МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка мобильного приложения для учета тайм-менеджмента «TimeLab»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 22-1  Ковалев Никита Анатольевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2025

Аннотация

Курсовой проект посвящён разработке мобильного приложения «TimeLab» для эффективного управления задачами и личным временем пользователей. Актуальность проекта обусловлена возросшей потребностью в инструментах тайм-менеджмента в условиях динамичного образа жизни и большого количества рабочих и учебных задач.

В рамках проекта проведён анализ существующих аналогов, таких как TimeTune и Sectograph, выявлены их сильные и слабые стороны, на основании чего были сформулированы требования к функционалу и интерфейсу нового приложения.

Разработка велась на языке Kotlin с использованием архитектурного шаблона MVVM. Для хранения пользовательских данных применялся механизм SharedPreferences, обеспечивающий надёжное локальное сохранение задач.

Приложение «TimeLab» предоставляет пользователю возможности по созданию, редактированию, удалению и сортировке задач, установке напоминаний и фильтрации выполненных задач. Все функции были протестированы, что подтвердило стабильную и корректную работу программы.

Результаты курсового проекта могут быть использованы для дальнейшего расширения функциональности приложения, интеграции с облачными сервисами и улучшения аналитических возможностей для пользователей.

Ключевые слова: мобильное приложение, тайм-менеджмент, планирование задач, Android, Kotlin, MVVM, SharedPreferences.

Объем работы: 48 страниц.

Приложение: «TimeLab» (APK-файл).

Среда Разработки: Android Studio.

GitHub: https://github.com/m0telblvck/TimeLab

Содержание

[Введение 4](#_Toc196597592)

[1 Анализ предметной области 6](#_Toc196597593)

[1.1 Аналоги разрабатываемого приложения 7](#_Toc196597594)

[1.2 Техническое задание 10](#_Toc196597595)

[1.3 Описание структуры приложения 12](#_Toc196597596)

[2 Описание разработки приложения 14](#_Toc196597597)

[2.1 Обоснование средств разработки 14](#_Toc196597598)

[2.2 Разработка интерфейса 16](#_Toc196597599)

[2.3 Разработка логики работы программы. Схема взаимодействия компонентов проекта 23](#_Toc196597600)

[2.3.1 Логика работы приложения 23](#_Toc196597601)

[2.3.2 Схема взаимодействия компонентов 23](#_Toc196597602)

[2.4 Описание переменных, компонентов, классов и подпрограмм 24](#_Toc196597603)

[3 Тестирование и установка программы 28](#_Toc196597604)

[Заключение 33](#_Toc196597605)

[Список использованной литературы 34](#_Toc196597606)

Приложение А. [Листинг программы 35](#_Toc196597607)

Приложение Б. [Руководство пользователя 43](#_Toc196597608)

Введение

Современный мир предъявляет высокие требования к управлению личным и рабочим временем. В условиях постоянно растущего потока информации, увеличивающегося числа задач и необходимости эффективного планирования, навык тайм-менеджмента становится незаменимым для достижения поставленных целей. Люди стремятся оптимизировать свою деятельность, используя различные методики и инструменты, которые помогают структурировать повседневные задачи и рационально распределять ресурсы.

Одним из наиболее удобных решений в области управления временем являются мобильные приложения, которые позволяют пользователям систематизировать свою работу, отслеживать выполнение задач и получать аналитику по использованию времени. В данной курсовой работе рассматривается разработка мобильного приложения «TimeLab», предназначенного для организации и оптимизации процессов планирования.

Приложение «TimeLab» создано для повышения продуктивности пользователей за счёт эффективного управления задачами и временными затратами. Оно предоставляет удобные инструменты для добавления задач, установки приоритетов, мониторинга выполнения и анализа продуктивности.

Основная цель проекта — создание интуитивно понятного и функционального программного продукта, который поможет пользователям эффективно управлять своим временем и достигать желаемых результатов.

Актуальность темы разработки данного приложения обусловлена высоким спросом на инструменты для организации рабочего времени. В современном мире люди сталкиваются с многочисленными задачами и дедлайнами, и отсутствие чёткого плана приводит к снижению продуктивности. Многие существующие решения, такие как «TimeTune» или «Sectograph», обладают полезными функциями, но имеют определённые ограничения, связанные с анализом временных затрат и гибкостью настройки. Разрабатываемое приложение «TimeLab» ориентировано на устранение этих недостатков и создание удобного инструмента для пользователей с различными уровнями подготовки в области тайм-менеджмента.

Основными задачами проекта являются:

* Разработка удобного интерфейса для управления задачами и временем;
* Создание функционала для анализа продуктивности и выявления неэффективных практик;
* Обеспечение интеграции с внешними календарными сервисами для удобства планирования;
* Разработка облачного хранения данных для обеспечения доступа к информации с разных устройств;
* Тестирование и оптимизация приложения для достижения высокой производительности и стабильности.

Таким образом, разработка мобильного приложения «TimeLab» представляет собой актуальный и востребованный проект, направленный на создание удобного инструмента для повышения продуктивности и оптимизации рабочего процесса пользователей.

1. Анализ предметной области

Приложение «TimeLab» для управления личным временем – это инновационное программное решение, разработанное для удобного и эффективного планирования задач и оптимизации рабочего процесса. Цель проекта заключается в создании инструмента, который автоматизирует процесс тайм-менеджмента, позволяя пользователям легко организовывать свои задачи.

Приложение предоставляет возможности для ввода и хранения информации о задачах, установки приоритетов. Особенностью приложения «TimeLab» является его интуитивно понятный интерфейс, который позволяет пользователям быстро освоить все функции и операции. Благодаря этому, даже те, кто не имеет опыта в управлении временем, смогут эффективно использовать данное приложение для повышения своей продуктивности и организации рабочего времени.

Предметная область тайм-менеджмента охватывает различные аспекты, включая планирование задач, отслеживание времени и анализ продуктивности. В условиях постоянного дефицита времени важно иметь инструмент, который поможет эффективно организовать рабочий процесс и повысить личную эффективность. В рамках данного курсового проекта будет проведен анализ предметной области, определены ключевые функции приложения «TimeLab» и выполнена сравнительная оценка существующих решений на рынке.

Ключевыми функциями приложения «TimeLab» являются:

Создание задач — возможность быстро добавить новую задачу с названием, описанием и датой выполнения.

Редактирование задач — удобное изменение информации о задаче, включая статус выполнения.

Удаление задач — возможность удалять ненужные задачи в один клик.

Сортировка по дате — автоматическая или ручная сортировка задач по дате выполнения.

Фильтрация выполненных задач — скрытие или отображение задач, уже отмеченных как выполненные.

Умные напоминания — система уведомлений, позволяющая не забыть о предстоящих задачах и важных дедлайнах.

Интуитивный интерфейс — современный дизайн с акцентом на простоту и удобство использования.

Отображение пустого состояния — визуальные подсказки при отсутствии задач для повышения вовлеченности пользователя.

1.1 Аналоги разрабатываемого приложения

«TimeTune» — это популярное приложение для тайм-менеджмента, которое помогает пользователям организовывать свое время и планировать задачи. Оно предлагает функции, такие как создание расписаний, отслеживание времени на выполнение задач и анализ продуктивности. «TimeTune» имеет простой и интуитивно понятный интерфейс, что делает его удобным для пользователей с разным уровнем опыта.

Преимущества:

* Удобство использования: «TimeTune» предлагает простой интерфейс, который позволяет быстро создавать расписания и управлять задачами.
* Анализ продуктивности: Приложение предоставляет отчеты о выполнении задач, что помогает пользователям оценивать свою продуктивность и вносить коррективы в планирование.
* Гибкость: Пользователи могут настраивать свои расписания и приоритеты в зависимости от личных потребностей и целей.

