|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «Разработка мобильных приложений»

**Тема курсовой работы**

Мобильное приложение «Учет финансов и ведение семейного бюджета»

**Студент группы** ИКБО-20-20 Матвеев И. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель курсовой работы** доцент Петренко А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена к защите «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Допущен к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Москва 2022

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Утверждаю

Заведующий кафедрой МОСИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Головин С.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы по дисциплине**

«Разработка программных приложений»

Студент Матвеев Иван Алексеевич Группа ИКБО-20-20

**Тема работы:** Мобильное приложение «Учет финансов и ведение семейного бюджета»

**Исходные данные:** задание на курсовую работу, тема курсовой работы.

**Функционал разрабатываемого программного приложения должен** обеспечивать необходимую функциональность, которая формируется в зависимости от заданной темы и предметной области изучаемых вопросов, c предоставлением основных возможностей интерактивных систем с понятным дружественным графическим интерфейсом.

**Перечень подлежащих описанию в работе вопросов и обязательного графического материала:**

- инсталляция и настройка операционной системы Android с применением виртуальных сред и других необходимых для выполнения работы сред и систем программирования: Android Studio, эмуляторов Android, IntelliJ IDEA;

- предъявление к разрабатываемому программному приложению требований;

- написание технического задания на разработку программного приложения;

- описание жизненного цикла программного приложения и технологии его создания с применением языка программирования JAVA;

- реализация в создаваемом программном приложении хранения данных в виде файлов или баз данных, обеспечение их создания, чтения, записи во внутреннем и внешнем хранилище;

- обоснование возможности реализации и реализация клиентской части программного приложения на смартфоне или планшетнике через WAP или подобные ей системы;

- обеспечение безопасности информации при работе разрабатываемого программного приложения.

- тестирование и диагностика созданного программного продукта;

- описание возможности портирования написанной программной системы на внешние хосты в сети Интернет;

- отчет по курсовой работе в виде расчетно-пояснительной записки с рисунками, таблицами и графиками (при необходимости).

**Срок представления к защите курсовой работы:** до «5» июня 2022 г.

**Задание на курсовую работу выдал** \_\_\_\_\_\_\_\_\_\_\_\_ Петренко А.А.

«15» февраля 2022 г.

**Задание на курсовую работу получил**  \_\_\_\_\_\_\_\_\_\_\_\_ Матвеев И.А.

# **АННОТАЦИЯ**

В данном отчёте описан процесс реализации курсовой работы на тему Мобильное приложение «Учет финансов и ведение семейного бюджета». Данное направление разработки было выбрано в связи с неуменьшающейся потребностью людей в контроле своих финансов, что является финансовой грамотностью.

Главной целью данной работы являлась разработка приложения для ведения учета финансов. В задачи данного проекта входили: выбор оптимальной цветовой палитры, создание макета с учётом удобства взаимодействия с программными компонентами, выбор средств и способов реализации данной задумки, создание разметки и функционала приложения, выбор организации сохранения заметок и реализация самого сохранения.

В 1-ой главе представлено описание предварительной подготовки – анализ приложений с аналогичной тематикой для определения функционала приложения и выбор программного обеспечения для работы над проектом.

Во 2-ом разделе представлено описание идей внешнего вида приложения (цвета, количество явлений, желаемые атрибуты явлений) и показан результат работы над макетом будущего приложения. Также рассказано о процессе отбора графических материалов для приложения.

В 3-ей и 4-ой главах рассказывается о самом процессе создания, начиная от подготовки макета и заканчивая программной реализацией появляющихся панелей инструментов.

В 5-ом разделе представлены рассуждения о наиболее выгодном способе хранения заметок для данного приложения, описаны структуры, используемые для преобразований данных к определённому формату.

