Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Колледж информатики и программирования

ПМ. 01. Разработка модулей программного обеспечения для компьютерных систем

Группа: 4ИСИП-519

УТВЕРЖДАЮ
Председатель предметно-цикловой комиссии информационных систем и программирования
_____ /Т.Г. Титов/
«__»_____2022 г.

ПРОЕКТ КУРСОВОЙ

На тему: «Мобильное приложение «Ежедневник»»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Руководитель курсового проекта
_____/М.В. Морозова/
Исполнитель курсового проекта
_____ /И.В. Берестнев/
Оценка за проект: _____
«___» ноября 2022 г.

Москва
2022

Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)

Колледж информатики и программирования

ОТЗЫВ РУКОВОДИТЕЛЯ НА КУРСОВОЙ ПРОЕКТ

Студента группы 4ИСИП-519

Берестнева Ивана Владимировича

Специальность: 09.02.07 Информационных систем и программирования

Тема курсового проекта: «Мобильное приложение «Ежедневник»»

Актуальность работы:

Отличительные положительные стороны работы:

Практическое значение работы:

Уровень сформированности компетенций, продемонстрированный в ходе работы над курсовым проектом (высокий, средний, низкий):

Отношение обучающегося к выполнению курсового проекта, проявленные/не проявленные им способности:

Степень самостоятельности обучающегося и его личный вклад в раскрытие проблемы, разработку предложений по их решению:

Недостатки и замечания:

Выводы:

Оценка:

Руководитель:
Морозова Мария Владимировна _____ / _____ /

Дата:

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

У каждого человека есть дела и обязанности, которые ему нужно завершить к определенному сроку. Но не всегда получается распределить их все по важности и срокам, и именно для этого существуют приложения для планирования своего времени.

Переоценить важность планирования своего времени в современном мире невозможно. От планировки своего времени зависит очень многое: эффективность вашей работы, оптимальное время отдыха для повышения эффективности как следствие и просто будет оставаться больше свободного времени.

На рынке мобильных приложений планирования своего времени присутствует множество приложений перегруженные функциями, которые большинство пользователей просто не будет использовать, но при этом эти функции будут как занимать лишнее место в памяти устройства, так и уменьшать автономность устройства.

Целью данного курсового проекта это разработка удобное для использования пользователем мобильное приложение. Для этого следует поставить основные задачи, которые нужно выполнить:

- Проанализировать рынок мобильных приложений планирования времени.
- Выделить главные функции данных приложений.
- Разработать дизайн приложения.
- Написать код приложения.
- Выбрать язык и среду разработки приложения.

В первую очередь надо проанализировать какие функции основные для всех приложений планирования времени, какие функции зачастую используются, а какие остаются без внимания.

Для того чтобы проанализировать часто используемые функции приложений планировщиков, я просмотрю и буду использовать приложения

из магазина приложений на моем мобильном устройстве, а также проведу опрос об основных и часто используемых функциях данных приложений.

Дизайн данных приложений должен быть таким, чтобы пользователь чувствовал себя комфортно при его использовании. Основные критерии, которые можно выделить:

– Интуитивно понятен.
– Минималистичен.
– Не перегружен.
– Приятен глазу.

Приложение будет разрабатываться на языке Kotlin в среде разработки Android Studio.

ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1 Описание предметной области

Тайм-менеджмент (планирование времени) – это методы и техники для управления своим временем. Они включают в себя самоорганизацию и управление собой. Тайм-менеджмент помогает человеку с большей эффективностью использовать свое время и экономить ресурсы на выполнение поставленных задач.

Приведу пример, если у вас накопилось большое количество дел, и вы не знаете за выполнение какой задачи приступить в первую очередь, то надо начать с расставления приоритетов. Можно воспользоваться матрицей Эйзенхауэра, которая поможет вам разобраться, какая задача является срочной и важной, а какая просто займет у вас время более важной задачи. Чем больше мы успеваем сделать, тем выше качество нашей работы и жизни.

Матрица Эйзенхауэра — это метод тайм-менеджмента, помогающий расставлять приоритеты: делать важное и не тратить время на ненужное. Он подойдет всем, кто хочет разобраться с личным и рабочим временем, научиться планировать расписание, которое не съезжает. Матрица состоит из четырех квадратов: срочно и важно, не срочно и важно, срочно и неважно, не срочно и неважно.

Также помимо матрицы Эйзенхауэра существует очень удобная система 4D, она позволяет распределить задачи на 4 категории:

– Do (сделать) – самые важные задачи, которые надо завершить в первую очередь.

– Delegate (делегировать) – передача задачи кому-либо, если вы не можете выполнить данную задачу.

– Delete (удалить) – задачи не имеющие важности и срочности.

– Delay (отложить) – задачи, которые не требуют срочного выполнения, но желательно поставить дату возвращения к данной задаче.

Но система 4D не подходит для долгосрочного планирования.

Основные принципы тайм-менеджмента, из которых состоят все методы управления временем:

– Приоритизация– определение насколько задача срочная, сложная и важная, и только после этого приступить к ее выполнению.

– Планирование – нужно проанализировать, когда приступить к выполнению данной задачи и сколько времени понадобиться чтобы ее выполнить.

– Структурирование – отслеживание выполнения и результатов задачи.

Главные плюсы там-менеджмента можно выделить сразу – более эффективное использование времени и улучшение производительности. Но не стоит забывать и о менее заметных плюсах:

– В результате получается четкий перечень задач.

– Более простая оценка ситуации помогает эффективно использовать время.

– Проще работать, когда ты видишь перед собой цель.

– Концентрация на ключевых задачах повышает эффективность.

– Хорошее планирование помогает человеку проще ориентироваться в рабочем процессе.

Приложения, помогающие нам в планировании времени, становятся всё популярнее. Результатом поиска в AppStore/PlayMarket по ключевому слову "ToDo" будет огромный список результатов. В большинстве случаев отличий между ними может быть много, но есть основные особенности.

У данных приложений имеется несколько основным групп:

– Метод GTD (Getting Things Done).

– Метод "Список покупок".

– Метод "текстового редактора".

Приложения GTD имеют основную идею, основанную на концепции того, что человек должен освободить разум от запоминания дел, которые ему

нужно совершить, полностью сосредоточившись на работе, таким образом повысив свою эффективность.

Такие приложения зачастую имеют такие функции:

– Список дел, которые нужно совершить.
– Напоминания.
– Выделение их важности каким-либо образом.
– Планировщик по дням.
– Голосовой ввод.
– Календарь.

Приложения “список покупок” очень просты и подкупают этим, зачастую являются простым списком с возможностью помечать завершенные задачи.

Приложения “текстового редактора” эти приложения могут формировать подходящую иерархию задач из обычных txt-файлов, что является очень удобным если вы привыкли хранить все именно в таком виде.

1.2 Сравнительный анализ

Необходимо чтобы интерфейс приложения не был “перегружен” и был интуитивно понятен для пользователя.

Возможность добавлять, удалять и редактировать список дел в приложении планировщике.

Приятный дизайн.

Не должно быть функций, которые зачастую будут игнорироваться пользователем и просто перегружать интерфейс своими иконками.

Решение “Tappsk”

Имеет приятный минималистичный дизайн с планировщиком по дням, возможность добавления задач голосом, напоминания и подзадачи, но имеет платную подписку.

Рисунок 1. Главное меню приложения “Tappsk”

Решение “Мои Дела”

Имеет минималистичный, но немного перегруженный дизайн списка дел, распределение задач на каждый день текущий недели, возможность создать аккаунт, уведомления, голосовой ввод.

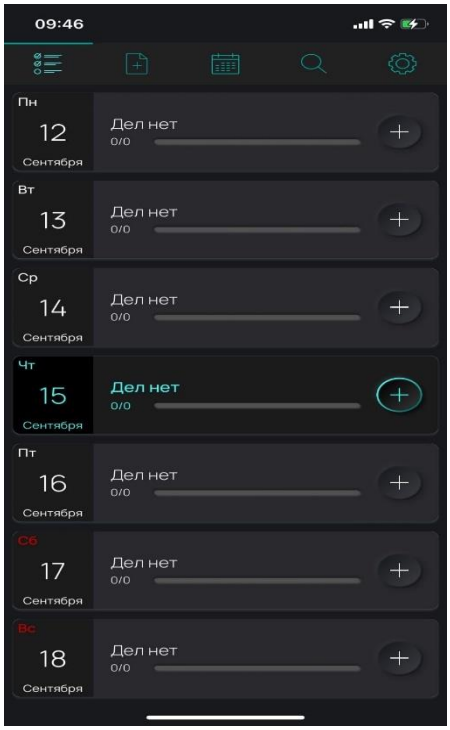

Рисунок 2. Главное меню приложения “Мои Дела”

Решение “Сделать”

Простой и удобный дизайн, сортировка задач, возможность отметить задачу 6 разными цветами.

Рисунок 3. Главное меню приложения “Сделать”

Таблица 1. Обзор предшествующих решений

Функции	Решение “Tappsk”	Решение “Мои Дела”	Решение “Сделать”	Мое решение
Приятный дизайн	+	-	+	+
Не перегруженность	+	-	+	+
Добавление голосом	+	+	-	-
Распределение на каждый день	+	+	-	-

На основе сравнительной таблицы можно сделать выводы о привлекательности моего решения. Любое решение имеет плюсы и минусы, поэтому при наличии всех вышеперечисленных функций можно говорить о его помощи в распределении дел для пользователя.

1.3 Постановка задачи

Входная информация

Таблица 2. Входная информация

Наименование	Идентификатор	Тип данных	Размер
Список пользователя	UserData	?	Динамический
Название задачи	TaskName	String	Не более 15 знаков
Описание задачи	TaskDescription	String	Не более 30 знаков
Важность задачи	TaskPriority	?	?

Выходные данные

Выходными данными приложения для планирования времени является список пользователя с задачами, которые он запланировал сделать в ближайшее время.