Недостатки:

* Ограниченный функционал: в сравнении с более комплексными решениями для управления временем, TimeTune может не предоставлять всех необходимых функций для более глубокого анализа или интеграции с другими приложениями.
* Платные функции: хотя базовая версия приложения бесплатна, некоторые функции могут быть доступны только в платной версии, что может ограничивать возможности пользователей.



Рисунок 1 – Логотип «TimeTune»



Рисунок 2 – Интерфейс «TimeTune»

Еще одним аналогом является «Sectograph» — это уникальное приложение для тайм-менеджмента, которое визуализирует расписание пользователя в виде круговой диаграммы, позволяя легко отслеживать и планировать задачи в течение дня. Оно предлагает пользователям возможность видеть, как распределено их время, что способствует более эффективному управлению задачами и повышению продуктивности.

Преимущества:

* Визуализация времени: Круговая диаграмма помогает пользователям наглядно увидеть, сколько времени отведено на каждую задачу, что делает планирование более интуитивным.
* Интеграция с календарями: Sectograph может синхронизироваться с популярными календарями (например, Google Calendar), что упрощает процесс управления задачами и событиями.
* Удобство использования: Приложение имеет простой интерфейс, который позволяет быстро добавлять и редактировать задачи.

Недостатки:

* Ограниченные функции анализа: хотя приложение хорошо справляется с визуализацией времени, оно может не предоставлять глубокий анализ продуктивности или отчетов о выполнении задач.
* Необходимость подключения к интернету: для синхронизации с календарями и получения обновлений требуется постоянное подключение к интернету, что может быть неудобно для некоторых пользователей.



Рисунок 3 – Логотип «Sectograph»

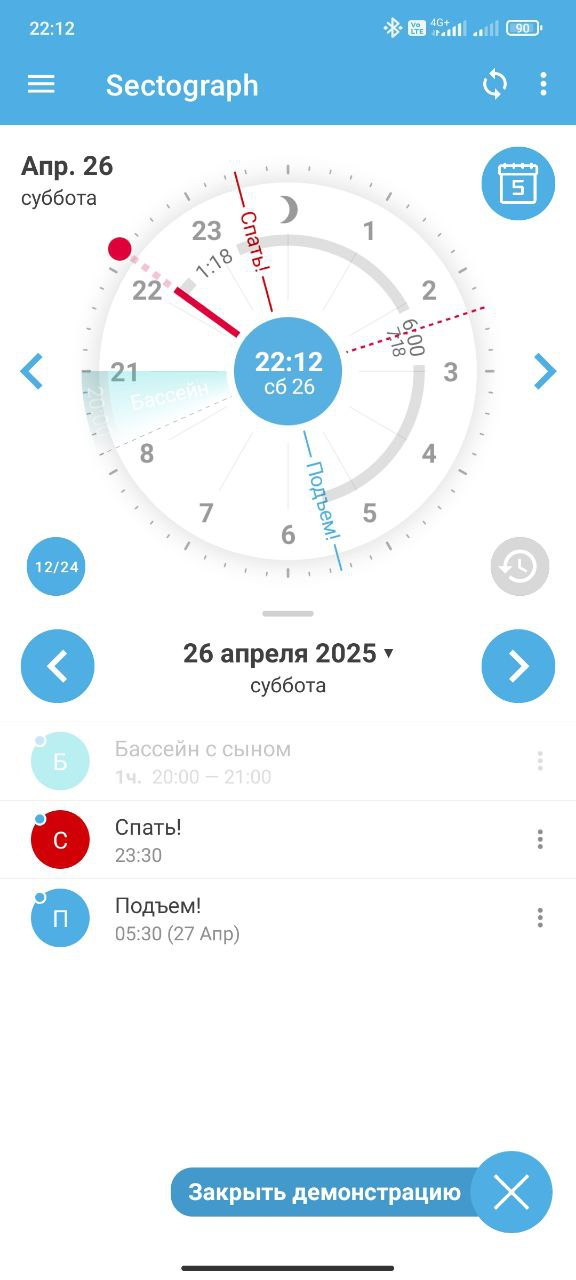


Рисунок 4 – Интерфейс «Sectograph»

1.2 Техническое задание

1. Общая информация:

Наименование: Мобильное приложение для управления временем «TimeLab».

Назначение и область применения: Программное приложение для мобильных устройств, предназначенное для планирования и учета и анализа времени, затраченного на выполнение задач и проектов.

1. Требование к программе:

Требования к функциональным характеристикам:

* Приложение должно быть разработано для платформы Android.
* Использовать локальную базу данных для хранения информации о задачах, проектах и временных затратах.
* Предоставить интуитивно понятный интерфейс с возможностью визуализации расписания календаря.
* Реализовать функционал для добавления, редактирования и удаления задач, а также возможность установки напоминаний.

Требования к надежности:

* Обеспечить стабильную работу приложения с использованием стандартных методов обработки ошибок и защиты целостности данных.
* Гарантировать безопасность пользовательских данных, включая шифрование конфиденциальной информации.

Требования к производительности:

* Оптимизировать алгоритмы обработки данных для быстрого выполнения операций добавления, редактирования и анализа задач.
* Минимизировать потребление ресурсов устройства во время работы приложения.

Требования к тестированию:

* Провести тестирование на соответствие требованиям производительности, надежности и функциональности приложения.
* Оценить все функции приложения, включая планирование задач и учет времени.

Требования к документации:

* Техническая документация должна включать полное техническое задание, спецификацию требований и документацию по программному продукту.
* Пользовательская документация должна содержать руководство пользователя с пошаговыми инструкциями по использованию приложения.

Требования к сопровождению:

* Обеспечить поддержку и регулярные обновления приложения после его внедрения в эксплуатацию.

1. Стадии разработки:

* Определение целей и задач разработки мобильного продукта.
* Анализ существующих аналогичных приложений на рынке.
* Изучение необходимых технологий и инструментов для разработки мобильного приложения.
* Проведение тестирования приложения на соответствие заданным критериям и устранение выявленных ошибок.

## 1.3 Описание структуры приложения

Приложение «TimeLab» предназначено для эффективного управления задачами и планирования времени. Оно разработано на языке Kotlin с использованием классического интерфейса на основе XML и архитектуры MVVM, что обеспечивает чёткое разделение логики и визуальной части.[3]

В качестве хранилища данных используется SharedPreferences, что позволяет сохранять задачи между сессиями работы приложения и быстро загружать их при повторном запуске.

Основная цель приложения — предоставить пользователю простой и надёжный инструмент для создания, редактирования и отслеживания повседневных задач.

Пользователь может:

* создавать задачи с названием, описанием, датой выполнения и статусом;
* редактировать задачи: обновлять текст, дату, статус выполнения;
* удалять задачи, которые больше не актуальны;
* сортировать задачи по дате для удобного планирования;
* скрывать выполненные задачи для фокусировки на текущих;
* добавлять напоминания о задачах через выбор даты и уведомления;
* сохранять все данные локально с помощью SharedPreferences, без необходимости подключения к интернету;
* использовать простой интерфейс, построенный на XML-разметке, с поддержкой BottomSheet и анимированных элементов управления.

# 2 Описание разработки приложения

## 2.1 Обоснование средств разработки

При разработке приложения TimeLab было принято решение использовать SharedPreferences для хранения данных о задачах и настройках приложения. SharedPreferences — это простой и удобный механизм хранения небольших объемов данных в виде пар "ключ-значение". Он идеально подходит для хранения конфигураций, пользовательских предпочтений и статусов задач в приложении.