Как итог было разработано рабочее приложение, выполняющее все необходимые функции, возможности которого продолжают улучшаться.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 3](#_Toc103183795)

[ВВЕДЕНИЕ 5](#_Toc103183796)

[ОСНОВНАЯ ЧАСТЬ 6](#_Toc103183797)

[1. Подготовка материалов по разрабатываемому приложению 6](#_Toc103183798)

[1.1. Сбор информации 6](#_Toc103183799)

[1.2. Анализ приложения «noteit» 6](#_Toc103183800)

[1.3. Анализ приложения «Floating Notes» 6](#_Toc103183801)

[1.4. Итоги анализа предметной области 7](#_Toc103183802)

[1.5. Определение необходимого функционала 8](#_Toc103183803)

[1.6. Определение желательного, но не обязательного функционала 9](#_Toc103183804)

[1.7. Определение необходимого программного обеспечения для разработки приложения 9](#_Toc103183805)

[2. Разработка макета будущего приложения 11](#_Toc103183806)

[2.1. Определение необходимых явлений 11](#_Toc103183807)

[2.2. Создание mindmap’а приложения 11](#_Toc103183808)

[2.3. Создание дизайна явлений приложения 12](#_Toc103183809)

[2.4. Определение оптимальных цветов для приложения 13](#_Toc103183810)

[2.5. Создание графических элементов для приложения 14](#_Toc103183811)

[3. Создание основы приложения 15](#_Toc103183812)

[4. Создание функционала для рисования 18](#_Toc103183813)

[5. Выбор способа и организация сохранения 22](#_Toc103183814)

[ЗАКЛЮЧЕНИЕ 25](#_Toc103183815)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 26](#_Toc103183816)

# **ВВЕДЕНИЕ**

Люди все чаще стараются следить за своими финансами, в этом им помогают и мобильные приложения, в которых можно очень удобно это делать. Данная тема была выбрана мной, так как данное приложение может стать идеальным помощником для людей в построении их финансовой грамотности.

Целью данной работы является создание приложения для учета финансов, которое предоставило бы пользователям удобный интерфейс для фиксирования финансов.

Задачи, которые предстоит решить для достижения данной цели:

* сбор информации о данном типе приложений и анализ конкурентов;
* определение и установка программного обеспечения для разработки мобильного приложения;
* создание макета приложения;
* определение и подбор оптимальных тем и цветов для приложения;
* реализация графического интерфейса пользователя (дизайн приложения);
* создание взаимодействий между явлениями и фрагментами, определение зависимостей;
* реализация сохранения записей на устройстве пользователя;
* реализация возможности создания новых или редактирование уже существующих заметок на устройстве друга;
* реализация возможности рисовать в заметках;
* тестирование и отладка приложения.

# **ОСНОВНАЯ ЧАСТЬ**

## **Подготовка материалов по разрабатываемому приложению**

## **Сбор информации**

Приложение для учета финансов и ведения семейного бюджета предоставляет пользователю удобный функционал. Для определения необходимых для приложения функций проанализируем конкурентов, после чего выделим общие и особые возможности.

На данную тематику мне удалось найти и протестировать работу нескольких приложений, а именно:

* CoinKeeper3: финансы, бюджет [1];
* Monefy – учет расходов [2].

## **Анализ приложения «CoinKeeper3: финансы, бюджет»**

После первого запуска приложение «Coinkeeper3: финансы, бюджет» предлагает создать новый профиль или войти в уже существующий. Предлагается также включить уведомления, чтобы постоянно получать отклик от приложения.

Приложение позволяет рассчитывать бюджет, опираясь на операции из банков. Есть возможность создать разные счета, например: банковский счет 1 – Сбербанк, наличные средства, электронные кошельки 1 – Qiwi и т.д. Также присутствует функционал советов для того, чтобы люди могли получить помощь на пути к их финансовой грамотности (статьи, курсы, допольнительные инструменты приложения).

## **Анализ приложения «Monefy – учет расходов»**

При первом запуске приложения «Monefy – учет расходов» открывается диаграмма со всеми тратами и возможностью сразу записать в определенную категорию трату. Присутствует функционал добавления дохода снизу от диаграммы.

Это приложение позволяет добавлять к своим доходам или расходам заметки, вносить новые расходы или доходы, которые повторяются, при помощи одной кнопки, не тратя лишнее время на написание, создавать собственные категории трат и доходов.