Требования к программе

Программа должна иметь следующие функции:

– Отображение списка задачи.
– Добавление задач в список.
– Редактирование задач в списке.
– Удаление задач из списка.

Данное однопользовательское приложения для планирования времени будет работать минимум на Android 7, и не иметь open source строения для защиты данных пользователей.

1.4 Характеристика инструментальных средств разработки

Android Studio — интегрированная среда разработки (IDE) для работы с платформой Android. В последней версии Android studio поддерживается Android 4.1 и выше.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, — официальное средство разработки Android приложений. Данная среда разработки доступна
для Windows, macOS и GNU/Linux. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в дополнение к Java и C++.

Возможности

Новые функции появляются с каждой новой версией Android Studio. На данный момент доступны следующие функции:

- Расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана.
- Сборка приложений, основанная на Gradle.
- Различные виды сборок и генерация нескольких .apk файлов.
- Рефакторинг кода
- Статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое.
- Встроенный ProGuard и утилита для подписывания приложений.
- Шаблоны основных макетов и компонентов Android.
- Поддержка разработки приложений для Android Wear и Android TV.
- Встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine.
- Android Studio 2.1 поддерживает Android N Preview SDK, а это значит, что разработчики смогут начать работу по созданию приложения для новой программной платформы.
- Новая версия Android Studio 2.1 способна работать с обновленным компилятором Jack, а также получила улучшенную поддержку Java 8 и усовершенствованную функцию Instant Run.

– Начиная с Platform-tools 23.1.0 для Linux исключительно 64-разрядная.

– В Android Studio 3.0 по стандарту включены инструменты языка Kotlin основанные на JetBrains IDE.

Kotlin — статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Также компилируется в JavaScript и в исполняемый код ряда платформ через инфраструктуру LLVM. Язык назван в честь острова Котлин в Финском заливе, на котором расположен город Кронштадт.

Авторы ставили целью создать язык более лаконичный и типобезопасный, чем Java, и более простой, чем Scala. Следствием упрощения по сравнению со Scala стали также более быстрая компиляция и лучшая поддержка языка в IDE. Язык полностью совместим с Java, что позволяет Java-разработчикам постепенно перейти к его использованию; в частности, язык также встраивается Android, что позволяет для существующего Android-приложения внедрять новые функции на Kotlin без переписывания приложения целиком.

Пользовательский интерфейс

Пользовательский интерфейс будет реализован с помощью встроенных функций и тегов **Android Studio.**

Система управления данными будет реализована с помощью Room Storage и SQLite.

Таблица состоит из 3 полей

– Id : Int

– Title : String

– Priority: String

Для построения диаграмм мной использовался онлайн ресурс Draw.io. Diagrams.net (ранее draw.io) - это бесплатное кроссплатформенное программное обеспечение для построения графиков с открытым исходным

кодом, разработанное на HTML5 и JavaScript. Его интерфейс можно использовать для создания диаграмм, таких как блок-схемы, каркасы, диаграммы UML, организационные диаграммы и сетевые диаграммы.

Diagrams.net доступно как онлайн в качестве кроссбраузерного веб-приложения, так и в качестве автономного настольного приложения для Linux, macOS и Windows.

ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1 Анализ требований и определение спецификаций

На входе в программу подаются данные авторизованного пользователя, а также названия и описание задач пользователя. На выходе получается готовый список задач.

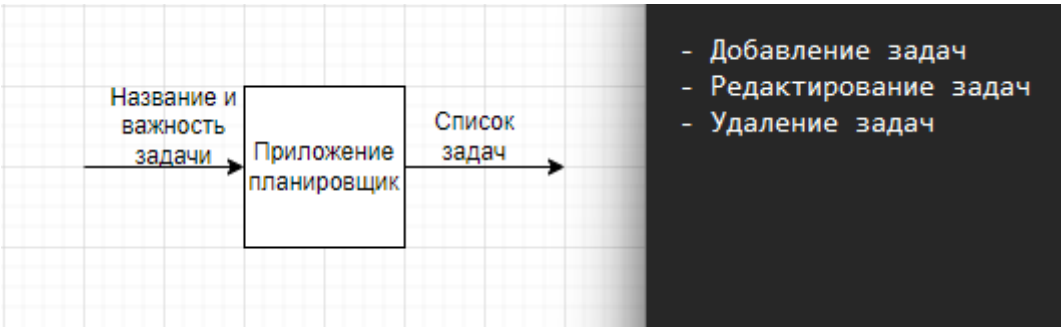

Рисунок 4. Краткая функциональная диаграмма

Мобильное приложение для планирования времени состоит из функций. В функцию ввода задачи входят данные о названии и описании задачи. Далее данные задачи переходят в функцию добавлении задачи. Удаление задачи удаляет данные конкретной задачи из списка. Функция отображения списка задач отображает список.

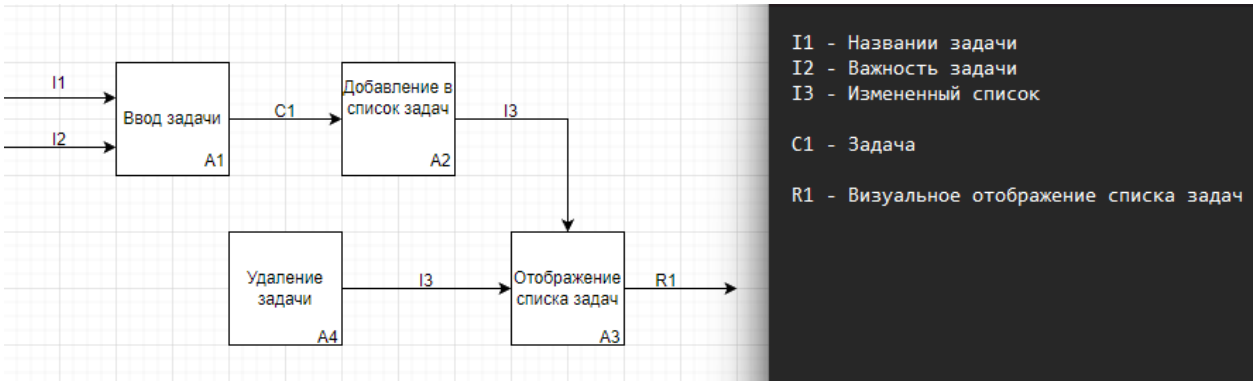

Рисунок 5. Функциональная диаграмма

В мобильном приложении планирования времени пользователь передает программе название и приоритет важности задачи, на выходе получая готовый план.

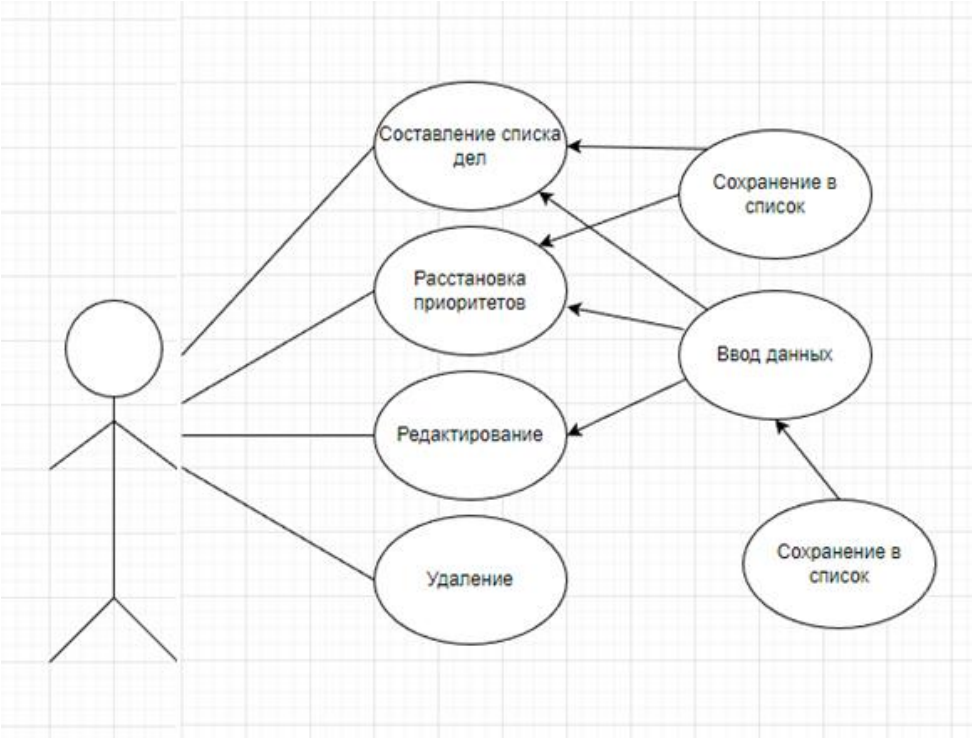

Рисунок 6. Диаграмма вариантов использования

При в ходе в мобильное приложения для планирования времени пользователь увидит меню с списком задач, которые были созданы ранее или их отсутствие в ином случае. Пользователь может добавить задачу, редактировать и удалить ее.

