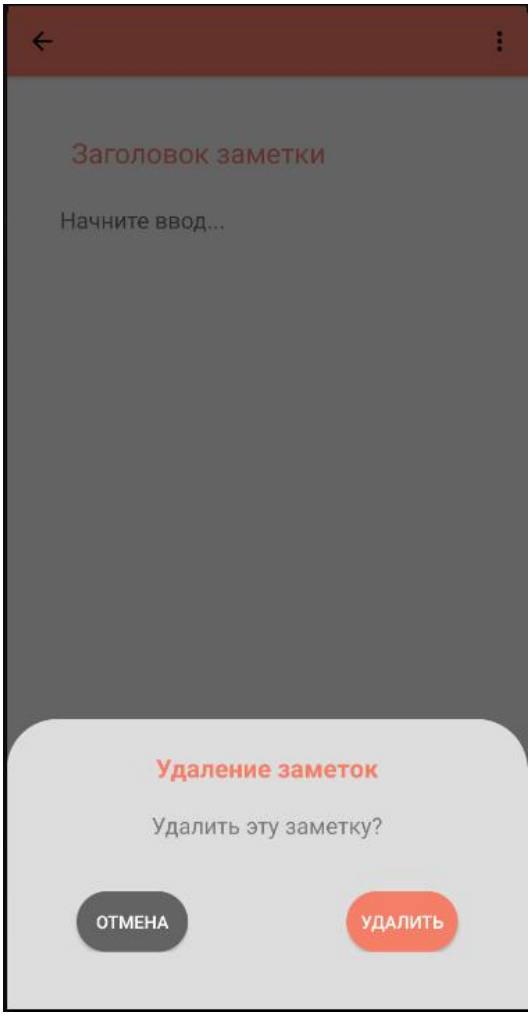

Рисунок 14 — Диалоговое окно «Удаление заметки»

2.1.6 Напоминание

Диалоговое окно «Создание напоминания» позволяется выбрать дату (рисунок 16) и время для показа Push-уведомления (рисунок 15). После выбора даты и времени пользователь устанавливает напоминание или отменяет операцию.