Функционал для обычных пользователей «Monefy – учет расходов» сильно ограничен, и, чтобы его расширить, нужно приобрести платную подписку.

## **Итоги анализа предметной области**

При подведении итогов анализа было выявлено, что оба приложения имеют свои достоинства и недостатки. Объединю выводы, полученные путём личного использования приложений, в таблицы 1.4.1-1.4.2.

Таблица 1.4.1 – Достоинства конкурентов

|  |  |
| --- | --- |
| «Coinkeeper3: финансы, бюджет» | «Monefy – учет расходов» |
| Легкое и понятное создание разных счетов (наличные, банковские карты и т.д.) | Наглядная диаграмма с возможностью быстро записать расход или доход |
| Современный интерфейс приложения | Удобная работа с категориями |
| Статистика расходов и доходов похожа на статистики, реализованные в банковских приложениях | Возможность добавления заметки к определенному расходу или доходу |
| Функционал советов, позволяющий людям погрузиться в мир финансовой грамотности более детально и вызвав желание пользоваться приложением чаще | Простой и понятный интерфейс |

Таблица 1.4.2 – Недостатки конкурентов

|  |  |
| --- | --- |
| «Coinkeeper3: финансы, бюджет» | «Monefy – учет расходов» |
| Нет возможности поменять валюту в мобильном приложении, для этого нужно воспользоваться ПК и потратить лишнее время и иметь возможность получить в данный момент доступ к ПК | Нет возможности быстро создать разные счета |
| Нагруженный интерфейс, из-за чего иногда нужно долго искать нужную функцию в приложении | Нет возможности добавить к расходу или доходу определенную дату |

## **Определение необходимого функционала**

На основе данных, приведённых в таблицах 1.4.1-1.4.2, определим, что обязательно должно быть в приложении и что хотелось бы видеть в нём. В приложении должен присутствовать следующий ряд функций:

* добавление расходов и доходов;
* удаление расходов и доходов;
* создание категорий;
* создание основной диаграммы с расходами и доходами;
* добавление заметки к расходу или доходу.

## **Определение желательного, но не обязательного функционала**

Функции, которые хотелось бы видеть в приложении:

* создание различных счетов.
* изменение темы приложения (светлая и тёмная);
* реализация смены валюты.

## **Определение необходимого программного обеспечения для разработки приложения**

Для удобной разработки и корректной работы приложения на устройстве пользователя ниже представлен список минимальных требований, которые помогут начать разработку и трезво оценить её результат.

Для разработки данного мобильного приложения требуется:

* Мобильное устройство с операционной системой Android 6.0 (SDK version >= API level 23 (Marshmellow)).

В качестве устройства для тестирования будет использоваться виртуальное устройство Pixel 2 с операционной системой Android API level 31. В дальнейшем планируется тестирование на реальном устройстве Samsung Galaxy S20 с операционной системой Android.

Для разработки данного приложения будет использоваться следующий ряд программных продуктов, устройств и типов подключений:

* мобильное устройство с системой Android (описано выше);
* компьютер с операционной системой macOS Monterey;
* доступ к интернету;
* интегрированная среда разработки Android Studio Bumblebee | 2021.1.1 [3] для работы с платформой Android и создания приложений под неё;
* интеллектуальная интегрированная среда разработки Intellij Idea [4], предназначенная для разработки разнообразных приложений на Java и других языках JVM;
* графический редактор Figma [5] для совместного проектирования сайтов, приложений и других дизайнерских продуктов;
* поисковые системы «Google» [6] и «Yandex» [7];
* источник бесплатных для использования иконок Flaticon [8];
* программа для создания векторных иллюстраций Adobe Illustrator 2020 [9].

## **2. Разработка макета будущего приложения**

## **2.1. Определение необходимых явлений**

Для удобной работы в приложении оптимальное число явлений равно двум:

* главное явление – позволит пользователю выбирать расход или доход для редактирования, создавать новый или удалять уже существующий;
* явление дохода или расхода – позволит пользователю редактировать выбранный расход или доход;

Также допустимо добавление ещё одного явления – настройки.