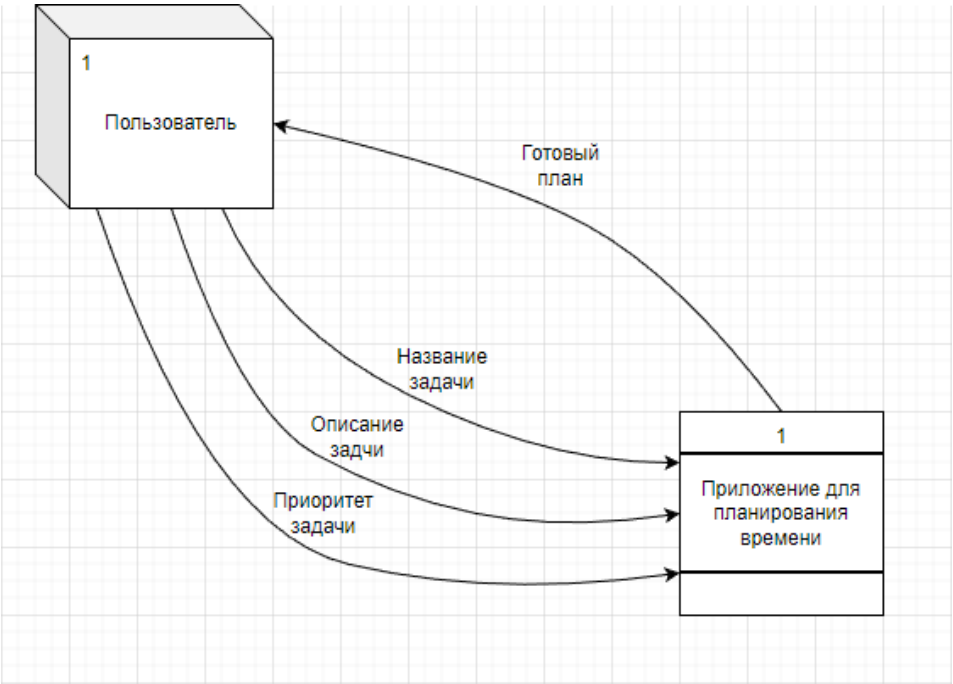

Рисунок 7. Краткая диаграмма потока данных

Мобильное приложение планирования времени апеллирует данными, которые переходят от одной функции к другой. Пользователь авторизуется/регистрируется, и после этого подгружается список данного пользователя и возможность взаимодействия с ним. Все действия пользователя, направленные на изменения списка пользователя, обрабатываются определенной функцией, после этого они изменяют список задач и снова отправляются в функцию отображения списка задач.

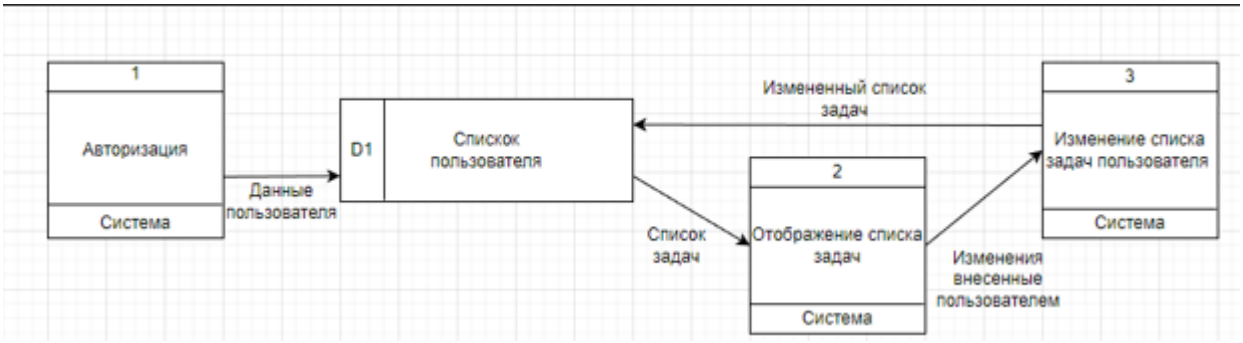

Рисунок 8. Диаграмма потока данных

В мобильном приложении планирования времени пользователь может составить список дел и расставить приоритет каждой задачи. Также пользователь может редактировать задачи: изменять название, описание и приоритет важности. Также пользователю доступна возможность удалять выполненные задачи.

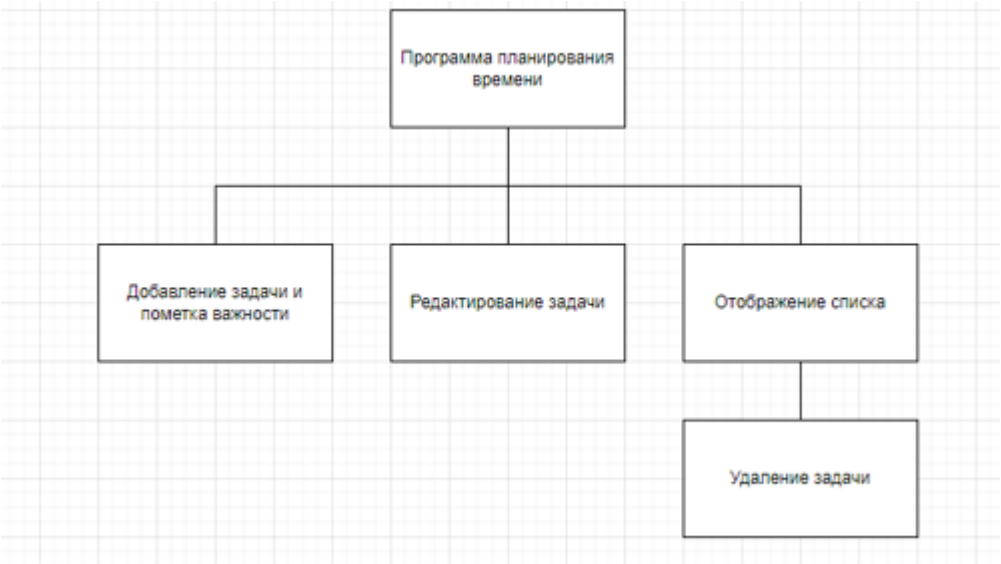

Рисунок 9. Структурная схема

В мобильном приложении планирования времени, пользователь при запуске приложения авторизуется/регистрируется в нем, после этого попадает в главное меню, где выводиться список задач. Пользователь может добавлять, редактировать и удалять задачи.

2.2 Проектирование мобильного приложения

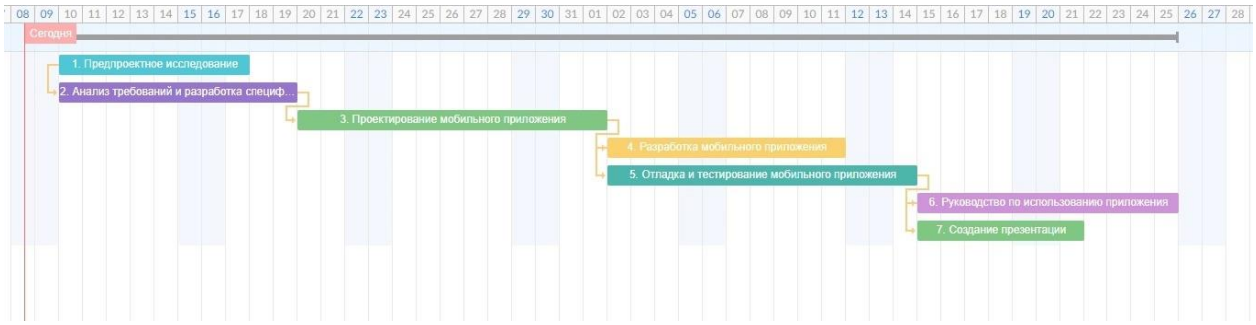

Рисунок 10. Диаграмма Ганта

На рисунке 7 изображена диаграмма Ганта.

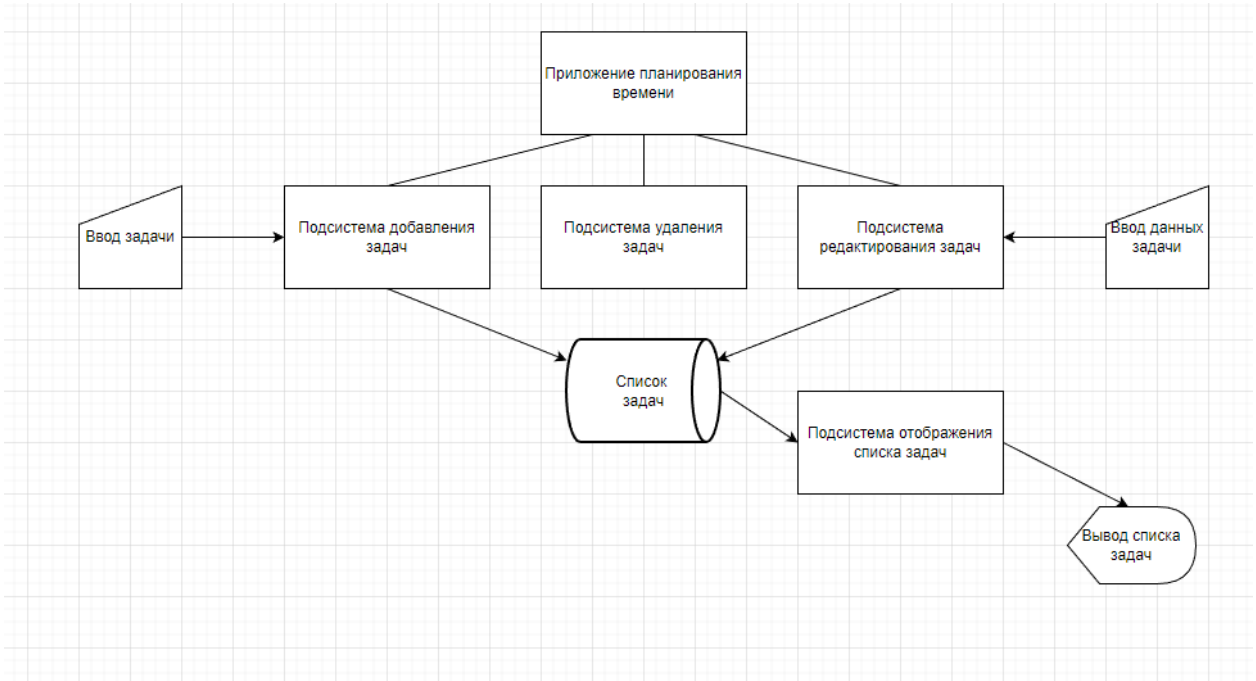

Рисунок 11. Функциональная схема

В мобильном приложении планирования времени пользователь передает программе название, описание и приоритет важности задачи, на выходе получая готовый план.

