



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практическим работам №9-10

по дисциплине «Системная и программная инженерия»

Выполнили:

Студенты группы ИКБО-11-22

Берчик А.С.
Андрусенко Л.Д.
Гришин А.В.
Малкин Г.Д.
Гоппен С.Д.

Проверил:

Преподаватель Михайлова Е.К.

Москва 2025

СОДЕРЖАНИЕ

1. ОРГАНИЗАЦИЯ РАЗРАБОТКИ.....	3
1.1. Выбор методологии управления процессом разработки.....	3
1.2. Создание удалённого git-репозитория.....	3
1.3. Стек технологий.....	4
2. ДОКУМЕНТИРОВАНИЕ РАЗРАБОТКИ ПО.....	5
2.1. Документация разработчика.....	5
2.2 Документация пользователя.....	5

1. ОРГАНИЗАЦИЯ РАЗРАБОТКИ

1.1. Выбор методологии управления процессом разработки

Для управления проектом UWasting была выбрана методология Scrum — гибкий подход (Agile), ориентированный на итеративную разработку и оперативное реагирование на изменения требований. Основные принципы Scrum:

Спринты: Работа разделена на двухнедельные итерации, в рамках которых реализуется набор функций из бэклога продукта.

Роли:

Scrum Master — обеспечивает соблюдение процессов и устраняет препятствия.

Product Owner — формирует приоритеты задач.

Команда разработчиков — отвечает за выполнение задач спринта.

Артефакты:

Бэклог продукта — список всех требований к приложению.

Бэклог спринта — задачи, выбранные для текущей итерации.

Инкремент — рабочий результат спринта.

Scrum был выбран из-за необходимости быстрой адаптации к изменениям и эффективного распределения задач между членами команды.

Для управления задачами использовался YouGile.

1.2. Создание удалённого git-репозитория

Командой разработчиков был выбран сервис GitHub для управления удалёнными репозиториями. Страницы созданных репозиторийев можно просмотреть на рисунке 1.1.