## **2.2. Создание дизайна явлений приложения**

Перед тем как начать работать, необходимо подготовить дизайн-макет будущего приложения (рис. 2.2.1). Данный макет является приблизительным представлением внешнего вида будущего приложения, так что в дальнейшим будут видны различия между отрисованным в Figma и реальным дизайнами приложения.



Рис. 2.2.1 – Первоначальный макет приложения

## **2.3. Определение оптимальных цветов для приложения**

Данное приложения разрабатываться для учета финансов, что означает, что пользователю должно быть комфортно следить за состоянием финансов, их направлением (уменьшение или рост). Для достижения данной задачи было решено использовать нейтральный неагрессивный цвет – зеленый – и его оттенки (рис. 2.3.1).

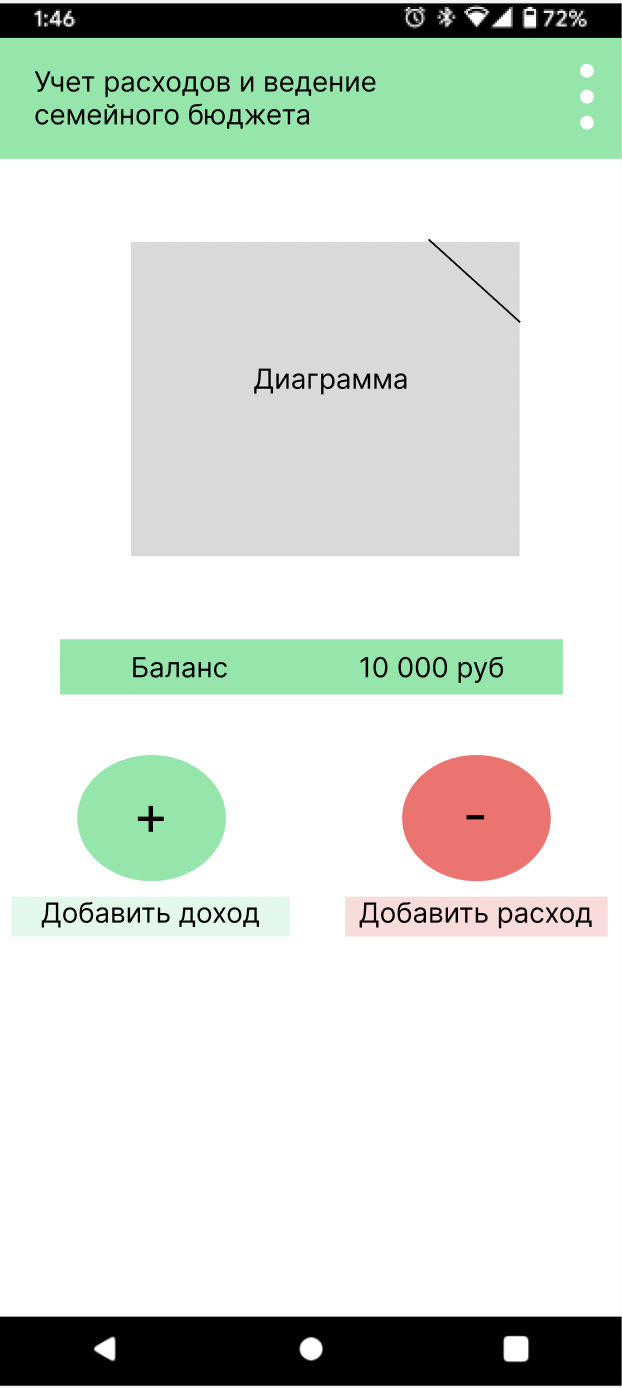


Рис. 2.3.1 – Цвета, определяющие светлую тему приложения

## **2.4. Создание графических элементов для приложения**

В данном приложении пользователь сможет выбрать пин (иконку) для заметки, чтобы ему было удобно различать её содержимое. Иконки были скачаны с сайта Flaticon. Из 816 иконок вручную были отобраны и перекрашены 68. Также в Illustrator’е были созданы иконки для панелей инструментов.