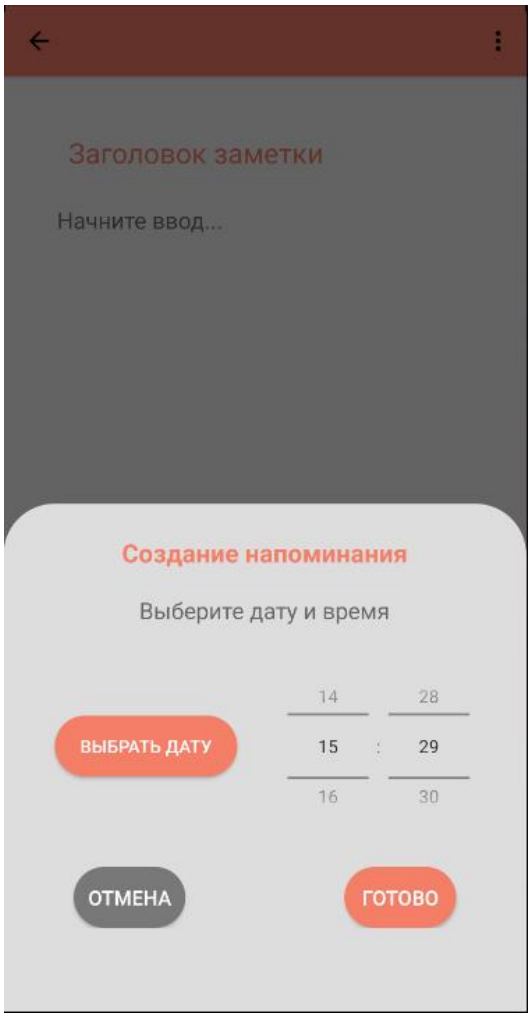

Рисунок 15 —
Экран расписания преподавателя

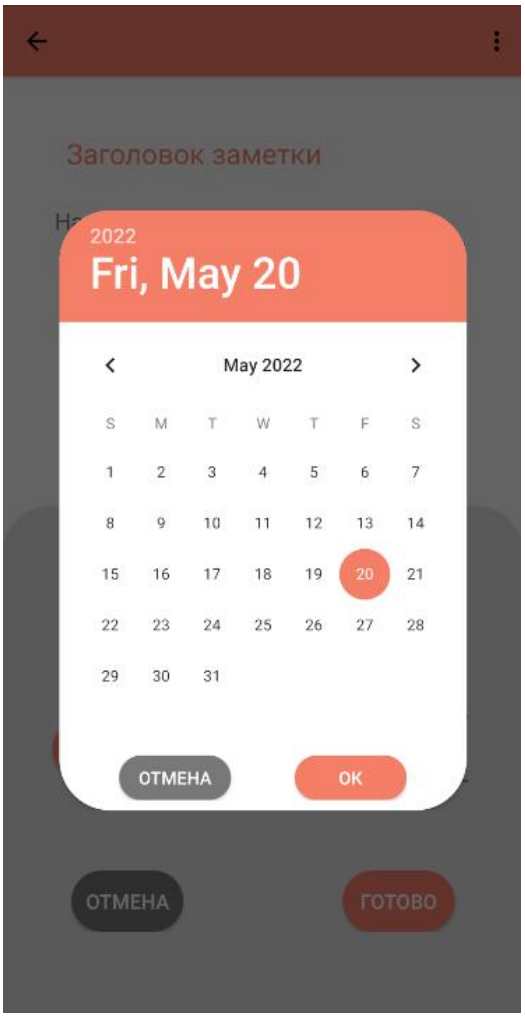

Рисунок 16 — Календарь

2.2 Выбор технологии, языка и среды программирования

Мобильное приложение разрабатывается под Android, так как большинство имеют смартфон на базе этой ОС. Поэтому в качестве среды разработки выбрана Android Studio. Данная IDE обладает конструктором пользовательских интерфейсов, который позволяет облегчить визуальное проектирование приложения, а также эмуляцией устройств и поддержкой системы контроля версий.

В качестве языка программирования выбран Java, т.к. данный язык использует виртуальные машины, что способствует защите приложения. Кроме того, язык поддерживается сообществом, что упрощает решение возникающих проблем, а также поддерживается концепция ООП. Другие языки программирования в той или иной степени привязаны к функциям программно-аппаратных платформ, но слоган Java гласит: «Напиши один раз, запускай где угодно». Кроссплатформенность способствует распространению языка.

3 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

3.1 Разработка базы данных

Схема сущностей — это описание сущностей базы данных, которые хранят в себе, преимущественно, большие объёмы данных. Базы данных активно используются для динамической загрузки данных.

В базе данных хранится одна сущность для хранения информации, введённой пользователем. Схема сущности представлена в таблице 1.

Таблица 1 – Схема сущности «TABLE_NOTES»

Содержание поля	Имя поля	Тип, длина	Примечания
ID-записки	_id	Int(11)	Первичный ключ
Заголовок	title	nvarchar(MAX)	Не обязательно поле
Содержимое записки	note	nvarchar(MAX)	Не обязательно поле
Дата создание записки	created_at	Date	Обязательно поле

Для работы с базой реализуется класс DBHelper. Этот класс даёт возможность получать данные из базы данных, а также обновлять её версию.

3.2 Описание разработанных процедур и функций

3.2.1 Добавление, удаление и редактирование данных

Для данных функций был создан класс, создающий базу данных в приложении. При загрузке главной активности создаётся база данных, реализованная в классе DBHelper (листинг 1) в которую можно вносить данные и редактировать их с помощью различных SQL-запросов (листинг 2).

Листинг 1 — Класс DBHelper для работы с базой данных

```
public class DBHelper extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION=1;
    public static final String DATABASE_NAME="notesDB";
    public static final String TABLE_NOTES="notes";

    public static final String KEY_ID ="_id";
    public static final String KEY_TITLE ="title";
    public static final String KEY_NOTE ="note";
    public static final String KEY_CREATED_AT = "created_at";

    public DBHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CreateTable =
                "create table "
                        + TABLE_NOTES + "("
                        + KEY_ID + " integer primary key,"
                        + KEY_TITLE + " text,"
                        + KEY_NOTE + " text,"
                        + KEY_CREATED_AT + " text"
                        + ")";

        sqLiteDatabase.execSQL(CreateTable);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int
newVersion) {
        sqLiteDatabase.execSQL("drop table if exists " + TABLE_NOTES);

        onCreate(sqLiteDatabase);
    }
}
```