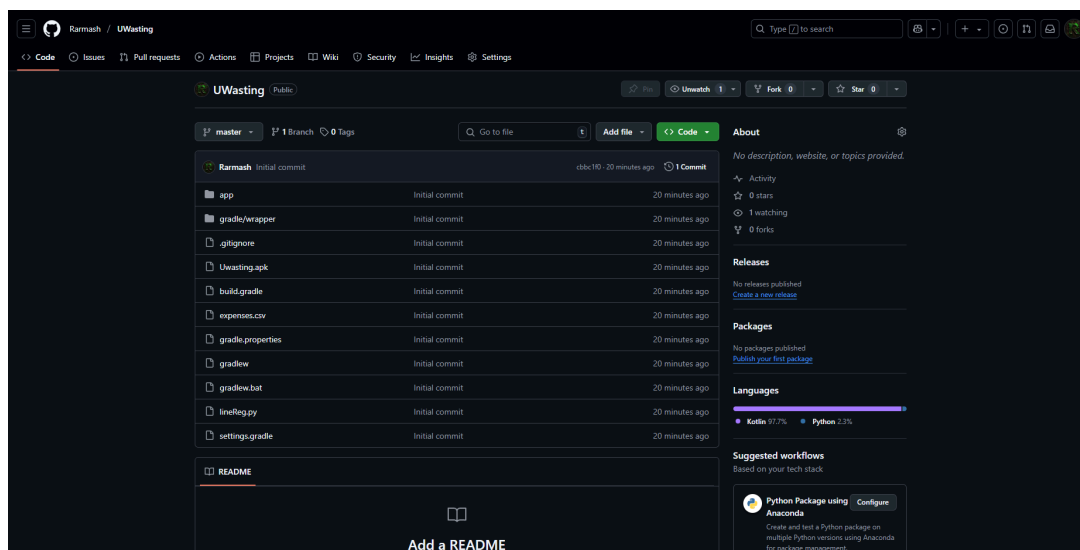


Рисунок 1 – Страница созданного git-репозитория в GitHub

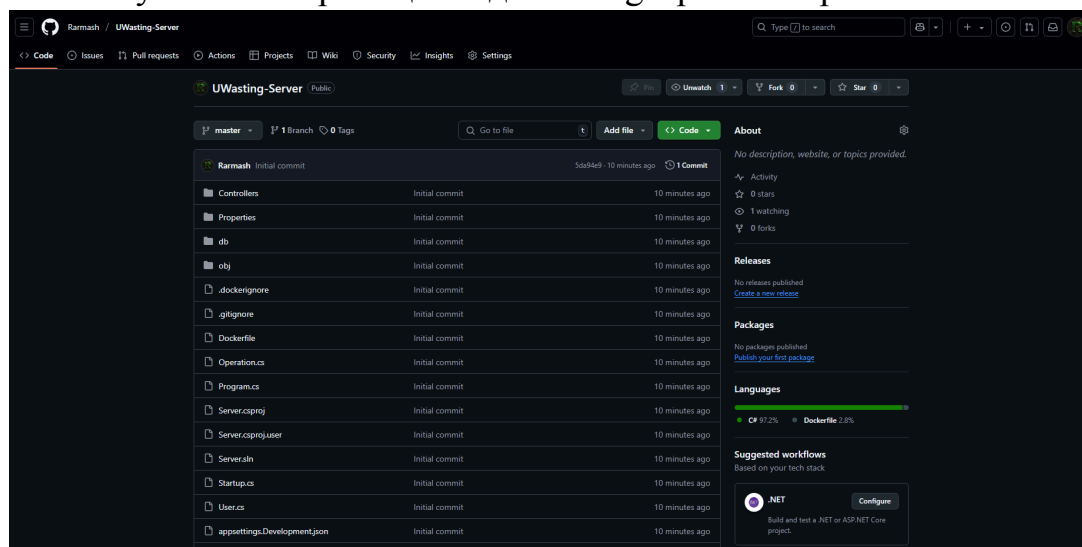


Рисунок 2 – Страница созданного git-репозитория в GitHub

1.3. Стек технологий

В процессе разработки программного обеспечения UWasting — мобильного приложения для учёта расходов и контроля финансов —

использовался следующий стек технологий. Он включает современные средства и подходы, обеспечивающие надёжность, удобство сопровождения и гибкость разработки.

1 Клиентская часть

Язык программирования:

- **Kotlin** — используется для создания Android-приложения, предоставляющего пользователю интерфейс для ведения и анализа личных финансов.

Фреймворки и библиотеки:

- **AndroidX (core-ktx, appcompat, lifecycle-viewmodel)** — обеспечивает совместимость с современными Android-компонентами и поддержку архитектурных подходов (MVVM).
- **Material Components** — библиотека компонентов интерфейса от Google, обеспечивающая визуальную согласованность и удобство взаимодействия.
- **Retrofit + Gson + OkHttp** — надёжный стек для сетевого взаимодействия с сервером по REST API.
 - *Retrofit* обеспечивает формирование запросов;
 - *Gson* — преобразование JSON в объекты Kotlin и обратно;
 - *OkHttp* с логирующим интерцептором — базовый HTTP-клиент с возможностью отслеживания запросов.
- **RxJava2 (RxAndroid)** — используется для асинхронной обработки данных и реактивного программирования.
- **MPAndroidChart** — библиотека для визуализации данных в виде графиков и диаграмм, применяется для отображения финансовой статистики.

- **Kotlin-Statistics** — облегчает выполнение математических операций и анализа (например, линейной регрессии) при работе с пользовательскими финансовыми данными.

Инструменты сборки и тестирования:

- **Gradle** — система управления зависимостями и автоматизации сборки проекта.
- **JUnit, Espresso** — библиотеки для модульного и UI-тестирования приложения.

Форматы данных и вспомогательные библиотеки:

- **Apache Commons CSV** — используется для работы с файлами CSV, например, при импорте или экспорте пользовательских расходов.