Преимущества использования SharedPreferences:

* Простота и быстрота: SharedPreferences позволяет быстро сохранять и извлекать данные, такие как настройки пользователя или статусы задач, без необходимости сложной работы с базой данных.
* Поддержка синхронизации: SharedPreferences синхронизируется в фоновом режиме, что обеспечивает быстрый доступ к данным, не влияя на производительность приложения.
* Минимальные ресурсы: SharedPreferences не требует значительных ресурсов и идеально подходит для хранения небольших данных, таких как настройки времени или статусы выполнения задач.
* Безопасность данных: SharedPreferences позволяет хранить данные как в открытом виде, так и в зашифрованном формате, если требуется дополнительная защита информации.
* Использование SharedPreferences для хранения задач и настроек позволило снизить сложность разработки, обеспечив приложение надежным и быстрым механизмом для сохранения пользовательских данных.[2]

Язык программирования:

* Kotlin был выбран для разработки TimeLab благодаря следующим преимуществам:
* Чистый и лаконичный синтаксис: Kotlin позволяет писать код быстрее и с меньшим количеством ошибок по сравнению с Java.
* Совместимость с Java: полностью совместим с Java, что упрощает интеграцию с существующими проектами и библиотеками.
* Поддержка современных подходов: поддерживает корутины и другие современные конструкции, что упрощает работу с асинхронным кодом.
* Активная поддержка от Google: Kotlin является официальным языком для Android-разработки, что обеспечивает постоянное обновление и улучшение языка.[5]



Рисунок 5 – Логотип «Kotlin»

Среда разработки:

Android Studio используется для разработки приложения TimeLab и предлагает следующие преимущества:

* Интуитивно понятный интерфейс: легко осваиваемая среда с мощными функциями для редактирования кода.
* Инструменты для профилирования: встроенные инструменты для оптимизации производительности и анализа работы приложения.
* Поддержка эмуляторов и реальных устройств: возможность тестирования приложения на различных устройствах и эмуляторах для обеспечения стабильной работы.
* Отладка и тестирование: мощные возможности для отладки и написания тестов, что помогает создавать более стабильные и качественные приложения.[1]



Рисунок 6 – Логотип «Android Studio»

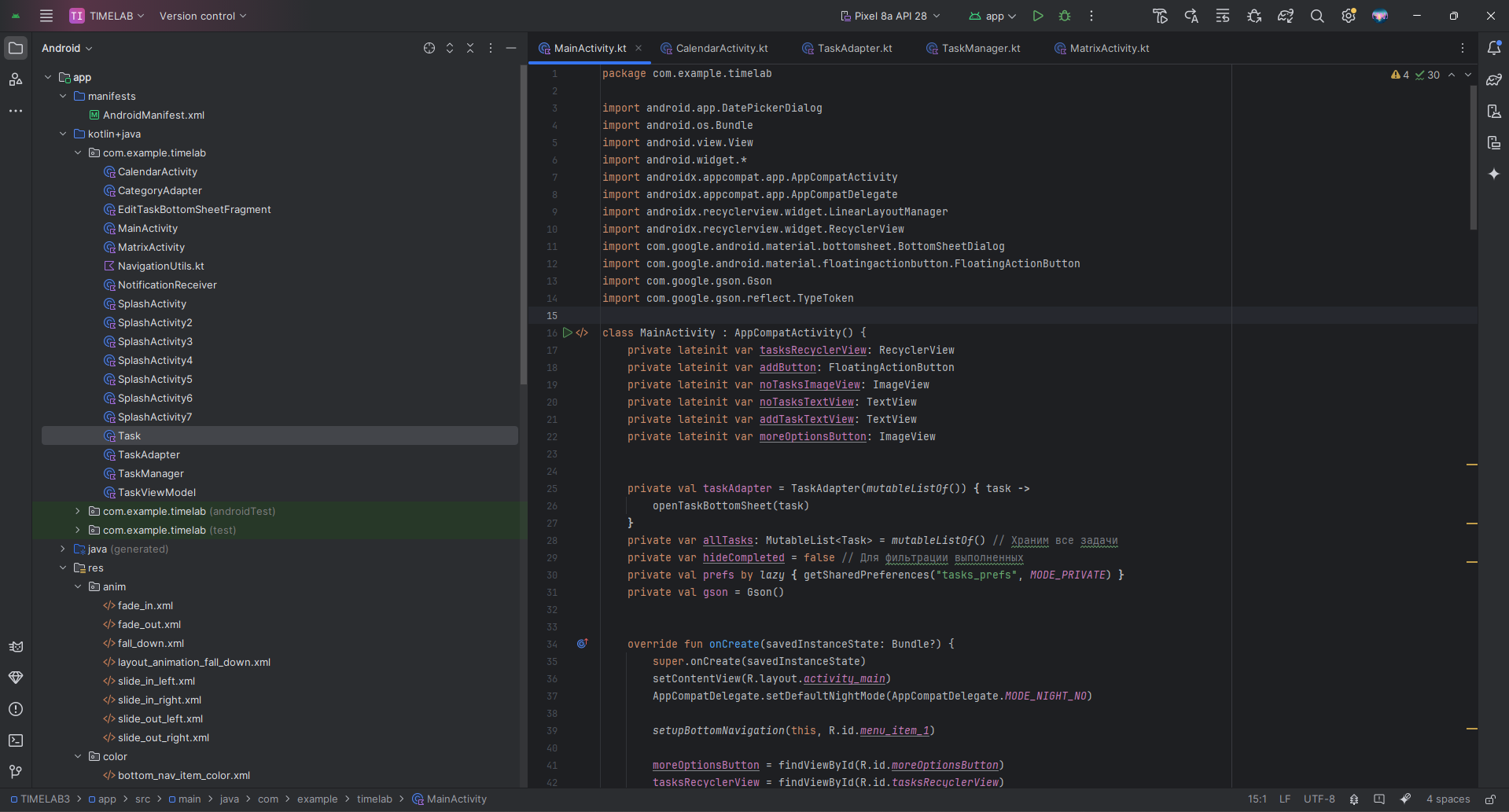


Рисунок 7 – Интерфейс «Android Studio»

## 2.2 Разработка интерфейса

При разработке интерфейса мобильного приложения «TimeLab» особое внимание было уделено простоте, удобству использования и визуальной чистоте. Основной задачей являлось создание понятной и минималистичной среды для работы с задачами, чтобы пользователь мог легко добавлять, редактировать и отслеживать выполнение своих дел.

Основные принципы разработки интерфейса:

* Интуитивность: все элементы управления логично расположены, а их использование не требует дополнительных инструкций;
* Минимализм: отсутствуют излишние детали, акцент сделан на задачах и основных действиях;
* Адаптивность: интерфейс корректно отображается на различных устройствах с разными размерами экранов;
* Эстетика: использован современный стиль оформления с использованием светлой темы и плавных анимаций.

Интерфейс приложения был начат с разработки макета в Figma. Макет представлен на рисунках с 8 по 13.

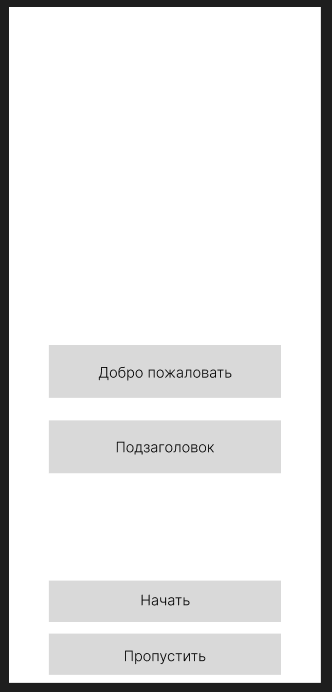


Рисунок 8 – Макет загрузочного экрана



Рисунок 9 – Макет главного экрана

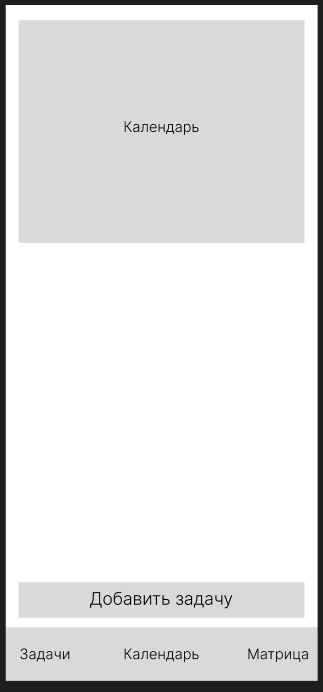


Рисунок 10 – Макет экрана «Календарь»



Рисунок 11 – Макет экрана «Матрица»

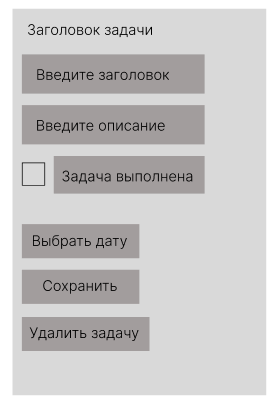


Рисунок 12 – Макет меню редактирования задачи

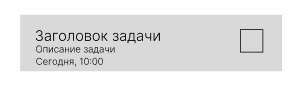


Рисунок 13 – Макет задачи

Далее макеты были доработаны уже в Android Studio и перенесены в код приложения. Итоговый интерфейс представлен на рисунках с 14 по 19.



