

# Общее описание

Это Android-приложение, написанное на Kotlin с использованием Jetpack Compose для отслеживания времени, потраченного на работу, личную жизнь и отдых.

## Основные файлы

### MainActivity.kt

Является основной точкой входа в приложение Work-Life-Rest Tracker. Это главная Activity, которая:

- Запрашивает необходимые разрешения
- Настраивает навигацию между экранами
- Применяет тему приложения

## Основные компоненты

MainActivity - Класс, наследуемый от `ComponentActivity`, является главной Activity приложения.

### Методы:

`onCreate()` - Основной метод инициализации Activity:

1. Проверяет и запрашивает разрешения (для Android 13+)  
Для версий Android 13 и выше запрашивается `Manifest.permission.POST_NOTIFICATIONS` - разрешение на показ уведомлений
2. Устанавливает Compose-контент
3. Настраивает навигацию между экранами - Реализовано с помощью `NavHost` и `rememberNavController()`:

## Экраны

### 1. MainScreen.kt

Главный экран приложения, служащий точкой входа для навигации между основными функциями. Экран реализован с использованием Jetpack Compose.

## Компоновка

- Вертикальный `Column`, занимающий все доступное пространство
- Элементы выровнены по центру по горизонтали
- Основное содержимое расположено по центру экрана

## Элементы интерфейса

1. **Заголовок приложения** - текст "Work-Life-Rest"
2. **Кнопка "Трекер"** - переход на экран трекера времени
3. **Кнопка "Советчик"** - переход на экран с рекомендациями

## 2. AdvisorScreen.kt

Экран с рекомендациями по балансу работы, личной жизни и отдыха на основе статистики пользователя. Экран реализован с использованием Jetpack Compose.

### Компоненты экрана

#### Главная compose AdvisorScreen

- Получает статистику через `TrackerData`
- Управляет навигацией через `NavController`
- Содержит вертикальный скролл

### Вспомогательные compose

#### RecommendationSection

- Отображает заголовок "Рекомендация"
- Показывает сгенерированный совет

#### AdviceSection

- Список из советов ( `AdviceItem` )
- Управляет состоянием раскрытия через `expandedStates`
- Использует `ExpandableAdviceButton` для каждого совета

#### ExpandableAdviceButton

- Кнопка с заголовком и иконкой стрелки
- Анимированное раскрытие содержимого
- Изменяет иконку в зависимости от состояния

## BackButton

- Кнопка возврата с иконкой стрелки
- Выравнивание по правому краю

## Логика работы

generateRecommendation - Расчет процентов и сравнение с идеалом

private fun buildRecommendationText(...): String -Сравнивает текущие % с идеальными. Формирует строку с конкретными советами

data class AdviceItem - Модель для совета

## 3. TrackerScreen.kt

Экран содержит реализацию экрана трекера баланса работы, жизни и отдыха в приложении Work-Life-Rest Tracker. Экран состоит из двух вкладок: "Таймер" и "Статистика".

## Основные компоненты

### 1. TrackerScreen

Главная compose функция, представляющая экран трекера.

**Параметры:**

- navController: NavController - контроллер навигации для управления переходами между экранами

**Функционал:**

- Содержит TabRow с двумя вкладками: "Таймер" и "Статистика"
- Отображает соответствующее содержимое в зависимости от выбранной вкладки
- Включает кнопку "Назад" для возврата на предыдущий экран

### 2. TimerTab

Compose функция, реализующая вкладку таймера.

**Функционал:**

- Отображает текущее время таймера в формате MM:SS
- Показывает прогресс выполнения в виде линейного индикатора
- Предоставляет кнопки управления:

- Старт/Продолжить
- Пауза
- Сброс
- Отображает сообщение "Время вышло!" по завершении таймера
- Показывает диалог отчета после завершения таймера

### 3. ReportDialog

Диалоговое окно для ввода отчета о распределении времени.

**Параметры:**

- `onDismiss: () -> Unit` - функция закрытия диалога
- `onSubmit: (Int, Int, Int) -> Unit` - функция отправки отчета

**Функционал:**

- Позволяет выбрать количество минут для каждой категории (Work, Life, Rest)
- Проверяет, что сумма минут равна 60
- Отображает ошибку, если сумма не соответствует требованию

### 4. DiagramTab

Compose функция, реализующая вкладку статистики.

**Параметры:**

- `repo: TrackerData` - источник данных для статистики

**Функционал:**

- Отображает круговую диаграмму (PieChart) с распределением времени
- Показывает процентное соотношение категорий
- Отображает легенду с цветовыми индикаторами
- Показывает сообщение "Нет данных для отображения", если статистика отсутствует

### 5. PieChart

Compose функция для отрисовки круговой диаграммы.

**Параметры:**

- `slices: List<Triple<Int, String, Color>>` - данные для диаграммы (значение, метка, цвет)
- `total: Int` - общая сумма значений

## 6. Вспомогательные функции

- `formatTime(millis: Long): String` - форматирует миллисекунды в строку MM:SS
- `BackButton(onClick: () -> Unit)` - кнопка "Назад"

## **\*\*Данные**

### TimerState.kt

Это data-класс, представляющий состояние таймера в приложении. Он используется для хранения и передачи текущих параметров таймера между компонентами приложения

### Свойства класса

Свойство	Описание
<code>totalTimeMillis</code>	Общее время таймера в миллисекундах
<code>remainingTimeMillis</code>	Оставшееся время в миллисекундах
<code>isRunning</code>	Флаг, указывающий запущен ли таймер
<code>isFinished</code>	Флаг, указывающий завершился ли таймер

### TrackerData.kt

Класс для работы с хранилищем данных приложения, использующий Android DataStore для сохранения и обработки статистики времени по категориям: работа, личная жизнь и отдых.