2 Серверная часть

Язык программирования:

- **C#** — основной язык серверной части. Выбран за счёт высокой производительности, читаемости и тесной интеграции с .NET-платформой.

Фреймворк и архитектура:

- **ASP.NET Core** — современный кроссплатформенный фреймворк для разработки веб-приложений и API. В проекте используется для реализации REST-интерфейса, предоставляющего доступ к функциям учёта и анализа финансов.
- **Архитектура MVC** — применяется для структурирования проекта. Обеспечивает разделение логики, данных и представления, что упрощает сопровождение и развитие системы.

Сетевое взаимодействие:

- **REST API** — серверная часть предоставляет интерфейс, к которому обращается клиентское приложение.
- **Формат данных JSON** — используется для обмена данными между клиентом и сервером.

Хранение данных:

- **PostgreSQL** — реляционная база данных, используемая для долговременного хранения пользовательских данных, включая расходы, категории и связанные с ними параметры.
- **Entity Framework Core** — ORM (Object-Relational Mapping), применяемый для удобной работы с базой данных. Позволяет выполнять запросы к таблицам с помощью LINQ и автоматизировать миграции.

Развёртывание и запуск:

- Серверная часть запускается локально или на собственном сервере.
- **Docker** — используется для контейнеризации приложения и упрощения его развёртывания.
- **Docker Compose** — позволяет запускать приложение и базу данных PostgreSQL как связанные сервисы в едином окружении.