Листинг 2 — Примеры запросов SQL

```
if (Memory.DeleteChek){
    String querydelete = "DELETE " + " FROM " + DBHelper.TABLE_NOTES + "
WHERE " + DBHelper.KEY_ID + " = " + (Memory.EditID);
    database.execSQL(querydelete);
    try {
```

```
        String idGenerate = "UPDATE " + DBHelper.TABLE_NOTES + " SET  _id
=  id - " + 1 + " WHERE _id " + " > " + Memory.EditID;
        @SuppressLint("Recycle") Cursor cursor = database.rawQuery(idGenerate,
null);
        if (cursor.moveToFirst()) {
            int editidIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
            if (editidIndex != 0){
                database.execSQL(idGenerate);
            }
        }
    } catch (Exception exception){
        Log.d("mLog",exception.toString());
    }
    Memory.DeleteChek = false;
}
searchView.setQuery("", false);
String queryload = "SELECT " + DBHelper.KEY_ID + ", "
        + DBHelper.KEY_TITLE + ", " + DBHelper.KEY_NOTE + ", " +
DBHelper.KEY_CREATED_AT + " FROM " + DBHelper.TABLE_NOTES;
@SuppressLint("Recycle")
Cursor cursor = database.rawQuery(queryload, null);
CreateRecycleView(cursor);
```

3.2.2 Отображение данных

Для реализации были созданы несколько классов для работы с базой данных и отображения заметок. Из базы данных с помощью SQL-запроса (листинг 3) выбираются данные с помощью объект класса Cursor, после чего вызывается процедура создания списка записок (листинг 4).

Листинг 3 — Запрос выбора данных из базы

```
String queryload = "SELECT " + DBHelper.KEY_ID + ", "
        + DBHelper.KEY_TITLE + ", " + DBHelper.KEY_NOTE + ", " +
DBHelper.KEY_CREATED_AT + " FROM " + DBHelper.TABLE_NOTES;
@SuppressLint("Recycle")
Cursor cursor = database.rawQuery(queryload, null);
CreateRecycleView(cursor);
```

Листинг 4 — Процедура вывода списка заметок на экран

```
public  void CreateRecycleView(Cursor cursor){

    List<Model> mModelList = new ArrayList<>();

    if (cursor.moveToFirst()) {
        int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
        int noteIndex = cursor.getColumnIndex(DBHelper.KEY_NOTE);
        int titleIndex = cursor.getColumnIndex(DBHelper.KEY_TITLE);
        int dateIndex = cursor.getColumnIndex(DBHelper.KEY_CREATED_AT);
        do {
            Log.d("mLog",
                    "ID = " + cursor.getInt(idIndex)
                            + ", title = " + cursor.getString(titleIndex)
                            + ", note = " + cursor.getString(noteIndex)
                            + ", date = " + cursor.getString(dateIndex));

            int IdI = cursor.getInt(idIndex);
            String TitleS;
            String NoteS;
            String DateS;
            char a;
            int checker = 0;
            for (int i = 0; i < cursor.getString(noteIndex).length(); i++){
                a = cursor.getString(noteIndex).charAt(i);
                if (a == '\n'){
                    checker=i;
                    break;
                }
            }

            if (checker != 0)
                NoteS = cursor.getString(noteIndex).substring(0, checker) + "...";
            else if (cursor.getString(noteIndex).length() > 25){
                NoteS = cursor.getString(noteIndex).substring(0, 25) + "...";
            } else {
                NoteS = cursor.getString(noteIndex);
            }

            DateS = cursor.getString(dateIndex);

            if (cursor.getString(titleIndex).length() > 25)
                TitleS = cursor.getString(titleIndex).substring(0, 25) + "...";
            else
```

```
            TitleS = cursor.getString(titleIndex);
            mModelList.add(new Model(TitleS,NoteS, DateS, IdI));
            Memory.EditID = cursor.getInt(idIndex);

        } while (cursor.moveToNext());
    }

    RecyclerView recyclerView = findViewById(R.id.LayoutForCard);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    adapter = new NoteAdapter(mModelList, item );
    adapter.setClickListener(this::onItemClick);
    recyclerView.setAdapter(adapter);
}
```

3.2.3 Создание карточек с данными

В основной активности создана процедура, представленная на рисунке 19, позволяющая создать и отобразить карточки с данными . В процедуре создаётся модель данных (листинг 5) которая заполняется результатом запроса SQL. Данные отправляются в адаптер элемента (листинг 6) в котором создаются карточки. В элементы карточки записываются данные из запроса, после чего карточка создаётся на слое. Также адаптер позволяет получить позицию карточки для дальнейшей работы с данными.