Рисунок 14 – Загрузочный экран

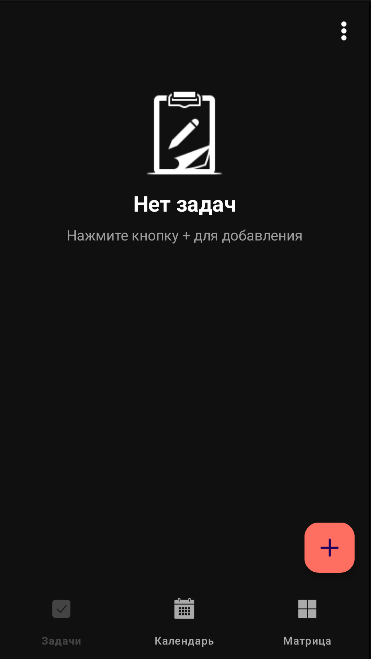


Рисунок 15 – Главный экран

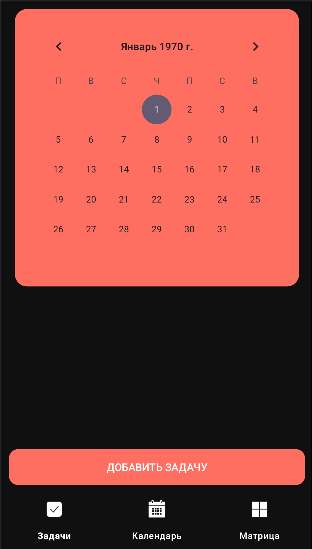


Рисунок 16 – Экран «Календарь»



Рисунок 17 – Экран «Матрица»

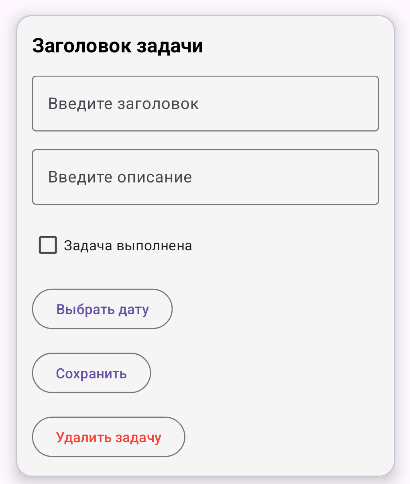


Рисунок 18 – Меню редактирования задачи

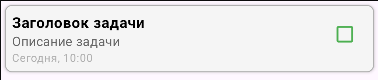


Рисунок 19 – Карточка задачи

## 2.3 Разработка логики работы программы. Схема взаимодействия компонентов проекта

2.3.1 Логика работы приложения

Логика работы мобильного приложения TimeLab строится на модульном подходе, где каждый компонент отвечает за выполнение определённой задачи. Приложение предоставляет пользователю основные функции управления временем и задачами, включая создание, редактирование, удаление и отслеживание выполнения задач. Все данные сохраняются локально.

Основные этапы взаимодействия пользователя с приложением:

* Создание задачи. Пользователь добавляет задачу, указывая название, описание, срок выполнения и приоритет. Все введенные данные сохраняются в локальной базе данных.
* Управление задачами. Пользователь может редактировать или удалять задачи, а также помечать их как выполненные, указывая фактическую дату выполнения.
* Обработка напоминаний. Приложение уведомляет пользователя о предстоящих задачах или сроках выполнения через встроенную систему уведомлений.
* Работа с локальной базой данных. Все данные, включая задачи и их статус, сохраняются локально с использованием SharedPreferences, что обеспечивает мгновенный доступ к информации даже без подключения к интернету.
* Обработка ошибок и отказоустойчивость. Приложение корректно обрабатывает возможные ошибки, такие как сбои при сохранении данных или неверный формат времени, и информирует пользователя о возникших проблемах.[6]

2.3.2 Схема взаимодействия компонентов

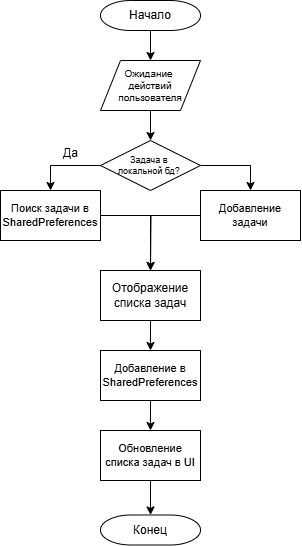


Рисунок 20 – Схема взаимодействия компонентов

## 2.4 Описание переменных, компонентов, классов и подпрограмм

Таблица 1 – Описание переменных, компонентов, классов и подпрограмм

|  |  |  |
| --- | --- | --- |
| Тип | Название | Описание |
| Класс | MainActivity | Основная активность, отображает список задач, позволяет добавлять, редактировать, удалять. |
| EditTaskBottomSheetFragment | Фрагмент в виде нижнего листа для редактирования задачи с уведомлениями и временем. |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
|  | Task | Модель данных для задачи: название, описание, дата, выполнена ли и т. д. |
| TaskAdapter | Адаптер для RecyclerView, отображает и обновляет список задач. |
| CalendarActivity | Активность для выбора даты задачи. |
| TimePickerFragment | Диалог выбора времени (часы и минуты). |
| Переменная | tasksRecyclerView | RecyclerView для отображения списка задач. |
| addButton | FloatingActionButton для добавления новой задачи. |
| noTasksImageView | ImageView, показывается, если нет задач. |
|  | noTasksTextView | TextView, отображает текст при отсутствии задач. |
|  | addTaskTextView | TextView, приглашение добавить первую задачу. |
|  | moreOptionsButton | ImageView, кнопка для вызова меню с дополнительными опциями. |
|  | taskAdapter | Адаптер для списка задач, обрабатывает отображение и редактирование. |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
|  | allTasks | Список всех задач (включая выполненные). |
| hideCompleted | Флаг, скрывать ли выполненные задачи. |
| Prefs | SharedPreferences для хранения задач. |
| Gson | Gson для сериализации/десериализации задач в JSON. |
| Title | Название задачи. |
| Description | Описание задачи. |
| Date | Дата задачи в формате строки. |
| isCompleted | Флаг, выполнена ли задача. |
| hasReminder | Флаг, есть ли напоминание. |
| reminderHour | Час напоминания. |
| reminderMinute | Минута напоминания. |
| Tasks | Список задач, отображаемый адаптером. |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
|  | calendarView | Компонент выбора даты. |
| confirmButton | Кнопка подтверждения выбранной даты. |
| Компонент | EditText | Поля редактирования названия, описания, даты. |
| CheckBox | Отметка о выполнении задачи, включение уведомлений. |
| Button | Кнопки для выбора даты, сохранения и удаления задачи. |
| TimePicker | Выбор времени напоминания. |

# 

# 3 Тестирование и установка программы

* Установка мобильного приложения

Для установки мобильного приложения необходимо получить .apk файл приложения.