## **3. Создание основы приложения**

Перед тем как начать работать над возможностями приложения, необходимо подготовить разметку явлений. Разметка явления для создания новой и выбора старой заметки для редактирования и удаления (рис. 3.1) содержит:

* строку поиска (устаревшее: с кнопкой поиска; новое: без кнопки, так как она не несёт смысловой нагрузки);
* место для заметок (устаревшее: LinearLiyout, новое: RecycleView);
* плавающую кнопку для добавления новой заметки.

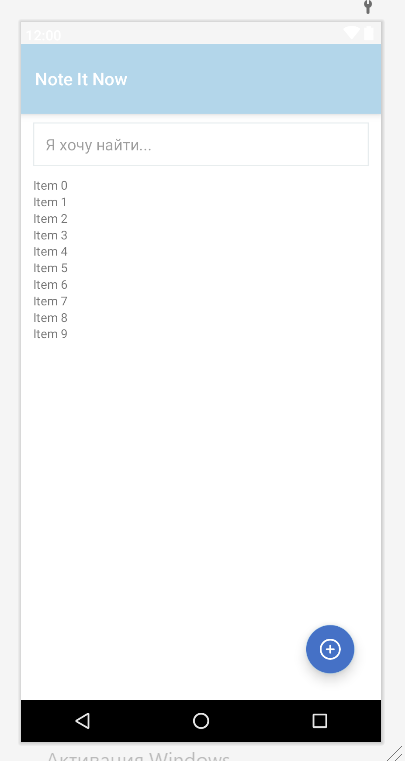


Рис. 3.1 – Разметка главного явления

Явление, содержащее конкретную заметку, имеет поле для ввода названия, поле содежимого заметки, кнопку выбора пользовательского значка (служит для удобства пользователя, функциональной нагрузки не несёт), а также панель инструментов, которая содержит кнопку добавления графической заметки (рис. 3.2).

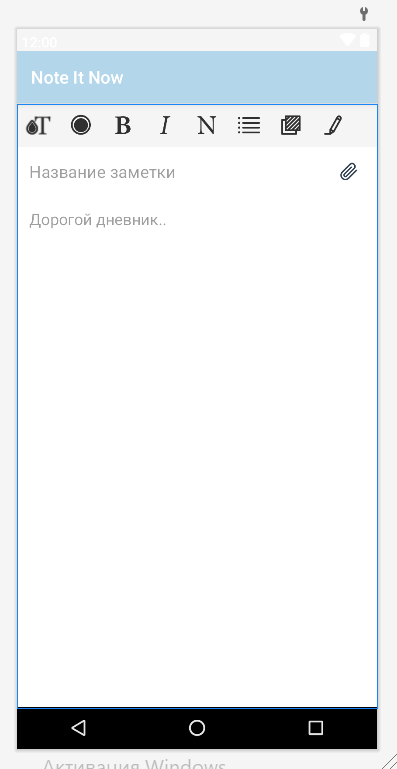


Рис. 3.2 – Явление для работы с конкретной заметкой

Явление для работы с графической заметкой содержит холст и панель инструментов для выбора кистей и цветов (рис. 3.3).

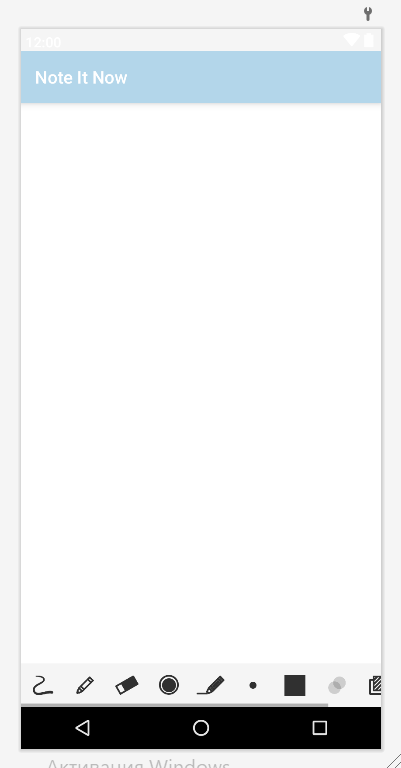


Рис. 3.3 – Явление для работы с графической заметкой

Также на ранней стадии работы сразу были заготовлены методы для программного создания новой заметки. Данный код является устаревшим (первоначальной версией, от которой было решено отказаться), поэтому здесь он представлен не будет, но его можно посмотреть в ранних commit’ах на GitHub’е [11].