Листинг 5 — Класс Model

```
private String tData;
private String nData;
private String dData;
private Integer iData;
public static boolean isSelected = false;

public Model(String tData, String nData, String dData, Integer iData) {
    this.tData = tData;
    this.nData = nData;
    this.dData = dData;
    this.iData = iData;
}

public String getTitle() {
    return tData;
```

```
}
public String getNote() {
   return nData;
}
public String getDate() {
   return dData;
}
public Integer getid() {
   return iData;
}

public static void setSelected(boolean selected) {
   isSelected = selected;
}

public static boolean isSelected() {
   return isSelected;
}
```

Листинг 6 — Класс NoteAdapter

```
public class NoteAdapter extends
RecyclerView.Adapter<NoteAdapter.ViewHolder> {

   private View item;
   private List<Model> mModelList;
   private ItemClickListener mClickListener;

   // data is passed into the constructor
   NoteAdapter(List<Model> modelList, View itemView) {
      item = itemView;
      mModelList = modelList;
   }
   // inflates the row layout from xml when needed
   @NonNull
   @Override
   public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
      View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_note, parent, false);
      return new ViewHolder(view);
   }
   // binds the data to the TextView in each row
   @Override
   public void onBindViewHolder(ViewHolder holder, int position) {
      final Model model = mModelList.get(position);
```

```
        holder.TitleTV.setText(model.getTitle());
        holder.NoteTV.setText(model.getNote());
        holder.DateTV.setText(model.getDate());
    }

    // total number of rows
    @Override
    public int getItemCount() {
        int len = mModelList.size();
        if (len > 0)
        {
            return mModelList.size();
        } else{
            return 0;
        }
    }
    // stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
        TextView TitleTV;
        TextView NoteTV;
        TextView DateTV;
        TextView idTV;

        ViewHolder(View itemView) {
            super(itemView);
            idTV = itemView.findViewById(R.id.idTV);
            TitleTV = itemView.findViewById(R.id.TitleTV);
            NoteTV = itemView.findViewById(R.id.NoteTV);
            DateTV = itemView.findViewById(R.id.DateTV);
            itemView.setOnClickListener(this);

        }

        @Override
        public void onClick(View view) {
            if (mClickListener != null) mClickListener.onItemClick(view,
getAdapterPosition());
        }
    }
    Integer getItem(int position) {
        final Model model = mModelList.get(position);
        return model.getid();
    }
    void setClickListener(ItemClickListener itemClickListener) {
```

```
        this.mClickListener = itemClickListener;
    }
    public interface ItemClickListener {
        void onItemClick(View view, int position);
    }
}
```

Таким образом, используя запросы выбора данных из локальной базы данных, модель данных и адаптеры для их обработки, производится вывод информации в карточки, которые располагаются на элементе RecyclerView главной активности.

4 ТЕСТИРОВАНИЕ

Тестирование дизайна будет проводиться на самом минимальном (Android API 29) и на более позднем (Android API 30) с разными размерами экранов, чтобы проверить вёрстку приложения.

Каждый экран успешно прошёл проверку на наличие грамматических ошибок. Также каждый экран прошёл проверку на разных API на корректное отображение элементов соответствующих экранов. Примеры проверок отображения элементов на экране представлены на
рисунках 17, 18, 19, 20, 21 соответственно.