После установки приложение готово к запуску и первичной проверке.

* Тестирование программы

Таблица 2 – Тест-кейсы к программе

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название тест-кейса | Описание | Ожидаемый результат |
| 1 | Установка APK-файла | Установить приложение из загруженного .apk-файла | Приложение успешно устанавливается и запускается |
| 2 | Запуск приложения | Открыть установленное приложение | Главное окно приложения открывается без ошибок |
| 3 | Добавление новой задачи | Ввести данные новой задачи и сохранить | Задача появляется в общем списке |
| 4 | Редактирование задачи | Изменить текст и дату у уже существующей задачи | Обновлённая информация отображается корректно |
| 5 | Отметка выполнения задачи | Отметить задачу как выполненную | Задача визуально изменяется |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Удаление задачи | Удалить существующую задачу из списка | Задача исчезает из общего списка |
| 7 | Открытие экрана редактирования | Открыть нижнюю панель для редактирования задачи по нажатию | BottomSheet с полями редактирования появляется на экране |
| 8 | Выбор даты задачи | Открыть календарь и выбрать новую дату | Новая дата сохраняется и отображается в задаче |
| 9 | Сохранение состояния | Перезапустить приложение после добавления задач | Все задачи остаются сохранёнными |



Рисунок 21 – Результат тестов 1 и 2



Рисунок 22 – Результат теста 3

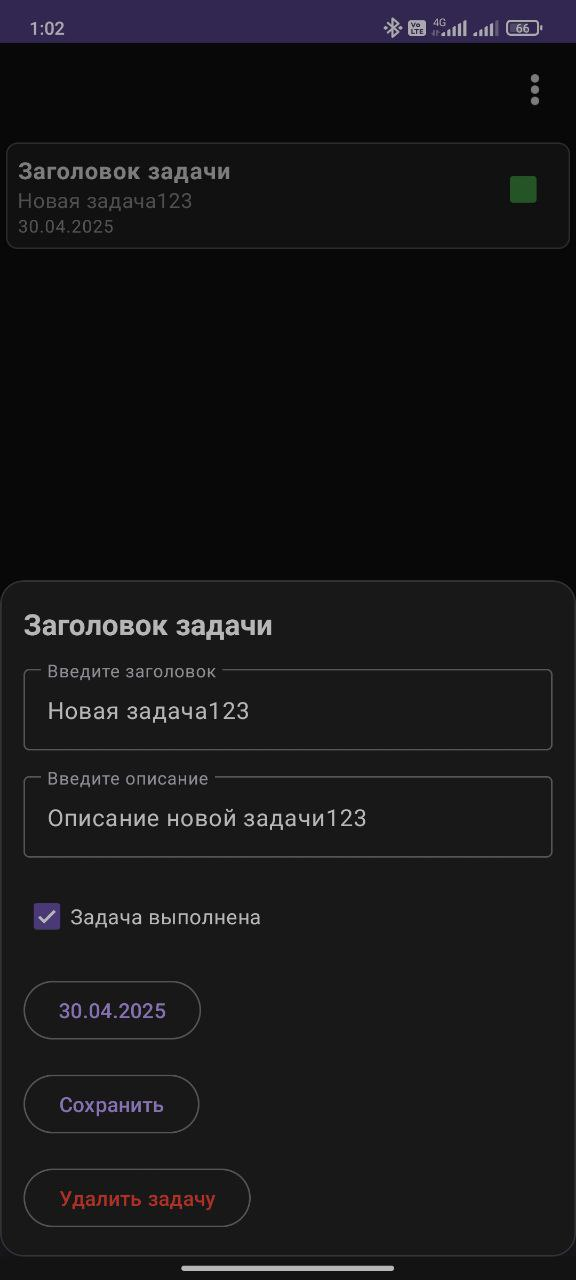


Рисунок 23 – Результат тестов 4, 5, 7, 8, 9



Рисунок 24 – Результат теста 6

После тестирования, можно сделать вывод, что все тест кейсы были успешны и программа работает правильно.[4]

Заключение

В ходе выполнения курсового проекта было разработано мобильное приложение «TimeLab», предназначенное для эффективного планирования задач и управления временем пользователей. В процессе работы были проанализированы существующие решения в области тайм-менеджмента, определены их достоинства и недостатки, на основе чего были сформулированы требования к собственному приложению.

Разработка велась на языке программирования Kotlin с использованием архитектуры MVVM и среды Android Studio. Для хранения данных использовались механизмы SharedPreferences, что обеспечило быстрое и надёжное сохранение пользовательских задач. Приложение обладает удобным и интуитивно понятным интерфейсом, поддерживает создание, редактирование, удаление и сортировку задач, установку напоминаний, а также фильтрацию выполненных задач. Особое внимание было уделено простоте взаимодействия пользователя с приложением и обеспечению высокой производительности.

В рамках тестирования приложения все основные функции были успешно проверены и подтвердили свою работоспособность. Программа корректно обрабатывает пользовательские действия, сохраняет данные между сессиями и предоставляет стабильную работу без сбоев.

Разработка «TimeLab» показала важность правильной организации процессов проектирования, кодирования и тестирования программного продукта. Полученные в ходе работы знания и навыки позволили успешно реализовать поставленные задачи и создать удобное мобильное приложение для повышения личной эффективности пользователей.

Результаты проекта могут служить основой для дальнейшего развития функционала, включая интеграцию с облачными сервисами, расширение аналитических возможностей и создание более гибкой системы напоминаний.

Список использованной литературы

1. Android Developers. Руководство по разработке приложений для Android [Электронный ресурс]. – Режим доступа: https://developer.android.com, свободный. – (дата обращения: 25.04.2025).
2. Блинов А.О., Шевченко Г.И. Разработка мобильных приложений для платформы Android: учебное пособие. – М.: БХВ-Петербург, 2020. – 320 с.
3. Макконнелл С. Совершенный код. Мастер-класс по разработке программного обеспечения. – 2-е изд. – М.: Вильямс, 2020. – 896 с.
4. Фаулер М. Архитектура корпоративных приложений. – СПб.: Питер, 2017. – 560 с.
5. Харди Дж., Филлипс Б. Android. Программирование для профессионалов. – М.: ДМК Пресс, 2021. – 600 с.
6. Шилдт Г. Java. Полное руководство. – 11-е изд. – М.: Вильямс, 2020. – 1280 с.

Приложение А

Листинг программы

MainActivity

package com.example.timelab

import android.app.DatePickerDialog

import android.os.Bundle

import android.view.View

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

import androidx.appcompat.app.AppCompatDelegate

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.google.android.material.bottomsheet.BottomSheetDialog

import com.google.android.material.floatingactionbutton.FloatingActionButton

import com.google.gson.Gson

import com.google.gson.reflect.TypeToken

class MainActivity : AppCompatActivity() {

private lateinit var tasksRecyclerView: RecyclerView

private lateinit var addButton: FloatingActionButton

private lateinit var noTasksImageView: ImageView

private lateinit var noTasksTextView: TextView

private lateinit var addTaskTextView: TextView

private lateinit var moreOptionsButton: ImageView

private val taskAdapter = TaskAdapter(mutableListOf()) { task ->

openTaskBottomSheet(task)

}

private var allTasks: MutableList<Task> = mutableListOf() // Храним все задачи

private var hideCompleted = false // Для фильтрации выполненных

private val prefs by lazy { getSharedPreferences("tasks\_prefs", MODE\_PRIVATE) }

private val gson = Gson()