Для удобной работы с определёнными переменными и временными ресурсами были созданы два дополнительных статических (все методы и переменные имеют модификатор static) класса: PublicResources и TempResources. Первый класс содержит поля и методы, которые используются различными активностями и имеют константный характер. Второй класс используется для удобной работы с конкретной заметкой для исключения потребности постоянного запроса данных о заметке.

## **4. Создание функционала для рисования**

Для реализации рисования потребуется два класса (для визуального удобства): класс явления (CanvasActivity), которое позволяет пользователю взаимодействовать с холстом, и класс с самим холстом (DrawingView), инструментами для рисования и цветами.

В классе CanvasActivity на полученные из разметки по ID явления элементы панели инструментов были установлены обработчики кликов, которые вызывали определённые методы работы с объектом draw\_view класса холста. Перед настройкой кистей и цветов были созданы методы для динамического появления панели цветов и других элементов при клике на определённый элемент (Листинг 4.1).

Листинг 4.1. – Динамическое создание панели выбора цвета

*// класс CanvasActivity*

*// выбор цвета для рисования***private void** setColorChooserForPanelElement(ImageButton panel\_element,  
 Doings panel\_element\_color) {  
 **if** (**items\_hsv**.getVisibility() == View.***VISIBLE***) {  
 clearPanelItems();  
 }  
 **items\_hsv**.setVisibility(View.***VISIBLE***);  
 LinearLayout ll = PublicResources  
 .*getLLPanelWithColors*(**inflater**, R.layout.***popup\_menu\_layout***, **colors**);  
 **items\_hsv**.addView(ll);  
 View.OnClickListener color\_cl = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *// Log.d(PublicResourсes.DEBUG\_LOG\_TAG, String.valueOf(view.getId()));* **draw\_view**.setColor(view.getId(), panel\_element\_color);  
 setCurrentColorOnPanel(panel\_element);  
 clearPanelItems();  
 }  
 };  
 **for** (**int** i = 0; i < ll.getChildCount(); ++i) {  
 (ll.getChildAt(i)).setOnClickListener(color\_cl);  
 }  
}

*// класс PublicResources*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* создание цветового элемента \*/***private static void** colorItem(LayoutInflater inflater, **int** child\_layout,  
 LinearLayout parent, **int** color) {  
 ImageButton btn = (ImageButton) inflater  
 .inflate(child\_layout, parent, **false**);  
 btn.setColorFilter(color, PorterDuff.Mode.***SRC\_IN***);  
 btn.setId(color);  
 parent.addView(btn);  
}  
  
*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* получение панели цветов \*/***public static** LinearLayout getLLPanelWithColors(LayoutInflater inflater, **int** layout,  
 **int**[] colors) {  
 LinearLayout ll = (LinearLayout) inflater  
 .inflate(layout, **null**, **false**);  
 LinearLayout.LayoutParams lp = **new** LinearLayout  
 .LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***,  
 ViewGroup.LayoutParams.***MATCH\_PARENT***);  
 lp.**gravity** = Gravity.***CENTER\_VERTICAL***;  
 *// ll.setId(TempResources.LINEAR\_LAYOUT\_ID);* ll.setLayoutParams(lp);  
 **for** (**int** color : colors) {  
 *colorItem*(inflater, R.layout.***popup\_menu\_item\_layout***, ll, color);  
 }  
 **return** ll;  
}

После настройки и отладки появления всех необходимых для работы панелей была начата работа над классом DrawingView. Было решено, что в качестве инструментов рисования будут выступать кисть, маркер и ластик, и для удобства пользователя настройки толщин и цветов кисти и маркера разделены.