2.3 Разработка мобильного приложения

На рисунках 12-16 представлены скриншоты интерфейса приложения.

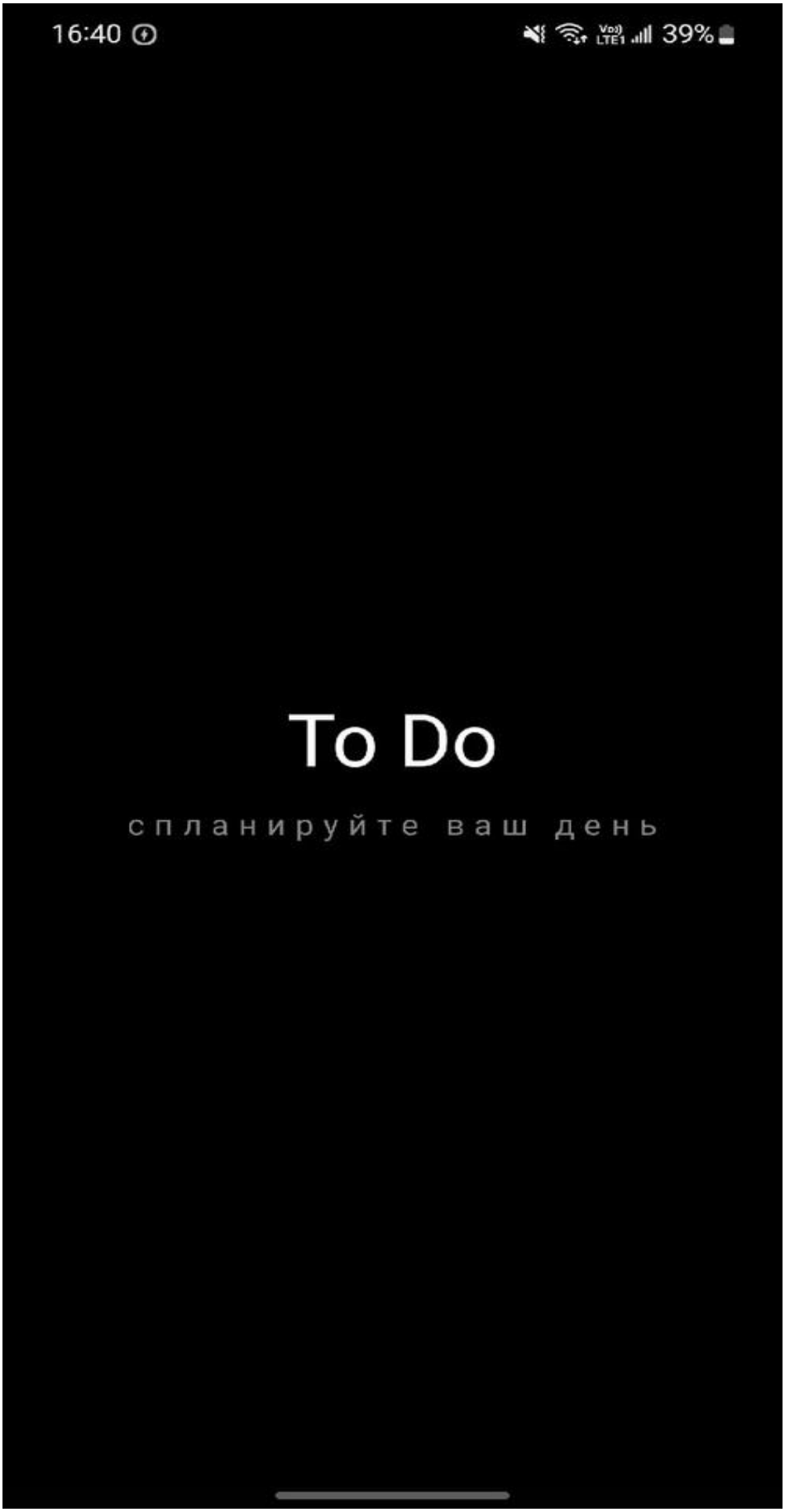

Рисунок 12. Экран приветствия

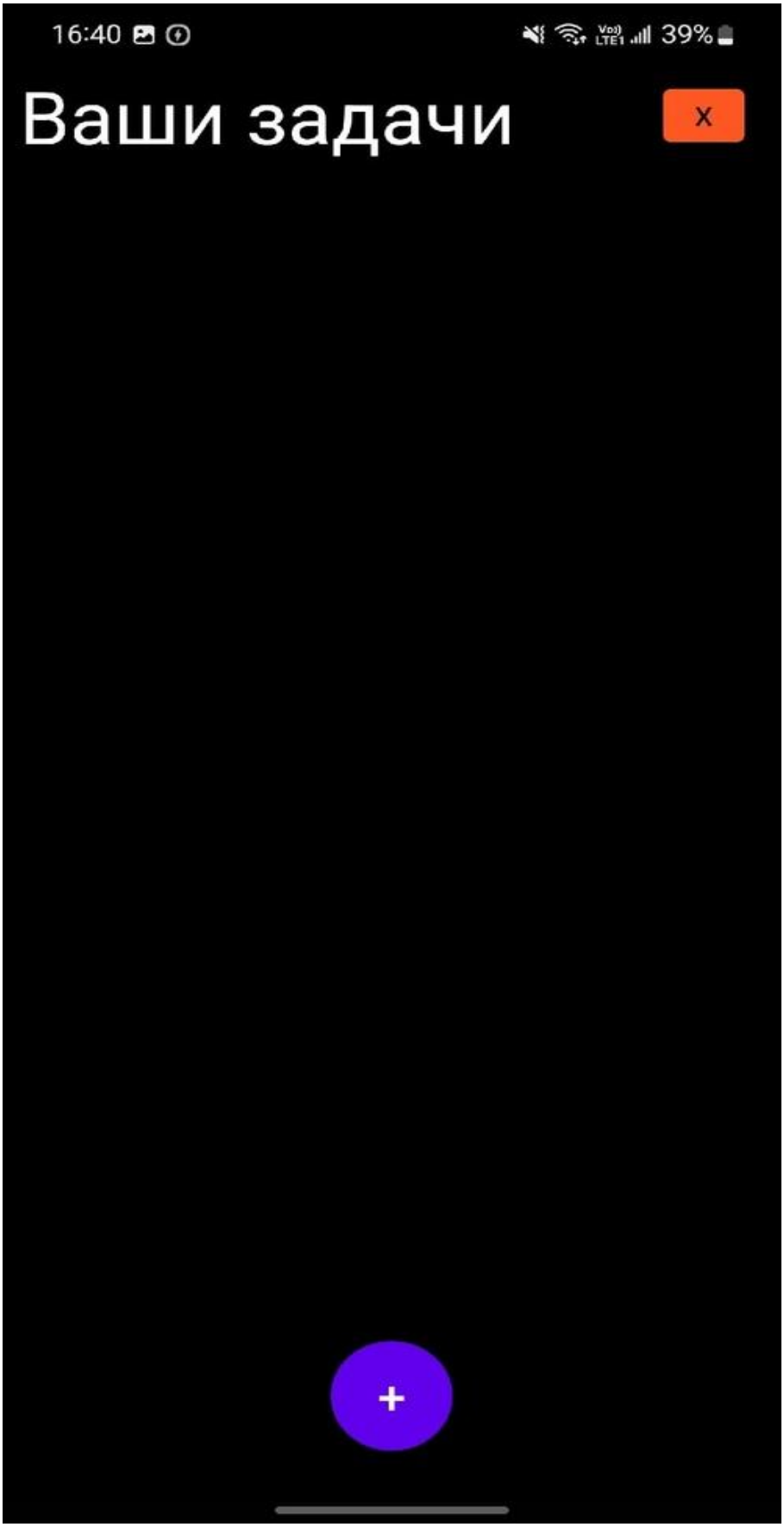

Рисунок 13. Главный экран приложения

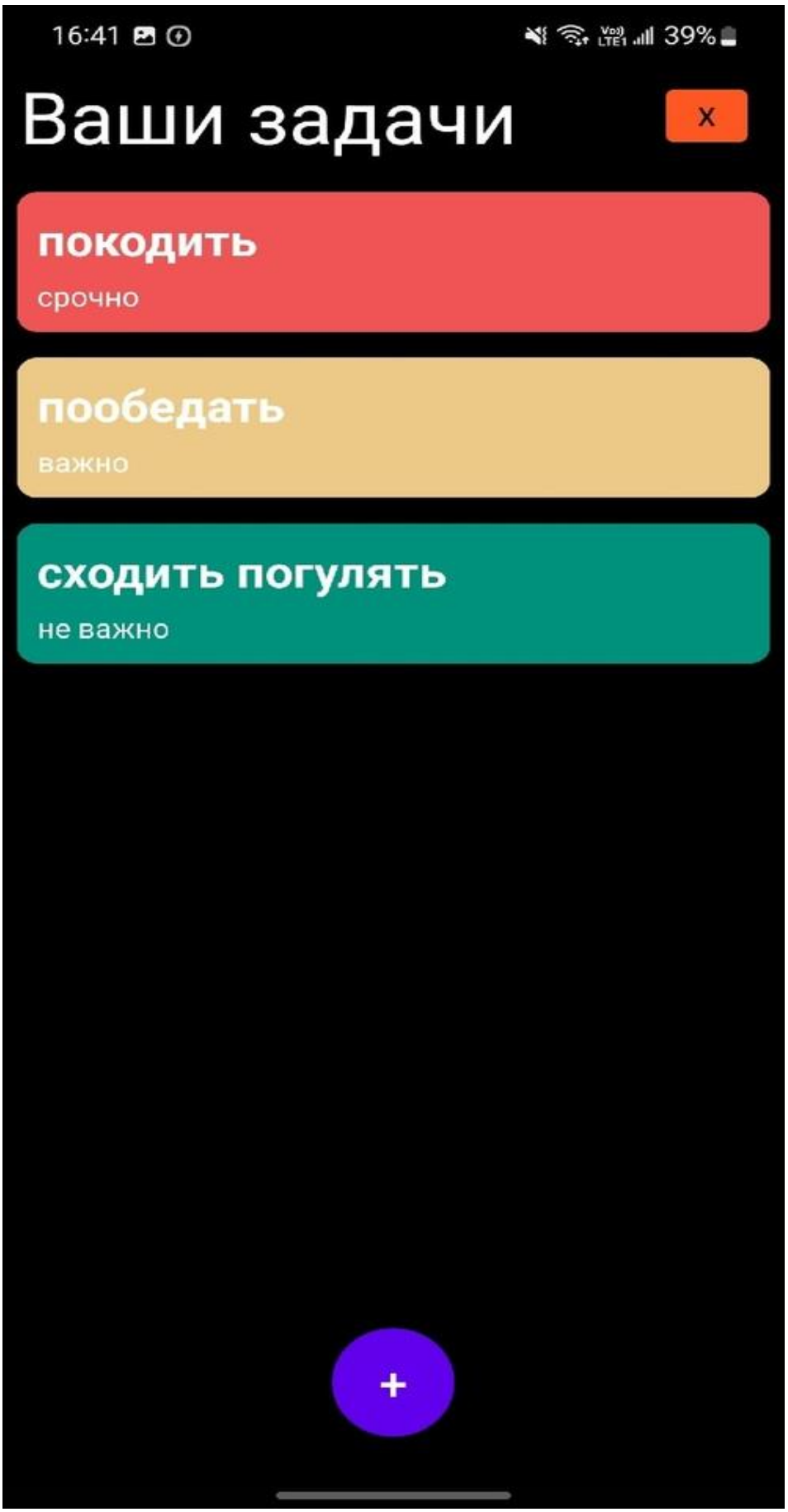

Рисунок 14. Главный экран с заполненным списком задач

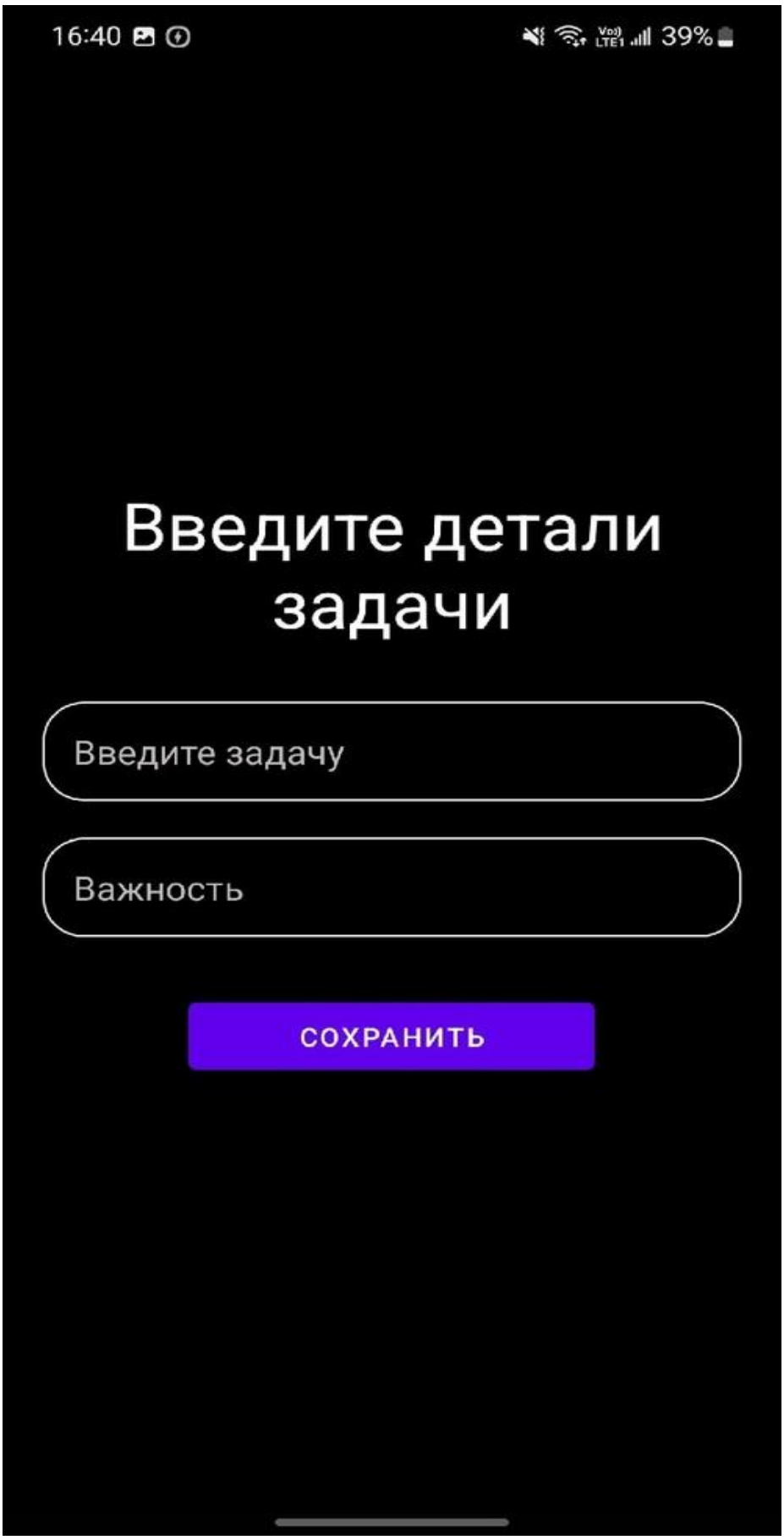

Рисунок 15. Экран добавления задачи

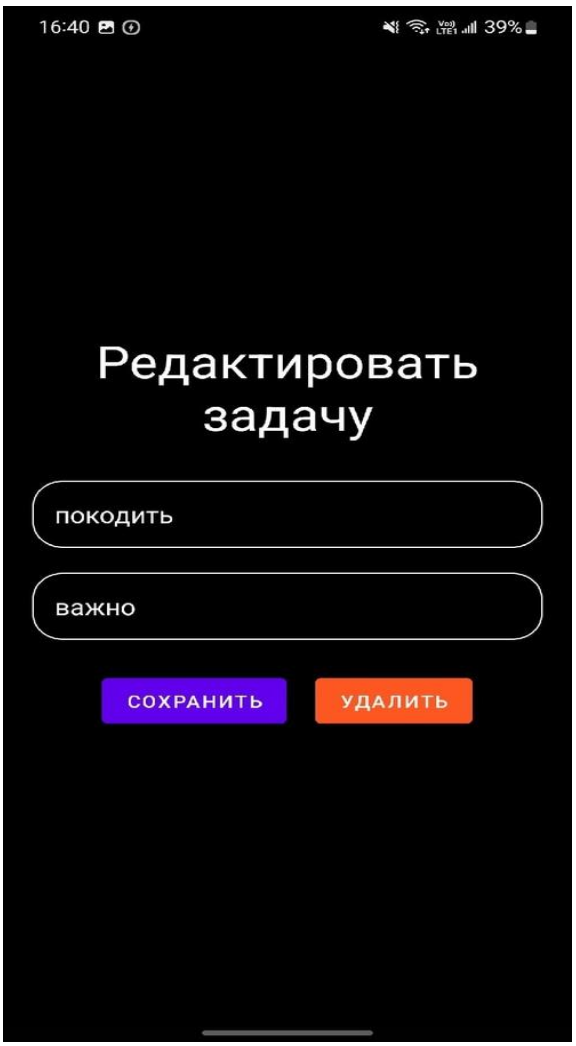

Рисунок 16. Экран редактирования задачи

Создание приложения планировщика времени начинается с главного экрана (см. рисунок 13), на котором будет отображаться список задач и взаимодействие с ним.

На форме представлены 2 кнопки TextView и RecycleView. Каждый элемент очень важен.

Круглая фиолетовая кнопка внизу экрана создана для добавления задчи в список. При нажатии она перенаправляет нас на экран добавления задачи (см. рисунок 15), после чего данные задачи нужно ввести в поля и нажать кнопку сохранить.

Прямоугольная красная кнопка в верхней правой части экрана создана для удаления всех имеющихся задач из списка.

TextView создан для более лаконичного дизайна и находится в верхнем левом углу.

RecycleView создан для отображения списка задач и находится в центральной части приложения.