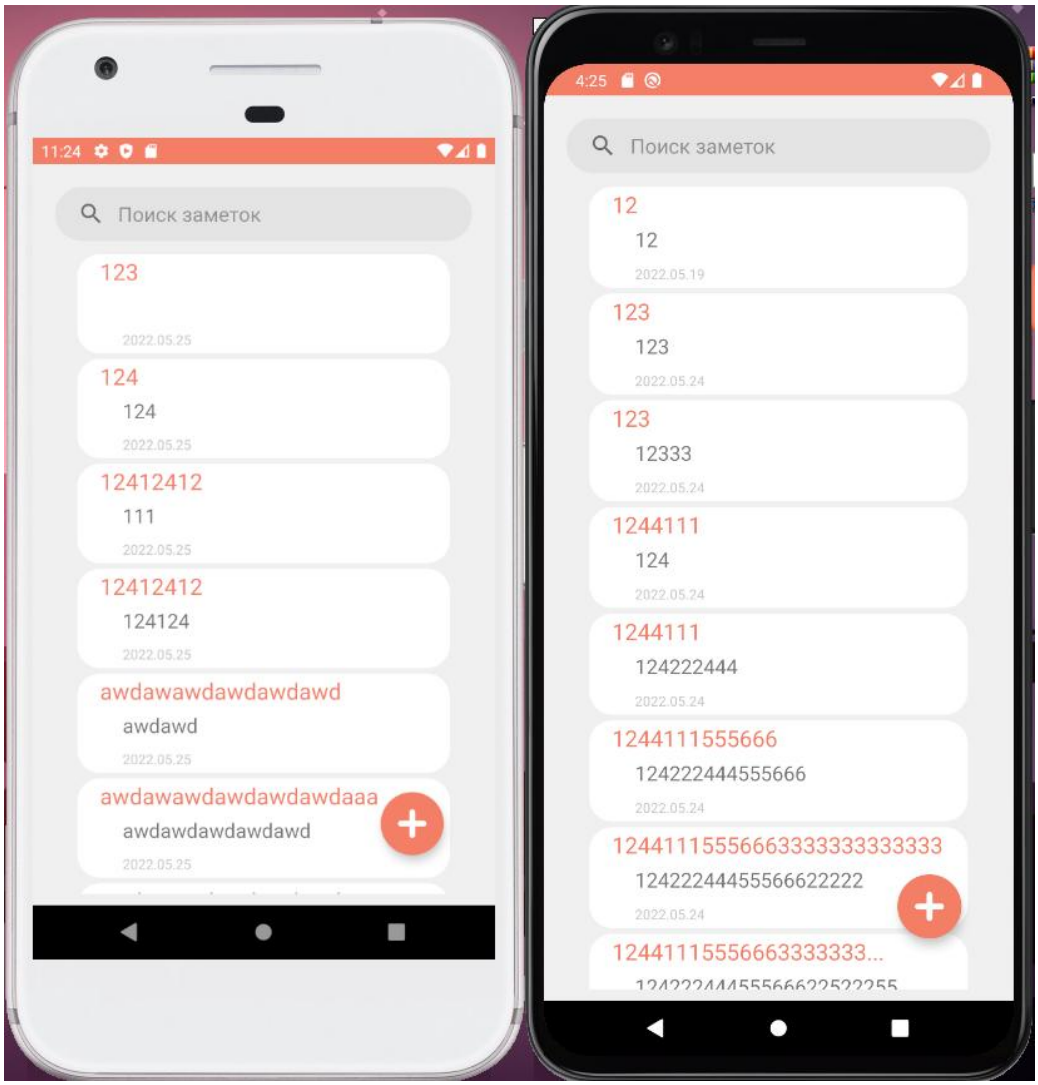

Рисунок 17 — Главный экран на 30 и 29 API соответственно

Рисунок 18 — Экран редактирования заметки (аналогично добавление) на 30 и 29 API соответственно