Отслеживание и обработка действий пользователя будет происходить в основном потоке, так как в данном случае активность для рисования является отдельной и не имеет сложного функционала, что позволяет работать в основном потоке и не сказывается на отзывчивости интерфейса. Всего будет отслеживаться три вида действий – палец поставили на экран, палец двигает по экрану и палец убрали. Каждое из действий обрабатывается определённым образом в методах touchStart, touchInMove, touchEnd (Листинг 4.2). TOUCH\_TOLERANCE служит для определения расстояния, при приодолении которого нужно соединить линии. Это нужно, чтобы не соединять абсолютно каждый пиксель, на холсте (это может замедлить производительность, так как обработка касания была бы в 4 раза чаще, так как TOUCH\_TOLERANCE = 4, иухудшить вид линий).

Листинг 4.2 – Методы для обработки касаний холста

*// Касания***public void** touchStart(**float** x, **float** y) {  
 **draw\_path**.reset();  
 **draw\_path**.moveTo(x, y);  
 **this**.**x** = x;  
 **this**.**y** = y;  
}  
**public void** touchInMove(**float** x, **float** y) {  
 **float** dx = Math.*abs*(x - **this**.**x**),  
 dy = Math.*abs*(y - **this**.**y**);  
 **if** (dx >= ***TOUCH\_TOLERANCE*** || dy >= ***TOUCH\_TOLERANCE***) {  
 **draw\_path**.quadTo(**this**.**x**, **this**.**y**, (x + **this**.**x**) / 2, (y + **this**.**y**) / 2);  
 **this**.**x** = x;  
 **this**.**y** = y;  
 }  
}  
**public void** touchEnd(**float** x, **float** y) {  
 **draw\_path**.lineTo(x, y);  
 **canvas**.drawPath(**draw\_path**, **draw\_paint**);  
 **draw\_path**.reset();  
}

Ластик позволит стирать объекты, нарисованные на холсте, кисть – рисовать непрозрачной кистью, а маркер – создавать полупрозрачные линии. Результат обработки рисования представлен на рисунке 4.1.



Рис. 4.1 – Рисование на холсте

## **5. Выбор способа и организация сохранения**

Для организации хранения был выбран способ хранения файлов, так как работа с заметками подразумевает обработку большого объёма данных (в частных случаях). Для сохранения данных были введены два (улучшено: три класса, но функциональное использование третьего на стадии доработки) класса структур: структура заметки в целом и структура рисунка. Рисунок содержит цвет фона и Bitmap карту изображения, что отображено в структуре хранения рисунка: строковое поле названия файла с Bitmap картой и целочисленное значение цвета фона. Структура самой заметки содержит строковые поля названия и содержимого заметки и названия файла с представлением заметки в нотации Gson объекта, целочисленные значения цвета фона и ID ресурса иконки заметки, булевое значение закрепления заметки и массив структур рисунков, так как количество рисунков ограничено только объёмом внутренне памяти устройства.

Как было упомянуто выше, данные структуры нужны для преобразования данных заметки в нотацию Gson объекта для удобного хранения и получения. Методы преобразования структур и сохранения заметок в файл представлены в листинге 5.1. Сохранение рисунков происходит в отдельном потоке, чтобы сохранить отзывчивость интерфейса в случае, если данные будут долго записываться в файл.

Листинг 5.1 – Сохранение данных

*// методы структуры NoteStructure*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GSON \*/***public static** String getNoteToGsonString(NoteStructure note) {  
 Gson gson = **new** Gson();  
 String gson\_string = gson.toJson(note);  
 **return** gson\_string;  
}  
  
**public static** NoteStructure getNoteFromGson(String gson\_string) {  
 Gson gson = **new** Gson();  
 NoteStructure note = **null**;  
 **try** {  
 note = gson.fromJson(gson\_string, NoteStructure.**class**);  
 } **catch** (Exception e) {  
 Log.*d*(PublicResources.***DEBUG\_LOG\_TAG***, e.getMessage());  
 }  
 **return** note;  
}