Код на xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:background="@color/black"
tools:context=".MainActivity">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="8dp"
android:gravity="center"
android:orientation="horizontal">
<TextView
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:fontFamily="sans-serif"
android:text="Ваши задачи"
android:textColor="@color/white"
android:textSize="40sp" />
<Button
android:id="@+id/deleteAll"
android:layout_width="40dp"
```

```
android:layout_height="40dp"
android:layout_marginEnd="10dp"
android:backgroundTint="#FF5722"
android:text="X"
android:textAllCaps="false"
android:textColor="@color/black"/>
</LinearLayout>
<FrameLayout
android:layout_width="match_parent"
android:layout_height="match_parent">
<androidx.recyclerview.widget.RecyclerView
android:id="@+id/recycler_view"
tools:listitem="@layout/view"
android:layout_width="match_parent"
android:layout_height="match_parent" />
<Button
android:id="@+id/add"
android:layout_width="60dp"
android:layout_height="60dp"
android:layout_gravity="center|bottom"
android:layout_marginBottom="25dp"
android:background="@drawable/circular_button"
android:text="+"
android:textColor="@color/white"
android:textSize="25sp" />
</FrameLayout>
</LinearLayout>
```

Код на Kotlin:

```
class MainActivity : AppCompatActivity() {
private lateinit var database: myDatabase //подключаем базу данных
```

```
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
database = Room.databaseBuilder(
applicationContext, myDatabase::class.java, "To_Do").build()
add.setOnClickListener {
val intent = Intent(this, CreateCard::class.java)
startActivity(intent)} // переходим на страницу создания задачи
deleteAll.setOnClickListener {
DataObject.deleteAll()
GlobalScope.launch {
database.dao().deleteAll()}
setRecycler()} // удалаем все задачи из списка
setRecycler()}
fun setRecycler() {
recycler_view.adapter = Adapter(DataObject.getAllData())
recycler_view.layoutManager = LinearLayoutManager(this)} // Объявлем
RecyclerView  и добавлем в него данные}
```

Следующим шагом будет создание объекта списка, который будет отображать задачу. Он будет состоять из двух TextView.

Оба TextView отображают данные о задаче.

Код на xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="7dp"
xmlns:app="http://schemas.android.com/apk/res-auto">
<androidx.cardview.widget.CardView
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:elevation="3dp"
android:backgroundTint="@color/black"
app:cardCornerRadius="10dp">
<LinearLayout
android:id="@+id/mylayout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="10dp"
android:orientation="vertical">
<TextView
android:id="@+id/title"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="ваша задча будет тут"
android:textStyle="bold"
android:textSize="24sp"
android:textColor="@color/white"
android:fontFamily="sans-serif"/>
<TextView
android:id="@+id/priority"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="приоритет"
android:textSize="15sp"
android:layout_marginTop="5dp"
android:textColor="@color/white"
android:fontFamily="sans-serif"/>
</LinearLayout>
</androidx.cardview.widget.CardView>
```

</LinearLayout>

Следующим шагом было создание экрана добавления задачи (см. рисунок 15). На нем находятся TextView, 2 EditText и кнопка.

Оба EditText выполняют задачу передачи 2 значений, такие как: title и priority.

TextView показывает пользователю что он находится на экране добавления задачи.

Кнопка отвечает за добавление задачи в список задач.

Код на xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:background="@color/black"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center"
android:padding="20dp"
tools:context=".CreateCard">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Введите детали задачи"
android:textColor="@color/white"
android:textSize="36sp"
android:gravity="center"
android:fontFamily="sans-serif"/>
<EditText
android:id="@+id/create_title"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Введите задачу"
android:layout_marginTop="30dp"
android:padding="15dp"
android:background="@drawable/custom_edittetext"
android:textColorHint="#BFBBBB"
android:textColor="@color/white"/>
<EditText
android:id="@+id/create_priority"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
android:hint="Важность"
android:padding="15dp"
android:background="@drawable/custom_edittetext"
android:textColorHint="#BFBBBB"
android:textColor="@color/white"/>
<Button
android:id="@+id/save_button"
android:layout_width="200dp"
android:layout_height="wrap_content"
android:layout_marginTop="30dp"
android:text="Сохранить"/>
</LinearLayout>
```

Код на Kotlin:

```
class CreateCard : AppCompatActivity() {
private lateinit var database: myDatabase // подключаем базу данных
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
```

```
setContentView(R.layout.activity_create_card)
database = Room.databaseBuilder(
applicationContext, myDatabase::class.java, "To_Do").build()
save_button.setOnClickListener {
if (create_title.text.toString().trim { it <= ' ' }.isNotEmpty()
&& create_priority.text.toString().trim { it <= ' ' }.isNotEmpty() // проверка на ввод значений) {
// объявляем переменные для хранения данных задачи
var title = create_title.getText().toString()
var priority = create_priority.getText().toString()
DataObject.setData(title, priority)
GlobalScope.launch {
database.dao().insertTask(Entity(0, title, priority))} // добавляем данные в базу данных
val intent = Intent (this, MainActivity::class.java)
startActivity(intent) // открываем главный экран}} // функция добавления задачи в список}}
```

Следующим шагом было создание экрана редактирования задачи (см. рисунок 16), которое вызывается при нажатии на задачу. На нем находятся TextView, 2 EditText и 2 кнопки.

TextView отвечает за информирование пользователя о том что он находится на экране редактирования задачи.

Оба EditText выполняют задачу передачи 2 значений, такие как: title и priority.

Кнопка “Сохранить” сохраняет все изменения, проведенные с задачей.

Кнопка “Удалить” удаляет выбранную задачу.

Код xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:background="@color/black"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center"
android:padding="20dp"
tools:context=".UpdateCard">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Редактировать задачу"
android:textColor="@color/white"
android:textSize="36sp"
android:gravity="center"
android:fontFamily="sans-serif"/>
<EditText
android:id="@+id/create_title"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Введите задачу"
android:layout_marginTop="30dp"
android:padding="15dp"
android:background="@drawable/custom_edittetext"
android:textColorHint="#BFBBBB"
android:textColor="@color/white"/>
<EditText
android:id="@+id/create_priority"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:layout_marginTop="20dp"
android:hint="Важность"
android:padding="15dp"
android:background="@drawable/custom_edittetext"
android:textColorHint="#BFBBBB"
android:textColor="@color/white"/>
<LinearLayout
android:layout_width="250dp"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:layout_marginTop="25dp">
<Button
android:id="@+id/update_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Сохранить"/>
<View
android:layout_width="0dp"
android:layout_height="0dp"
android:layout_weight="1"/>
<Button
android:id="@+id/delete_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:backgroundTint="#FF5722"
android:text="Удалить"/>
</LinearLayout>
</LinearLayout>
```

Код на Kotlin:

```
class UpdateCard : AppCompatActivity() {
```

```
private lateinit var database: myDatabase // подключаем базу данных
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_update_card)
database = Room.databaseBuilder(
applicationContext, myDatabase::class.java, "To_Do").build()
val pos = intent.getIntExtra("id", -1)
if (pos != -1) {
val title = DataObject.getData(pos).title
val priority = DataObject.getData(pos).priority
create_title.setText(title)
create_priority.setText(priority)
delete_button.setOnClickListener {
DataObject.deleteData(pos)
GlobalScope.launch {
database.dao().deleteTask(
Entity(pos + 1,create_title.text.toString(),create_priority.text.toString()))}
myIntent()} // функция удаления
update_button.setOnClickListener {
DataObject.updateData(pos,create_title.text.toString(),create_priority.text.toString)
) // создаем объект обновленной задачи
GlobalScope.launch {
database.dao().updateTask(
Entity(pos + 1, create_title.text.toString(),create_priority.text.toString()))} //
передаем обновленные данные в базу данных
myIntent()}}}
fun myIntent() {
val intent = Intent(this, MainActivity::class.java)
startActivity(intent)} // функция вызова главного экарана}
```

И последним шагом, связанным с интерфейсом приложения было создание приветственного экрана (см. рисунок 13). Он состоит из 2 TextView.

Оба TextView выводят приветственную информацию. Первый выводит название приложения “To Do”, а второй слоган “спланируйте ваш день”

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:background="@color/black"
android:orientation="vertical"
android:gravity="center_vertical"
android:layout_height="match_parent"
tools:context=".SplashScreen">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="To Do"
android:layout_gravity="center_horizontal"
android:textColor="@color/white"
android:textSize="40sp"
android:fontFamily="sans-serif"/>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="спланируйте ваш день"
android:layout_gravity="center_horizontal"
android:textColor="#8A8686"
android:layout_marginTop="10dp"
android:textSize="16sp"
```

```
android:letterSpacing="0.3"
android:fontFamily="sans-serif"/>
</LinearLayout>
```

Для того чтобы RecycleView работал корректно был создан класс Adapter.

Код на Kotlin:

```
class Adapter(var data: List<CardInfo>) :
RecyclerView.Adapter<Adapter.viewHolder>() {
class viewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
var title = itemView.title
var priority = itemView.priority
var layout = itemView.mylayout} // создаем объект задачи
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
viewHolder {
var itemView = LayoutInflater.from(parent.context).inflate(R.layout.view, parent,
false)
return viewHolder(itemView)}
override fun onBindViewHolder(holder: viewHolder, position: Int) {
when (data[position].priority.toLowerCase()) {
"срочно" -> holder.layout.setBackgroundColor(Color.parseColor("#F05454"))
"важно" -> holder.layout.setBackgroundColor(Color.parseColor("#EDC988"))
else -> holder.layout.setBackgroundColor(Color.parseColor("#00917C"))} //
создаем привязку цвета объекта задачи к важности задачи
// передаем данные
holder.title.text = data[position].title
holder.priority.text = data[position].priority
holder.itemView.setOnClickListener{
val intent=Intent(holder.itemView.context,UpdateCard::class.java)
intent.putExtra("id",position)
```

```
holder.itemView.context.startActivity(intent)} // переходим на экран
редактирования задачи при нажатии на нее}
override fun getItemCount(): Int {
return data.size} // получаем данные о количестве задач и передаем его}
```

Также создадим класс для объекта задачи.

Код на Kotlin:

```
data class CardInfo(
var title:String,
var priority:String)
```

Создадим класс базы данных.

Код на Kotlin:

```
@Database(entities = [Entity::class],version=1)
abstract class myDatabase : RoomDatabase() {
abstract fun dao():DAO}
```

Создадим объект данных с функциями для взаимодействия с ней.

Код на Kotlin:

```
object DataObject {
var listdata = mutableListOf<CardInfo>()
fun setData(title: String, priority: String) {
listdata.add(CardInfo(title, priority))} // передача задачи в список задач
fun getAllData(): List<CardInfo> {
return listdata} // получение списка задач
fun deleteAll(){
listdata.clear()} // удаление всех задач из списка
fun getData(pos:Int): CardInfo {
return listdata[pos]} // получение данных конкретной задачи
fun deleteData(pos:Int){
listdata.removeAt(pos)} // удаление конкретной задачи
fun updateData(pos:Int,title:String,priority:String){
listdata[pos].title=title
```