### Объект TrackerKeys

Ключ	Описание
WORK	Минуты работы
LIFE	Минуты личной жизни
REST	Минуты отдыха
TOTAL	Общее количество минут

### Класс TrackerData

### Основные свойства и методы

## Свойства:

- `stats: Flow<Triple<Int, Int, Int>>` - поток данных со статистикой в формате (работа, жизнь, отдых)

## Методы:

- `addReport(work: Int, life: Int, rest: Int)` - добавляет новую запись времени по категориям

# Timer

## TimerService.kt

Android Service, который управляет таймером обратного отсчета с возможностью работы в качестве сервиса переднего плана. Он предоставляет функциональность таймера, которым можно управлять через различные действия, и сохраняет свое состояние через TimerState LiveData.

## Основные компоненты

### Константы Companion Object

- `NOTIFICATION_ID` : ID для постоянного уведомления таймера (1)
- `CHANNEL_ID` : ID канала для уведомлений таймера ("timer\_channel")
- `FINISH_CHANNEL_ID` : ID канала для уведомлений о завершении ("timer\_finish\_channel")
- `ACTION_START_TIMER` : Действие Intent для запуска таймера
- `ACTION_PAUSE_TIMER` : Действие Intent для паузы таймера
- `ACTION_STOP_TIMER` : Действие Intent для остановки таймера

## Свойства

- `binder` : Экземпляр `TimerBinder` для привязки сервиса
- `countDownTimer` : Экземпляр `Android CountDownTimer`
- `timerState` : `MutableLiveData<TimerState>` содержит текущее состояние таймера

## Методы

### Публичные методы

- `startTimer()` : Запускает таймер обратного отсчета и переводит сервис в передний план
- `pauseTimer()` : Ставит таймер на паузу и убирает статус переднего плана

- `stopTimer()` : Полностью останавливает таймер и завершает сервис
- `resetTimer()` : Сбрасывает состояние таймера без влияния на жизненный цикл сервиса

#### Методы жизненного цикла сервиса

- `onBind(intent: Intent)` : Возвращает сервисный binder
- `onCreate()` : Инициализирует каналы уведомлений
- `onStartCommand(intent: Intent?, flags: Int, startId: Int)` : Обработывает действия таймера из Intent

#### Методы уведомлений

- `createNotificationChannel()` : Настраивает каналы уведомлений для Android 8+
- `createNotification()` : Создает постоянное уведомление таймера с элементами управления
- `updateNotification()` : Обновляет уведомление текущим временем
- `showFinishedNotification()` : Показывает оповещение о завершении таймера

#### Вспомогательные методы

- `formatTime(millis: Long)` : Форматирует миллисекунды в строку "мм:сс"

### Внутренние классы

**TimerBinder** - Класс Binder, предоставляющий доступ к экземпляру TimerService:

- `getService()` : Возвращает экземпляр TimerService

### TimerViewModel.kt

Это ViewModel, которая управляет взаимодействием между UI и TimerService. Она предоставляет интерфейс для управления таймером и наблюдения за его состоянием. Наследуется от AndroidViewModel для доступа к контексту приложения

### Основные компоненты

#### Свойства

Свойство	Описание
<code>timerService</code>	Ссылка на связанный сервис
<code>bound</code>	Флаг состояния привязки

Свойство	Описание
<code>_timerState</code>	Внутреннее состояние таймера
<code>timerState</code>	Публичное состояние для наблюдения

## ServiceConnection

Обрабатывает события привязки/отвязки сервиса:

- `onServiceConnected()` : вызывается при успешной привязке
- `onServiceDisconnected()` : вызывается при неожиданном отключении

## Жизненный цикл

### Инициализация

1. При создании автоматически вызывает `bindToService()`
2. Устанавливает соединение с `TimerService`

### Очистка

- `onCleared()` : автоматически отвязывает сервис при уничтожении `ViewModel`

## Методы управления

`startTimer()`

- Запускает таймер через `Intent`
- Использует действие `TimerService.ACTION_START_TIMER`

`pauseTimer()`

- Ставит таймер на паузу через `Intent`
- Использует действие `TimerService.ACTION_PAUSE_TIMER`
- `resetTimer()`
- Сбрасывает состояние таймера
- Вызывает метод `resetTimer()` непосредственно у сервиса

## Theme.kt

Определяет тему приложения `Work-Life-Rest Tracker`, включая цветовые схемы для светлого и темного режимов, а также настройки статус-бара.



# Основные компоненты

## 1. Цветовые схемы

DarkColorSchem - Темная цветовая схема

LightColorScheme - Светлая цветовая схема

## 2. WorkLifeRestTheme

Основная функция темы приложения.

### Функционал:

- Автоматически определяет, использовать ли темную тему, на основе системных настроек
- Настраивает цвет статус-бара в соответствии с выбранной темой
- Применяет выбранную цветовую схему и типографику ко всему содержимому

## Используемые библиотеки и технологии

### 1. Jetpack Compose

- Для построения современного UI
- Используется для всех экранов приложения

### 2. DataStore

- Для хранения данных пользователя
- Используется в TrackerData.kt

### 3. Navigation Compose

- Для навигации между экранами
- Реализовано в MainActivity.kt

### 4. Material3

- Для стилизации UI компонентов
- Используется в теме приложения

### 5. Coroutines

- Для асинхронных операций
- Используется в работе с DataStore и таймером

## Требования к системе

- Android API 31 (Android 12) или выше
- Поддержка Jetpack Compose
- Разрешение на отправку уведомлений