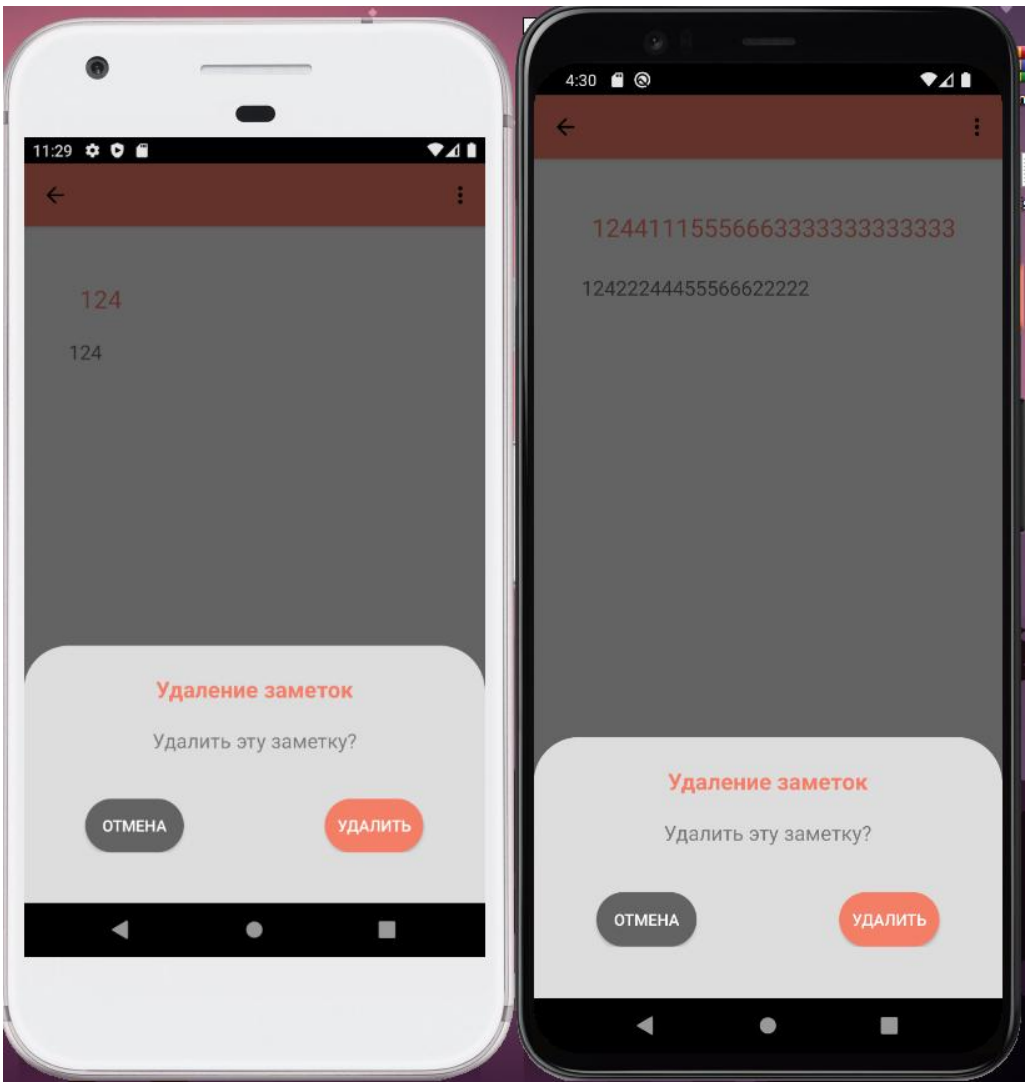

Рисунок 19 — Диалоговое окно «Удаление заметок» на 30 и 29 API соответственно

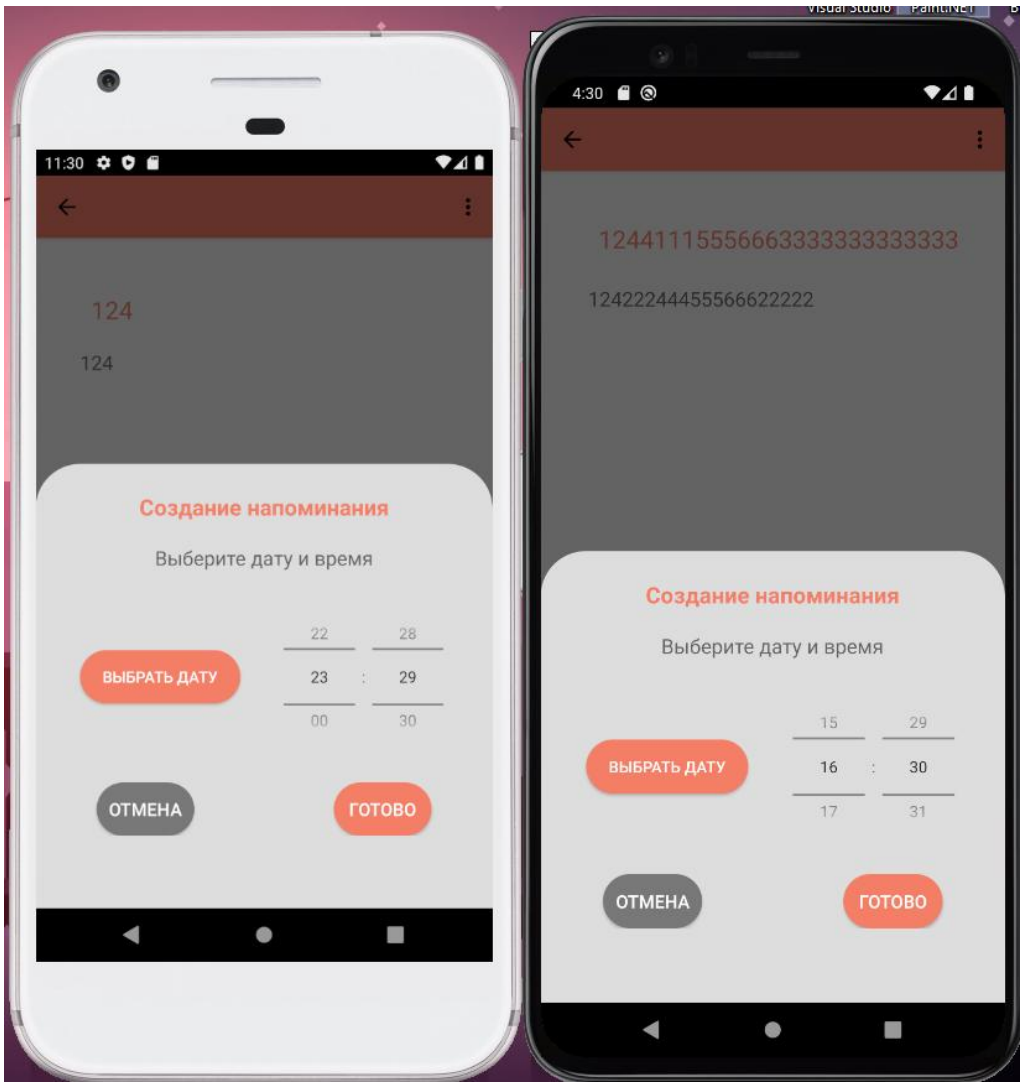

Рисунок 20 — Диалоговое окно «Создание напоминания» на 30 и 29 API соответственно