```
listdata[pos].priority=priority} // редактирование конкретной задачи}
```

Создадим класс для таблицы базы данных.

Код на Kotlin:

```
@Entity(tableName = "To_Do")
data class Entity(
@PrimaryKey(autoGenerate = true)
var id:Int,
var title:String,
var priority:String)
```

Создадим интерфейс для взаимодействия с базой данных.

Код на Kotlin:

```
@Dao
interface DAO {
@Insert
suspend fun insertTask(entity: Entity) // добавление задачи
@Update
suspend fun updateTask(entity: Entity) // редактирование задачи
@Delete
suspend fun deleteTask(entity: Entity) // удаление задачи
@Query("Delete from to_do")
suspend fun deleteAll() // удаление всех задач
@Query("Select * from to_do")
suspend fun getTasks():List<CardInfo> // получение задач}
```

2.4 Отладка и тестирование мобильного приложения

Для проверки работы реализованного приложения необходимо провести отладку и тестирование.

№	Входные данные	Вводимое значение	Ожидаемая реакция программы	Фактическая реакция	Ошибка выявлена
1	Данные задачи	«Погулять, не важно»	Задача окрасится зеленым	Рисунок 17	нет
2	Данные задачи	«Покодить, срочно»	Задача окрасится красным	Рисунок 18	нет
3	Данные задачи	«Помыть посуду, важно»	Задача окрасится оранжевым	Рисунок 19	нет
4	Изменение данных задачи	Изменить приоритет задачи «Помыть посуду, важно» на «не важно»	Цвет задачи изменится на зеленый	Рисунок 20	нет
5	Удаление задачи	Удаление задачи из списка	Задача будет удалена из списка задач	Рисунок 21	нет

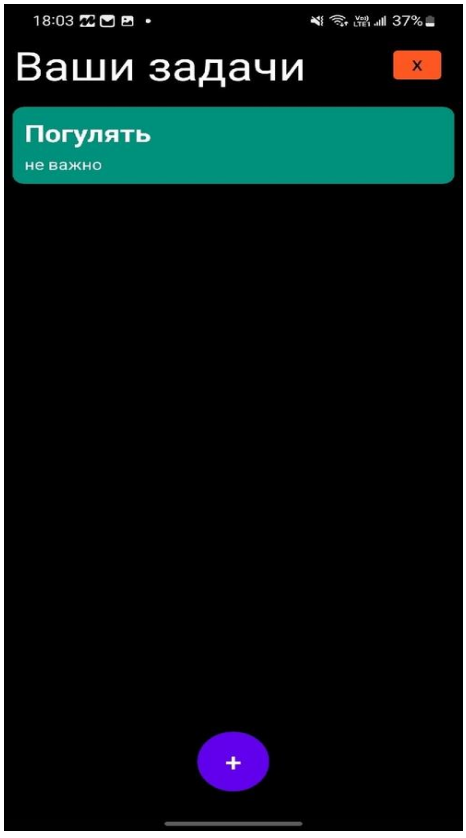

Рисунок 17. Фактическая реакция на правильный ввод данных

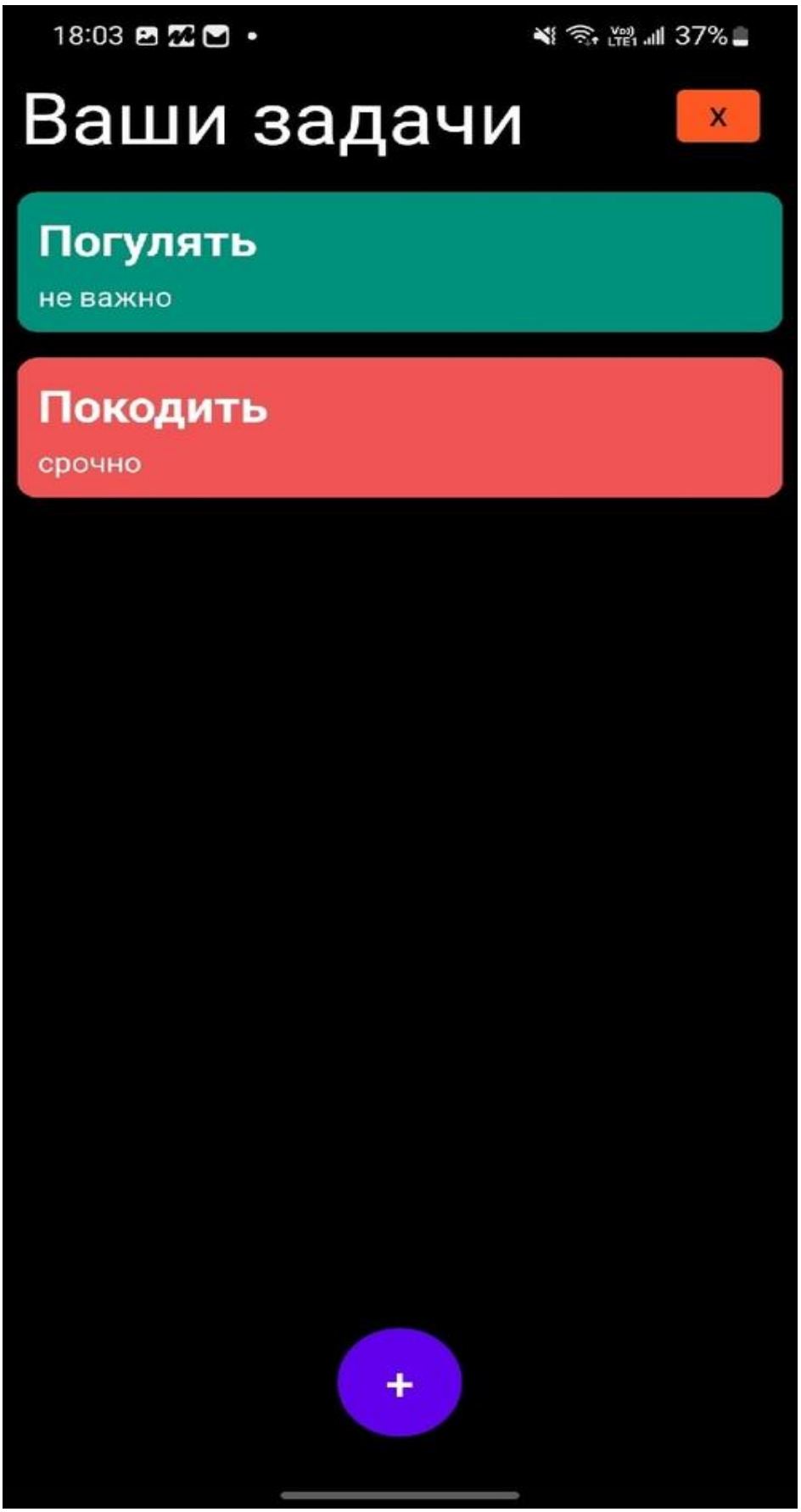

Рисунок 18. Фактическая реакция на правильный ввод данных

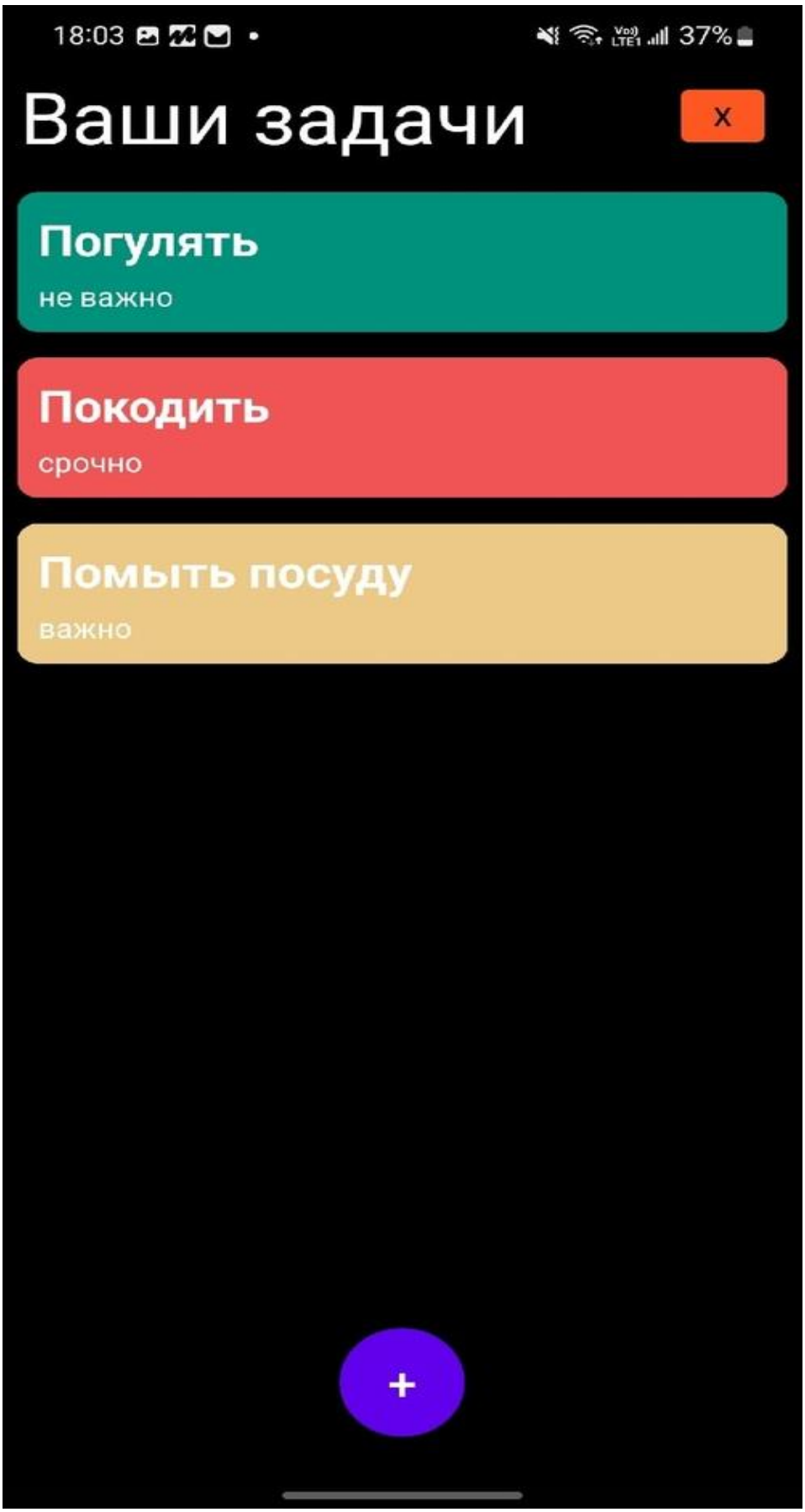

Рисунок 19. Фактическая реакция на правильный ввод данных

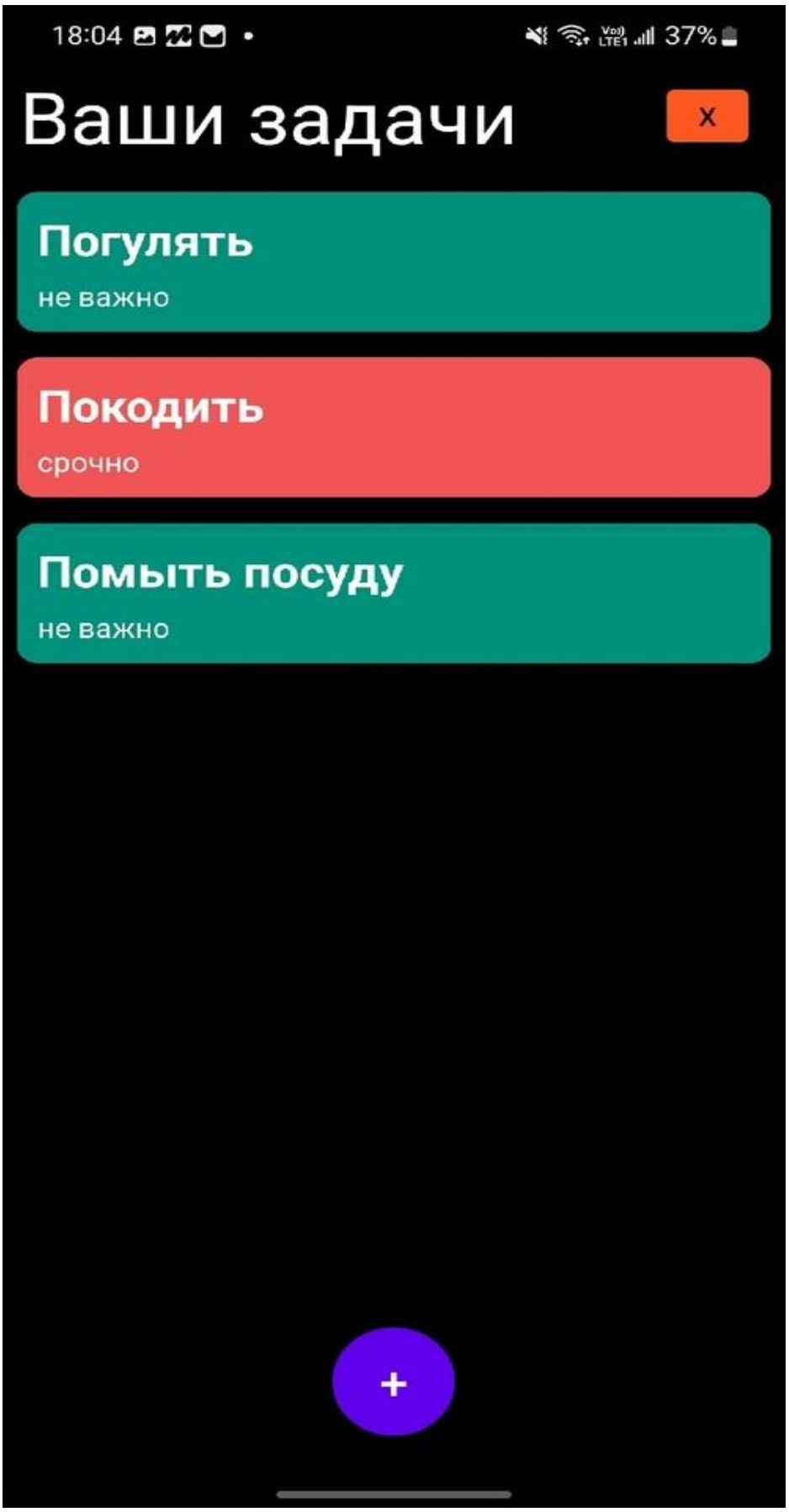

Рисунок 20. Фактическая реакция на правильный ввод данных

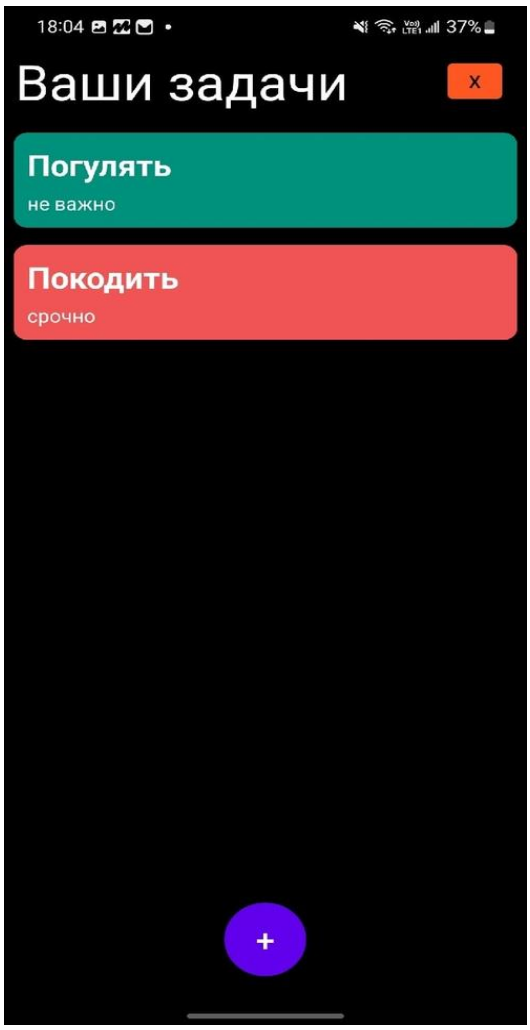

Рисунок 21. Фактическая реакция на правильный ввод данных

2.5 Руководство по использованию мобильного приложения

Руководство пользователя

Для взаимодействия с приложением необходимо соблюсти следующие эксплуатационные требования:

- Android 7 и выше
- 50 мб свободной памяти

Есть ряд правил работы с приложением:

1. При создании или редактировании задачи оба поля должны быть заполнены;

2. Для выставления важности следует использовать определенные слова: «важно», «срочно» и для самых маловажных можно использовать любые слова.

Руководство программиста

Назначение программы заключается в упрощении планирования свои дел.

Системные требования:

– Android 7 и выше
– 50 мб свободной памяти

Для возможности работы с приложением требуется мобильный телефон на базе Android 7 и выше

Для модернизации программы необходимо наличие инструментальной среды разработки Android Studio.