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE\_NIGHT\_NO)

setupBottomNavigation(this, R.id.menu\_item\_1)

moreOptionsButton = findViewById(R.id.moreOptionsButton)

tasksRecyclerView = findViewById(R.id.tasksRecyclerView)

addButton = findViewById(R.id.addButton)

noTasksImageView = findViewById(R.id.noTasksImageView)

noTasksTextView = findViewById(R.id.noTasksTextView)

addTaskTextView = findViewById(R.id.addTaskTextView)

tasksRecyclerView.layoutManager = LinearLayoutManager(this)

tasksRecyclerView.adapter = taskAdapter

loadTasks()

addButton.setOnClickListener {

val currentDate = java.text.SimpleDateFormat("dd.MM.yyyy", java.util.Locale.getDefault()).format(java.util.Date())

val newTask = Task(

title = "Новая задача",

description = "Описание новой задачи",

dueDate = currentDate

)

addTask(newTask)

saveTasks()

}

updateEmptyState()

moreOptionsButton.setOnClickListener { view ->

val popup = PopupMenu(this, view)

popup.menuInflater.inflate(R.menu.top\_bar\_menu, popup.menu)

popup.setOnMenuItemClickListener { item ->

when (item.itemId) {

R.id.menu\_clear\_all -> {

clearAllTasks()

true

}

R.id.menu\_sort -> {

sortTasksByDate()

true

}

R.id.menu\_hide\_completed -> {

toggleHideCompleted()

true

}

else -> false

}

}

popup.show()

}

}

private fun addTask(task: Task) {

allTasks.add(task)

taskAdapter.submitList(allTasks)

updateEmptyState()

}

private fun updateEmptyState() {

val isEmpty = taskAdapter.itemCount == 0

noTasksImageView.visibility = if (isEmpty) View.VISIBLE else View.GONE

noTasksTextView.visibility = if (isEmpty) View.VISIBLE else View.GONE

addTaskTextView.visibility = if (isEmpty) View.VISIBLE else View.GONE

}

private fun openTaskBottomSheet(task: Task) {

val dialog = BottomSheetDialog(this)

val view = layoutInflater.inflate(R.layout.bottom\_sheet\_task, null)

val editTitle = view.findViewById<EditText>(R.id.editTitle)

val editDescription = view.findViewById<EditText>(R.id.editDescription)

val completedCheckbox = view.findViewById<CheckBox>(R.id.taskCompletedCheckbox)

val selectDateButton = view.findViewById<Button>(R.id.selectDateButton)

val saveButton = view.findViewById<Button>(R.id.saveButton)

val deleteButton = view.findViewById<Button>(R.id.deleteButton)

editTitle.setText(task.title)

editDescription.setText(task.description)

completedCheckbox.isChecked = task.isCompleted

selectDateButton.text = task.dueDate ?: "Выбрать дату"

selectDateButton.setOnClickListener {

val dialogDate = DatePickerDialog(this)

dialogDate.setOnDateSetListener { \_, year, month, day ->

val selectedDate = String.format("%02d.%02d.%d", day, month + 1, year)

task.dueDate = selectedDate

selectDateButton.text = selectedDate

}

dialogDate.show()

}

saveButton.setOnClickListener {

task.title = editTitle.text.toString()

task.description = editDescription.text.toString()

task.isCompleted = completedCheckbox.isChecked

taskAdapter.notifyDataSetChanged()

dialog.dismiss()

Toast.makeText(this, "Задача обновлена", Toast.LENGTH\_SHORT).show()

}

deleteButton.setOnClickListener {

val position = taskAdapter.tasks.indexOf(task)

if (position != -1) {

taskAdapter.tasks.removeAt(position)

taskAdapter.notifyItemRemoved(position)

}

dialog.dismiss()

updateEmptyState()

Toast.makeText(this, "Задача удалена", Toast.LENGTH\_SHORT).show()

}

dialog.setContentView(view)

dialog.show()

}

private fun clearAllTasks() {

allTasks.clear()

taskAdapter.submitList(allTasks)

updateEmptyState()

Toast.makeText(this, "Все задачи удалены", Toast.LENGTH\_SHORT).show()

}

private fun sortTasksByDate() {

allTasks = allTasks.sortedBy { it.dueDate }.toMutableList() // Сортируем и обновляем список

taskAdapter.submitList(allTasks)

Toast.makeText(this, "Сортировка по дате", Toast.LENGTH\_SHORT).show()

}

private fun toggleHideCompleted() {

hideCompleted = !hideCompleted

val filteredList = if (hideCompleted) {

allTasks.filter { !it.isCompleted }.toMutableList()

} else {

allTasks // Возвращаем все задачи

}

taskAdapter.submitList(filteredList) // Обновляем список в адаптере

val msg = if (hideCompleted) "Скрыты выполненные" else "Показаны все"

Toast.makeText(this, msg, Toast.LENGTH\_SHORT).show()

}

private fun saveTasks() {

val json = gson.toJson(allTasks)

prefs.edit().putString("task\_list", json).apply()

}

private fun loadTasks() {

val json = prefs.getString("task\_list", null)

if (json != null) {

val type = object : TypeToken<MutableList<Task>>() {}.type

allTasks = gson.fromJson(json, type)

taskAdapter.submitList(allTasks)

}

}

}

CalendarActivity

package com.example.timelab

import android.app.AlarmManager

import android.app.PendingIntent

import android.content.Context

import android.content.Intent

import android.os.Bundle

import android.widget.Button

import android.widget.CalendarView

import android.widget.TimePicker

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.google.gson.Gson

import com.google.gson.reflect.TypeToken

import java.util.\*

class CalendarActivity : AppCompatActivity() {

private lateinit var calendarView: CalendarView

private lateinit var taskRecyclerView: RecyclerView

private lateinit var taskAdapter: TaskAdapter

private val taskManager = TaskManager()

private var selectedDateString: String = ""

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_calendar)

setupBottomNavigation(this, R.id.menu\_item\_2)

calendarView = findViewById(R.id.calendarView)

taskRecyclerView = findViewById(R.id.taskRecyclerView)

// Настройка RecyclerView

taskAdapter = TaskAdapter(mutableListOf()) { selectedTask: Task ->

val editDialog = EditTaskBottomSheetFragment(

task = selectedTask,

onTaskUpdated = { updatedTask ->

taskManager.updateTask(selectedDateString, selectedTask, updatedTask)

taskManager.saveToPrefs(this)

val calendar = taskManager.parseDate(selectedDateString)

updateTasksForDate(calendar)

},

onTaskDeleted = { deletedTask ->

taskManager.removeTask(selectedDateString, deletedTask)

taskManager.saveToPrefs(this)

val calendar = taskManager.parseDate(selectedDateString)

updateTasksForDate(calendar)

Toast.makeText(this, "Задача удалена", Toast.LENGTH\_SHORT).show()

}

)

editDialog.show(supportFragmentManager, "EditTask")

}

taskRecyclerView.layoutManager = LinearLayoutManager(this)

taskRecyclerView.adapter = taskAdapter

// Восстановление задач из SharedPreferences

taskManager.loadFromPrefs(this)

// Установка текущей даты

val calendar = Calendar.getInstance()

selectedDateString = taskManager.formatDate(calendar)

updateTasksForDate(calendar)

// Обработка выбора даты

calendarView.setOnDateChangeListener { \_, year, month, dayOfMonth ->

calendar.set(year, month, dayOfMonth)

selectedDateString = taskManager.formatDate(calendar)

updateTasksForDate(calendar)

}

// Кнопка добавления задачи

findViewById<Button>(R.id.addTaskButton).setOnClickListener {

val newTask = Task(

title = "Новая задача",

description = "Описание",

dueDate = selectedDateString

)

taskManager.addTaskForDate(selectedDateString, newTask)

taskManager.saveToPrefs(this)

val calendar = taskManager.parseDate(selectedDateString)

updateTasksForDate(calendar)

Toast.makeText(this, "Задача добавлена", Toast.LENGTH\_SHORT).show()

}

}

private fun updateTasksForDate(date: Calendar) {

val tasks = taskManager.getTasksForDate(taskManager.formatDate(date))

taskAdapter.submitList(tasks.toMutableList())