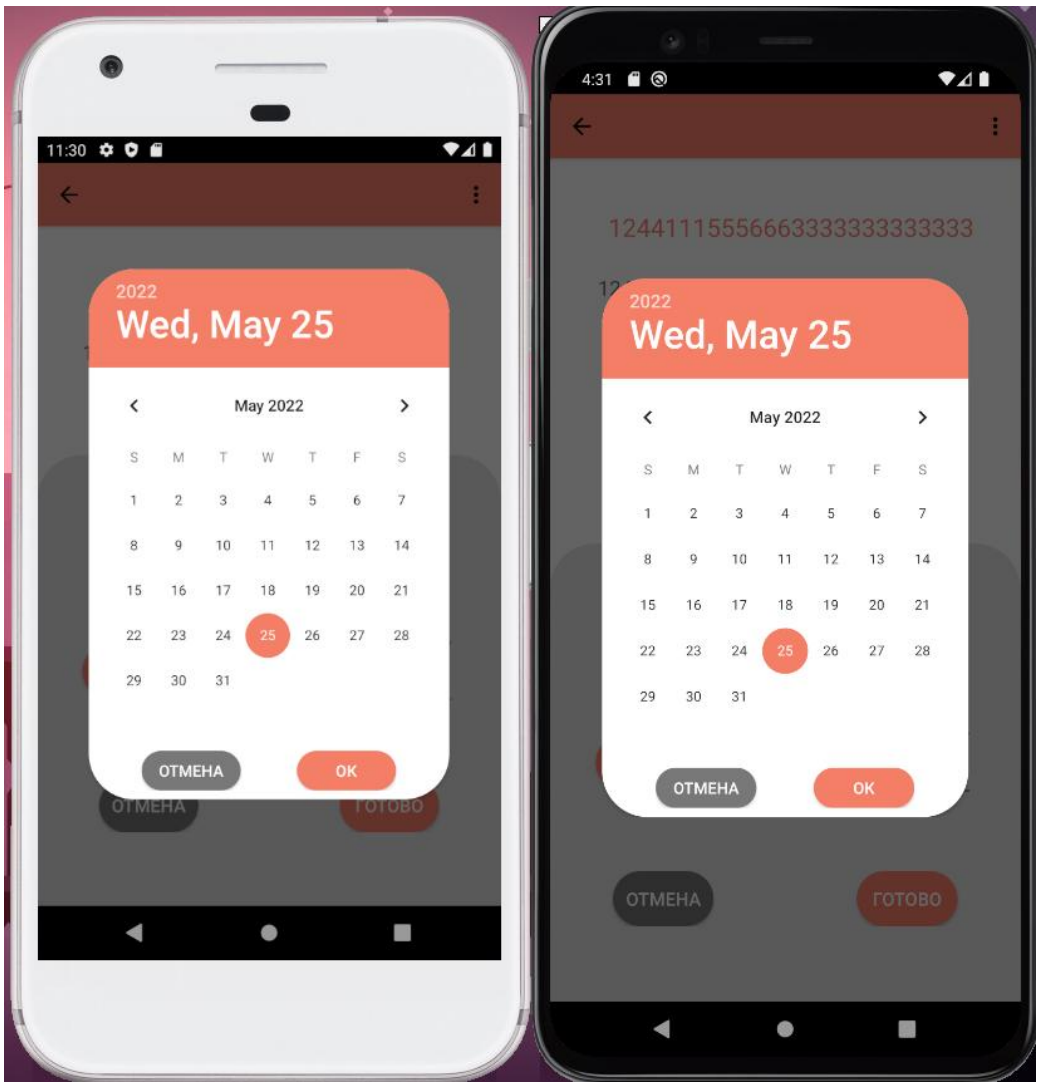

Рисунок 21 — Диалоговое окно «Календарь» на 30 и 29 API соответственно

В процессе тестирование не было выявлено критических ошибок в дизайне приложения ни на 29, ни на 30 API .