*// класс PublicResources*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Работа с json \*/  
 // чтение json* **public static** NoteStructure parseGsonFromFile(String file\_path) {  
 String temp\_str;  
 String gson\_string = **""**;  
 File file = **new** File(file\_path);  
 *// Log.d(DEBUG\_LOG\_TAG, file.getAbsolutePath());  
 // чтение* **try** {  
 BufferedReader reader = **new** BufferedReader(**new** FileReader(file));  
 **while** ((temp\_str = reader.readLine()) != **null**) {  
 gson\_string += temp\_str;  
 }  
 *// Log.d(DEBUG\_LOG\_TAG, gson\_string);* } **catch** (Exception e) {  
 Log.*d*(PublicResources.***DEBUG\_LOG\_TAG***, e.getMessage());  
 }  
 **return** NoteStructure.*getNoteFromGson*(gson\_string);  
 }  
  
 *// добавить json (2)* **public static void** saveOrReplaceGson(String file\_path, String gson\_string) {  
 File file = **new** File(file\_path);  
 *// Log.d(DEBUG\_LOG\_TAG, file.getAbsolutePath());  
 // запись* **try** {  
 BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(file));  
 writer.write(gson\_string);  
 writer.close();  
  
 } **catch** (Exception e) {  
 Log.*d*(PublicResources.***DEBUG\_LOG\_TAG***, e.getMessage());  
 }  
 }  
  
 *// создать или получить существующий файл для записи рисунка* **public static** File createFile(String file\_name, Doings extension) {  
 File file = **null**;  
 **switch** (extension) {  
 **case *PNG***:  
 file = **new** File(*IMAGES\_DIR*, file\_name + PublicResources.***IMAGE\_PNG***);  
 **break**;  
 **case *GSON***:  
 file = **new** File(*NOTES\_DIR*, file\_name + PublicResources.***TEXT\_GSON***);  
 **break**;  
 }  
*// if (file != null) {  
// Log.d(DEBUG\_LOG\_TAG, file.getAbsolutePath());  
// }* **return** file;  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* удалить файл \*/* **public static boolean** deleteFile(File file) {  
 **return** file.delete();  
 }  
  
 *// распарсить рисунок из файла* **public static** Bitmap parseBitmapFromDrawingFile(String file\_path) {  
 File file = **new** File(file\_path);  
 **return** BitmapFactory.*decodeFile*(file\_path);  
 }

Код всего приложения представлен на Github’е.

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

* Floating Notes [Электронный ресурс]. Приложение для создания плавающих заметок. URL: <https://floatingnotes.com/> (Дата обращения 16.04.2022);
* noteit [Электронный ресурс]. Приложение для обмена заметками с партнёром. URL: <https://apps.apple.com/ru/app/noteit-drawing-app/id1570369625?uo=4> (Дата обращения 17.04.2022);
* Android Studio Chipmunk | 2022.2.1 [Электронный ресурс]. URL: <https://developer.android.com/studio> (Дата обращения 03.02.2022);
* IntelliJ IDEA [Электронный ресурс]. Функциональная и органичная IDE для JVM. URL: <https://developer.android.com/studio> (Дата обращения 26.09.2021);
* Figma [Электронный ресурс]. Онлайн сервис для дизайнеров. URL: <https://www.figma.com/> (Дата обращения 11.05.2022);
* Google [Электронный ресурс]. URL: <https://www.google.com/> (Дата обращения 13.03.2022);
* Yandex [Электронный ресурс]. URL: <https://yandex.ru/> (Дата обращения 11.05.2022);
* Flaticon [Электронный ресурс]. Наборы бесплатных иконок и стикеров. URL: <https://www.flaticon.com/> (Дата обращения 26.04.2022);
* Adobe Illustrator 2020 [Электронный ресурс]. Приложение для создания графики. URL: <https://www.adobe.com/ru/products/illustrator.html> (Дата обращения 11.05.2022);
* XMind [Электронный ресурс]. Идеи растут на деревьях. URL: <https://www.xmind.net/> (Дата обращения 11.05.2022);
* Исходный код приложения [Электронный ресурс]. Код приложения NoteItNow. URL: <https://github.com/Plintus-bit/NoteItNow> (Дата обращения 10.05.2022).