}

fun setTaskReminder(task: Task, hour: Int, minute: Int) {

val alarmManager = getSystemService(Context.ALARM\_SERVICE) as AlarmManager

val intent = Intent(this, NotificationReceiver::class.java).apply {

putExtra("TASK\_TITLE", task.title)

putExtra("TASK\_DESCRIPTION", task.description)

}

val pendingIntent = PendingIntent.getBroadcast(

this, task.hashCode(), intent, PendingIntent.FLAG\_UPDATE\_CURRENT or PendingIntent.FLAG\_IMMUTABLE

)

val calendar = Calendar.getInstance().apply {

set(Calendar.HOUR\_OF\_DAY, hour)

set(Calendar.MINUTE, minute)

set(Calendar.SECOND, 0)

}

alarmManager.setExact(AlarmManager.RTC\_WAKEUP, calendar.timeInMillis, pendingIntent)

Toast.makeText(this, "Напоминание установлено на $hour:$minute", Toast.LENGTH\_SHORT).show()

}

}

TaskAdapter

package com.example.timelab

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.CheckBox

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

class TaskAdapter(

val tasks: MutableList<Task>,

private val onTaskClick: (Task) -> Unit,

) : RecyclerView.Adapter<TaskAdapter.TaskViewHolder>() {

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {

val view = LayoutInflater.from(parent.context)

.inflate(R.layout.item\_task, parent, false)

return TaskViewHolder(view)

}

override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {

val task = tasks[position]

holder.bind(task)

holder.itemView.setOnClickListener {

onTaskClick(task)

}

}

override fun getItemCount() = tasks.size

fun submitList(newTasks: List<Task>) {

tasks.clear()

tasks.addAll(newTasks)

notifyDataSetChanged()

}

class TaskViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

private val descriptionView: TextView = itemView.findViewById(R.id.taskDescription)

private val checkBox: CheckBox = itemView.findViewById(R.id.taskStatus)

private val taskDate: TextView = itemView.findViewById(R.id.taskDate)

fun bind(task: Task) {

descriptionView.text = task.title

checkBox.isChecked = task.isCompleted

taskDate.text = task.dueDate ?: "Без даты"

checkBox.isClickable = false

checkBox.isFocusable = false

}

}

}

TaskManager

package com.example.timelab

import android.content.Context

import com.google.gson.Gson

import com.google.gson.reflect.TypeToken

import java.text.SimpleDateFormat

import java.util.\*

class TaskManager {

private val tasksByDate = mutableMapOf<String, MutableList<Task>>()

private val tasksByCategory = mutableMapOf<String, MutableList<Task>>()

fun addTaskForDate(dateString: String, task: Task) {

val taskList = tasksByDate.getOrPut(dateString) { mutableListOf() }

taskList.add(task)

}

fun getTasksForDate(dateString: String): List<Task> {

return tasksByDate[dateString] ?: emptyList()

}

fun formatDate(date: Calendar): String {

val sdf = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())

return sdf.format(date.time)

}

fun saveToPrefs(context: Context) {

val shared = context.getSharedPreferences("tasks", Context.MODE\_PRIVATE)

val json = Gson().toJson(tasksByDate)

shared.edit().putString("tasks\_map", json).apply()

}

fun loadFromPrefs(context: Context) {

val shared = context.getSharedPreferences("tasks", Context.MODE\_PRIVATE)

val json = shared.getString("tasks\_map", null)

if (json != null) {

val type = object : TypeToken<MutableMap<String, MutableList<Task>>>() {}.type

val restored = Gson().fromJson<MutableMap<String, MutableList<Task>>>(json, type)

tasksByDate.clear()

tasksByDate.putAll(restored)

}

}

fun updateTask(dateKey: String, oldTask: Task, newTask: Task) {

val tasks = tasksByDate[dateKey] ?: return

val index = tasks.indexOfFirst { it == oldTask }

if (index != -1) {

tasks[index] = newTask

}

}

fun parseDate(dateString: String): Calendar {

val sdf = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())

val date = sdf.parse(dateString) ?: Date()

return Calendar.getInstance().apply { time = date }

}

fun removeTask(date: String, task: Task) {

tasksByDate[date]?.remove(task)

}

fun addTaskForCategory(category: String, task: Task) {

val taskList = tasksByCategory.getOrPut(category) { mutableListOf() }

taskList.add(task)

}

// Получение задач для конкретной категории

fun getTasksForCategory(category: String): List<Task> {

return tasksByCategory[category] ?: emptyList()

}

}

Приложение Б

Руководство пользователя

1. Назначение приложения

Приложение «TimeLab» предназначено для планирования задач, управления временем и повышения личной продуктивности пользователей. Оно позволяет создавать задачи, редактировать их, устанавливать напоминания, а также отслеживать выполнение целей в удобной форме.

1. Установка приложения

* Загрузите файл TimeLab.apk на ваше устройство.
* Откройте загруженный файл и подтвердите установку.
* После установки ярлык приложения появится на главном экране устройства.

1. Работа с приложением

После запуска приложения, Вы попадете на экран приветствия.



Рисунок 1 – Экран приветствия пользователя

Дальше Вы можете либо пропустить информацию о функциях приложения, нажав кнопку «Пропустить» и попадете сразу на главный экран. Либо просмотреть информацию о функциях приложения, нажав кнопку «Начать».

На главном экране Вы можете добавлять простые задачи и редактировать их.



Рисунок 2 – Главный экран приложения

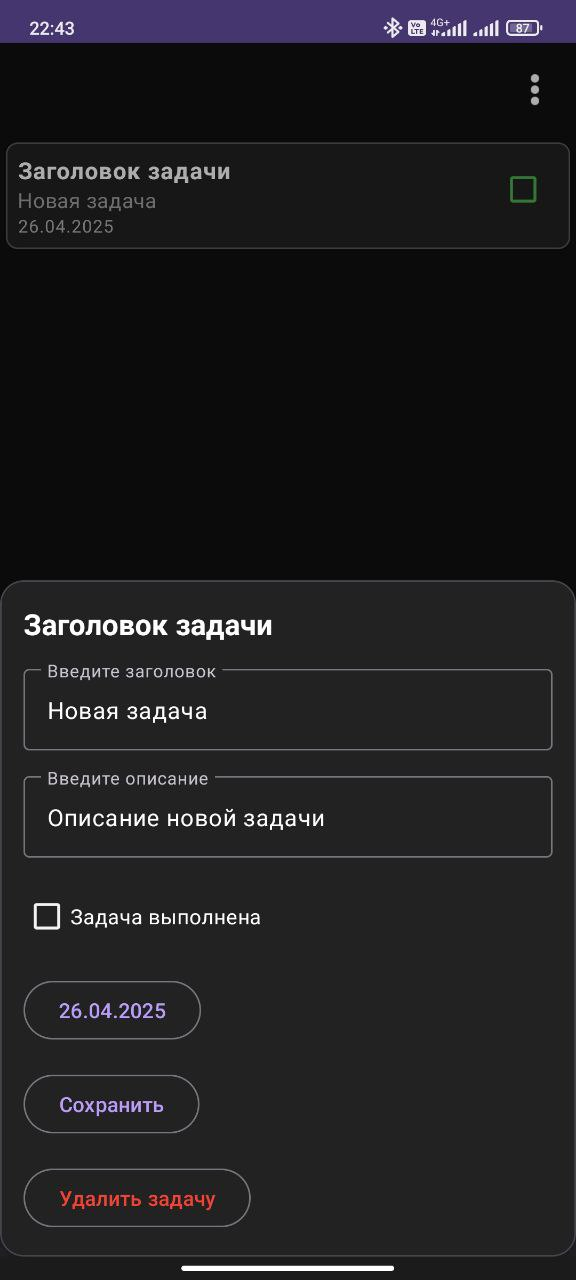


Рисунок 3 – Главный экран с меню редактирования задачи

В меню три точки можно отсортировать добавленные задачи по дате, удалить все и скрыть уже выполненные задачи.