ЗАКЛЮЧЕНИЕ

В ходе исследования предметной области было выяснено, что экономия времени и удобство работы с данными является преобладающим фактором. Чтобы упростить процесс работы с данными, был определён функционал. Этот функционал позволит хранить данные и работать с ними. Рассмотренные приложения перегружены функциями и было принято решение написать более простое приложение.

Приложение, решающее проблемы хранения данных, сокращает затраты времени на запись данных для дальнейшего их хранения. Мобильное устройство есть почти у всех и преимущественно на устройствах установлена ОС Android различных версий, в связи с чем для разработки функционала приложения выбрана среда разработки IDE Android Studio и язык программирования Java.

С целью реализации простого и доступного дизайна определены и разработаны основные экраны приложения. В качестве языка разметки выбран язык, встроенный в IDE Android Studio, XML. Экраны имеют минималистичный дизайн, что упрощает работу с приложением.

Для возможности хранения всех необходимых записей пользователя разработана локальная база данных SQLite. SQLite хранится локально на устройстве и занимает мало памяти. В данной базе данных легко работать с данными, т.к. имеется несколько способов их обработки, такие как библиотека Room или же SQL-запросы.

СПИСОК ИСТОЧНИКОВ

1. Save data using SQLite [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/training/data-storage/sqlite

2. RecyclerView.Adapter [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter

3. PopupWindow | Android Developers [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/reference/android/widget/PopupWindow

4. DatePicker | Android Developers[Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/reference/android/widget/ DatePicker

5. TimePicker|Android Developers [Электронный ресурс]: Режим доступа к руководству:https://developer.android.com/reference/android/widget/TimePicker

6. Shape | Android Developers [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/reference/android/graphics/drawable/shapes/Shape

7. Schedule alarms | Android Developers [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/reference/android/app/AlarmManager

8. AlarmManger | Android Developers [Электронный ресурс]: Режим доступа к руководству: https://developer.android.com/training/scheduling/alarms

Приложение А

Техническое задание

ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку мобильного приложения «Разработка мобильного приложения для создания интерактивных заметок», используемого для хранения информации.

Наименование программы: «DFT». Далее по тексту Приложение.

Краткая характеристика области применения: мобильное приложение предоставляет возможность хранить данные без больших затрат во времени.

1 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Основание для проведения разработки является Приказ №уч-041/4 от 17 марта 2022 года.

Наименование темы разработки − «Разработка мобильного приложения для создания интерактивных заметок».

2 НАЗНАЧЕНИЯ ДЛЯ РАЗРАБОТКИ

Функциональным назначением приложения является обеспечение удобного и эффективного интерфейса для пользователя, а также возможность установки даты напоминания информации.

3 ТРЕБОВАНИЕ К ПРИЛОЖЕНИЮ

3.1 Требования к функциональным характеристикам

Система должна обеспечивать возможность выполнения нижеперечисленных функций, описанных в таблице А.1.

Таблица А.1 – Выполняемые функции приложения

Номер	Функция
1	Добавление, редактирование удаление информации в заметке
2	Добавление, удаление заметки
3	Смена темы приложения
4	Отображение Push-уведомлений

3.2 Требования к надёжности

Требования к надёжности не предоставляются

3.3 Условия эксплуатации

Пользователь должен иметь практические навыки использования мобильного устройства под управлением операционной системы Android.

3.4 Требования к составу и параметрам технических средств

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android не ниже версии 10.0.

3.5 Требования к информационной и программной совместимости

Разработка приложения ведется на языке программирования Kotlin.

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android с версией SDK не ниже 29.

3.6 Требования к защите информации

Требования не представляются.

3.7 Требования к маркировке и упаковке

Требования не представляются.

3.8 Специальные требования

Специальные требования не предоставляются.

4 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать в себя:

- техническое задание;
- пояснительная записка.

5 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

6 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица А.2 – Стадии разработки

Этапы разработки КП	Срок выполнения	Отчётность
Определение цели и задач, объекта и предмета исследования	26.03.2022	Пояснительная записка
Описание предметной области	02.04.2022	Пояснительная записка
Выбор технологии, языка и среды программирования	09.04.2022	Пояснительная записка
Оформление технического задания	16.04.2022	Пояснительная записка
Проектирование UI/UX дизайна	23.04.2022	Пояснительная записка
Разработка мобильного приложения	30.04.2022	Пояснительная записка
Разработка базы данных	07.05.2022	Пояснительная записка
Отладка и тестирование приложения	14.05.2022	Пояснительная записка
Оформление документации	21.05.2022	
Защита	28.05.2022	

7 ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ

Виды испытаний – защита проекта.

Общее требования к приёмке:

- техническое задание;
- пояснительная записка;
- программный продукт.