ЗАКЛЮЧЕНИЕ

Данные проект был направлен на разработку приложения для планирования времени.

В ходу выполнения курсовой работы были выполнены все поставленные задачи, а именно:

– Проанализировать рынок мобильных приложений планирования времени.
– Выделить главные функции данных приложений.
– Разработать дизайн приложения.
– Написать код приложения.
– Выбрать язык и среду разработки приложения.

В разработанном приложении планирования времени предусмотрены следующие функции:

– Отображение списка задач
– Добавление задачи
– Редактирование задачи
– Удаление задачи
– Выделение приоритетности задачи с помощью разных цветов

Благодаря всем вышеперечисленных задачам, основная цель курсовой работы была достигнута. Было разработано мобильное приложение планирования времени.

Говоря о достоинствах мобильного приложения планирования времени, можно отметить приятный и интуитивно понятный дизайн и очень малый вес приложения.

Из недостатков можно выделить недостаток функций, который может заметить пользователь, нуждающийся в более расширенном функционале.

Одним из основных путей развития является добавление дополнительных функций в приложение.

Подводя итог, можно сказать, что с помощью мобильного приложения планирования времени пользователь сможет распланировать свои задачи по их важности и понять какую задачи выполнять в первую очередь.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Законодательные и нормативные акты:

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.

2. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2010. – 92 с.

3. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.

4. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.

5. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.

6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.

7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

Интернет-документы

8. Введение в язык Kotlin. – [Электронный ресурс] – URL: https://metanit.com/kotlin/tutorial/1.1.php

9. Введение в Kotlin JVM. [Электронный ресурс] – URL: https://stepik.org/lesson/66286/step/1?unit=43162

10. Kotlin в действии / Д. Жемеров, С. Исакова. – Москва: Издательство ДМК Пресс, 2018. Текст – электронный URL: https://yurecnt.ru/files/books/s1p4kl2p10b9y2xr1jlxtji5xg2yl6.pdf

11. Meet Android Studio. [Электронный ресурс] – URL:

https://developer.android.com/studio/intro

12. Build Your First Android App in Kotlin. [Электронный ресурс] – URL: https://developer.android.com/codelabs/build-your-first-android-app-kotlin#0

13. Современная Android разработка на Kotlin. [Электронный ресурс] – URL: https://habr.com/ru/post/341602/

14. Android Studio для начинающих (Kotlin). [Электронный ресурс] – URL: https://neco-desarrollo.es/android-studio-для-начинающих-kotlin