Рисунок 4 – Главный экран с меню три точки

На экране «Календарь» можно добавлять задачи для выбранной в календаре дате, а также редактировать их и ставить напоминание.

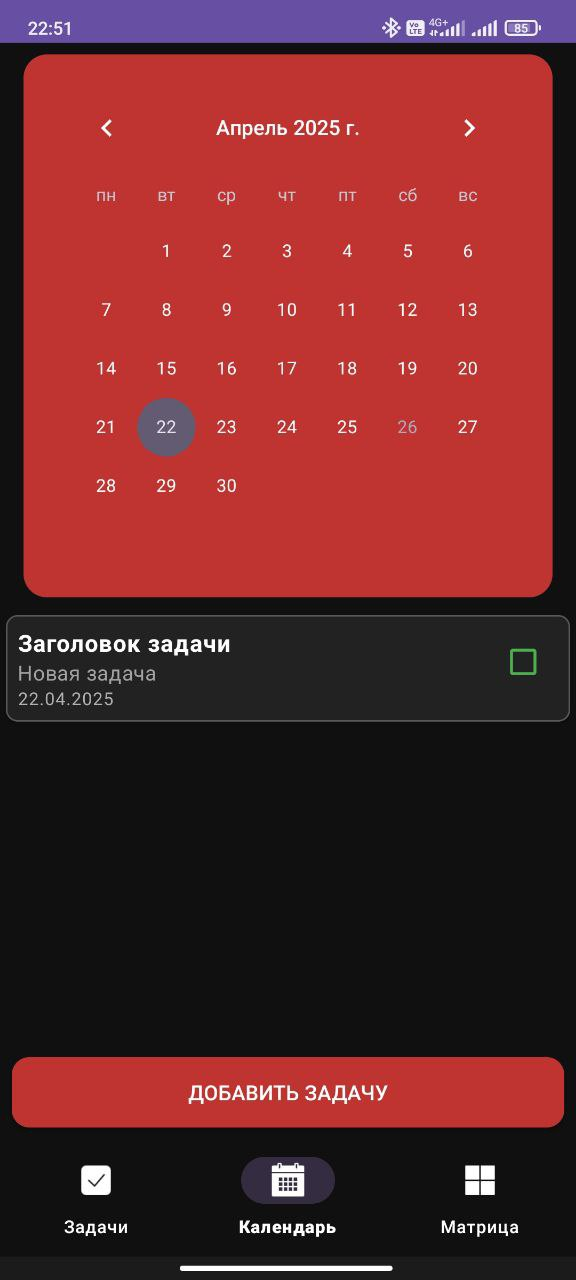


Рисунок 5 – Экран «Календарь»

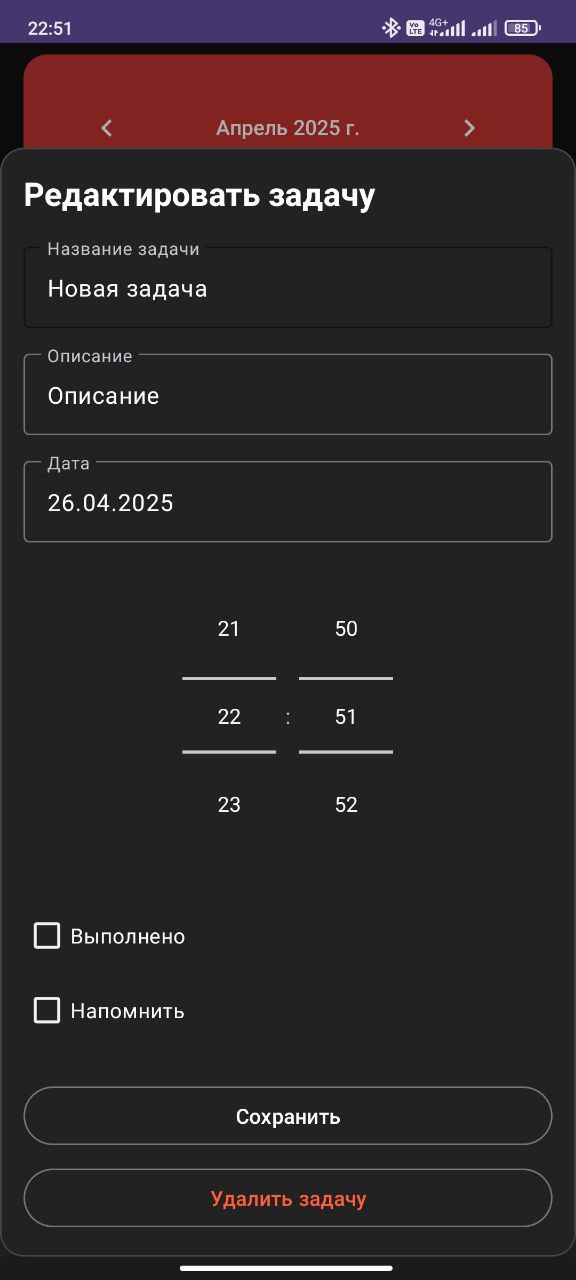


Рисунок 6 – Экран «Календарь» с меню редактирования задачи

На экране «Матрица», есть Матрица Эйзенхауэра, в которую можно добавлять задачи по важности, а также редактировать их и ставить напоминание.



Рисунок 7 – Экран «Матрица»

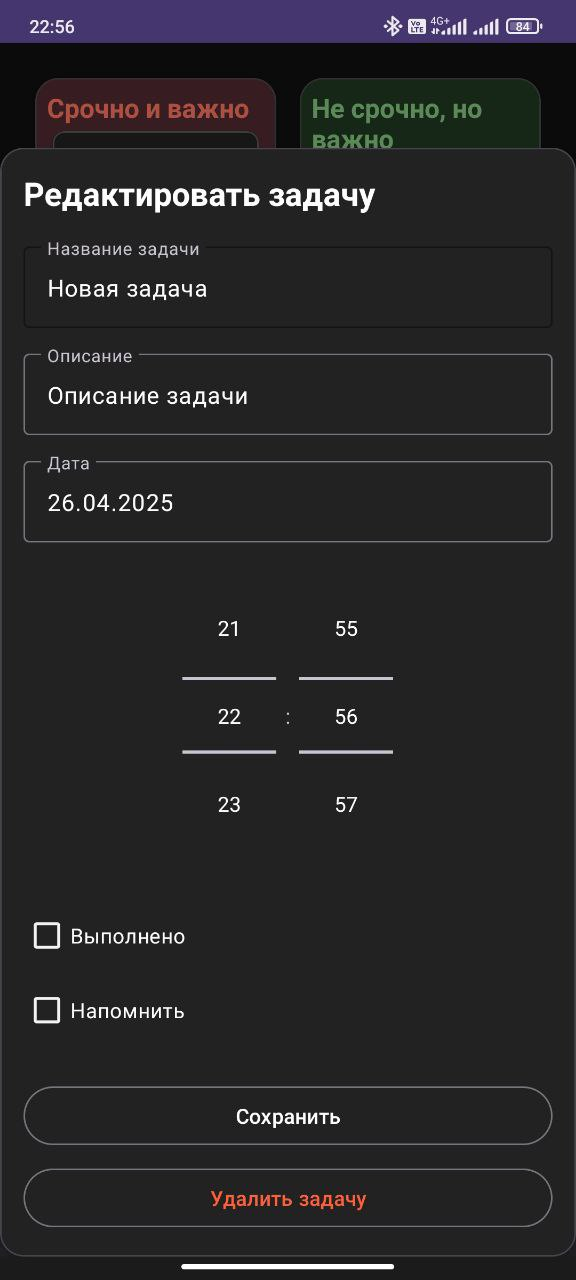


Рисунок 8 – Экран «Матрица» с меню редактирования задачи