2. ДОКУМЕНТИРОВАНИЕ РАЗРАБОТКИ ПО

2.1. Документация разработчика

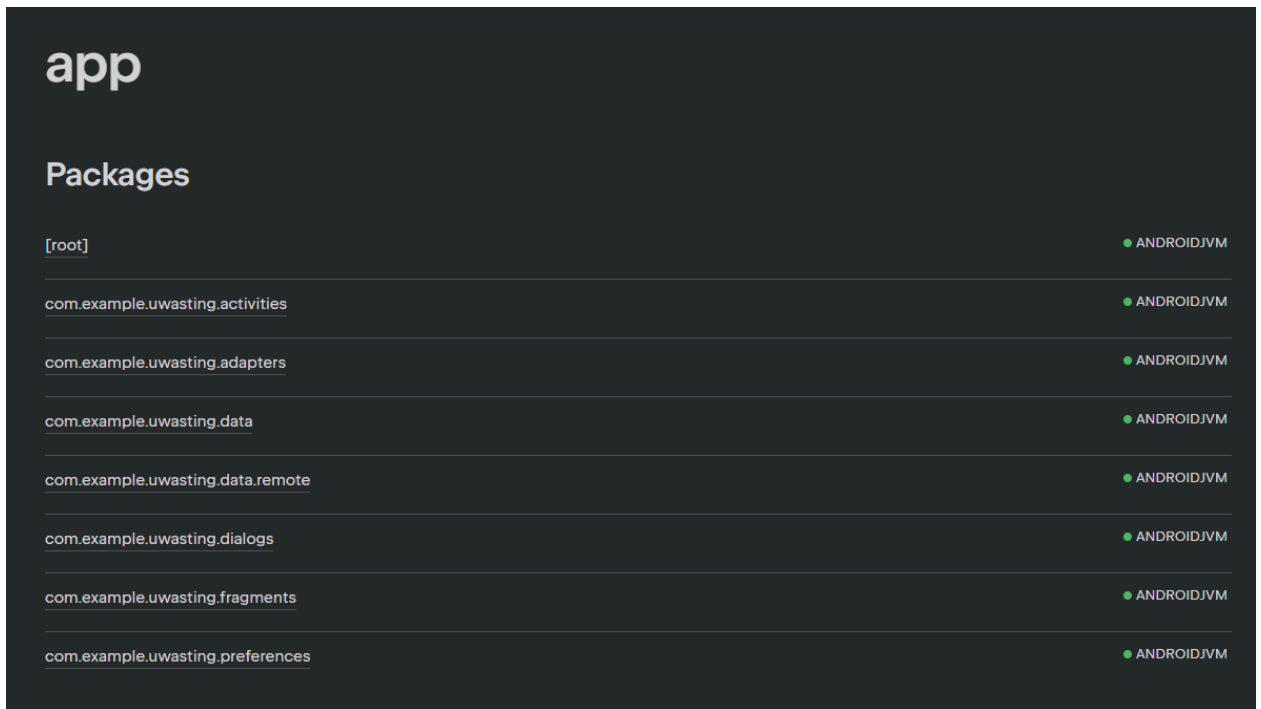


Рисунок 4 – Документация разработчика

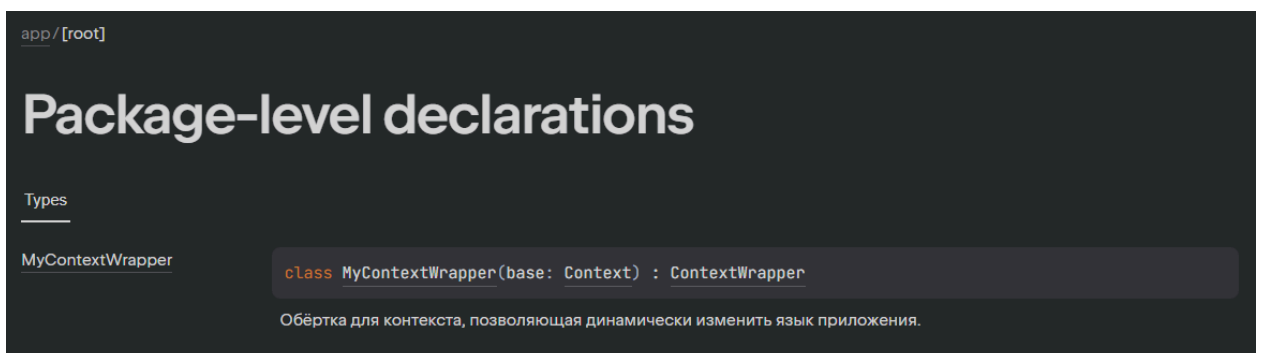


Рисунок 5 – Документация разработчика

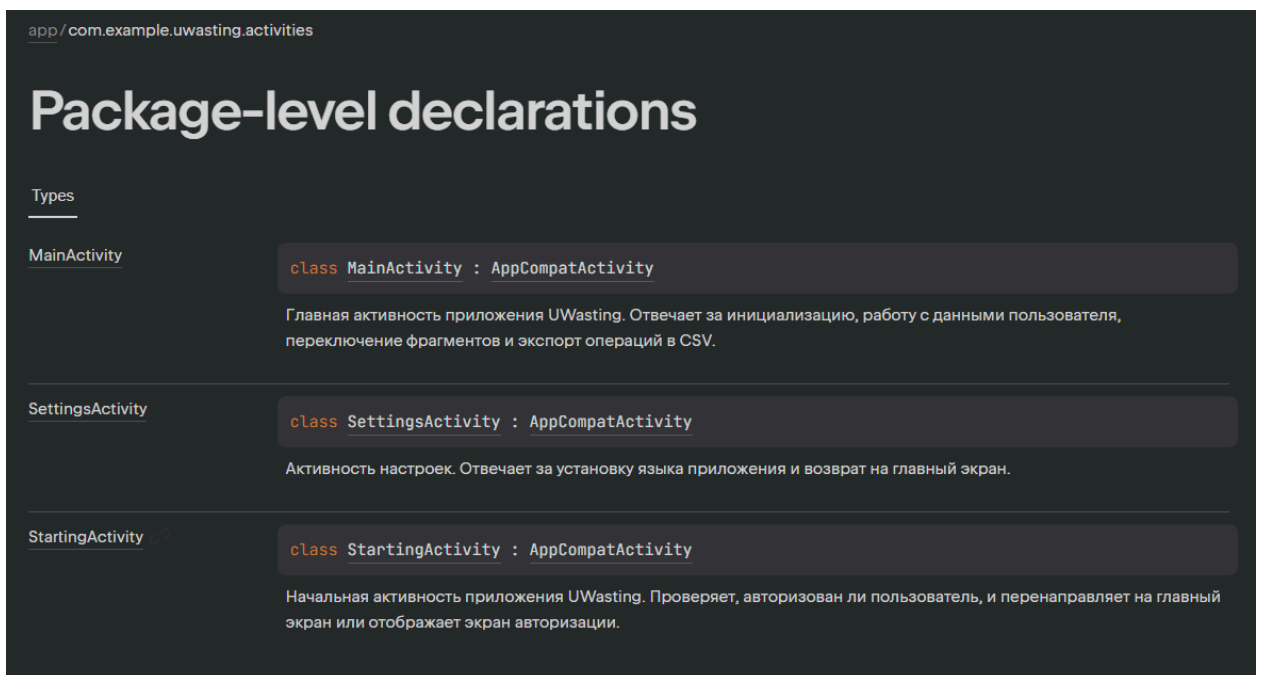


Рисунок 6 – Документация разработчика

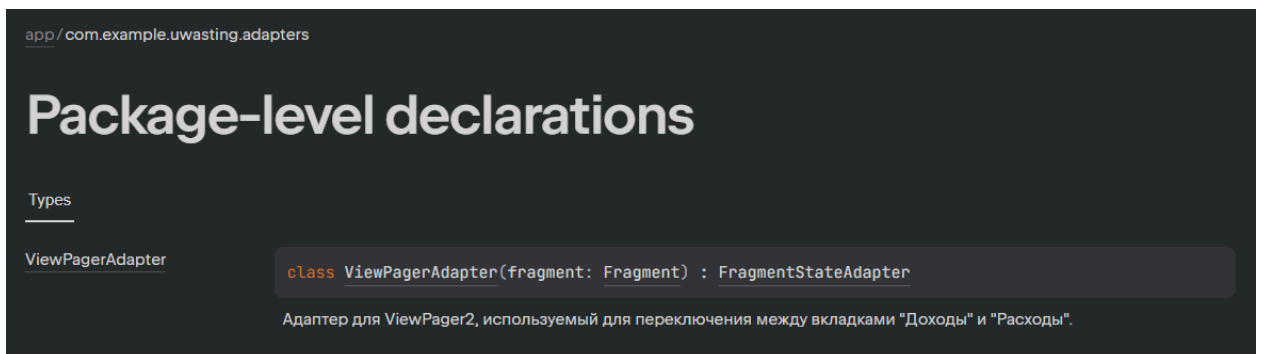


Рисунок 7 – Документация разработчика

app/com.example.uwasting.data	
<h1>Package-level declarations</h1>	
<u>Types</u>	
<u>Categories</u>	<pre>class Categories</pre> <p>Класс, содержащий predefined категории доходов и расходов, используемые в приложении UWasting для визуального и логического разделения операций.</p>
<u>Category</u>	<pre>class Category(val srcImage: Int, val name: String, val color: Int)</pre> <p>Модель категории операции (дохода или расхода).</p>
<u>CategoryRecyclerView</u>	<pre>class CategoryRecyclerView(data: ArrayList<Triple<Category, Int, Int>>, itemClickListener: OnItemClickListener, mainActivity: MainActivity) : RecyclerView.Adapter<CategoryRecyclerView.CategoryViewHolder></pre> <p>Адаптер для отображения списка категорий с информацией об операциях.</p>
<u>Constants</u>	<pre>class Constants</pre> <p>Класс, содержащий глобальные константы, используемые в приложении UWasting. Константы сгруппированы в объект-компаньон companion object для удобного доступа.</p>

Рисунок 8 – Документация разработчика




<u>LineReg</u> 	<pre>class LineReg(expenses: ArrayList<Operation>)</pre> <p>Класс, реализующий линейную регрессию для анализа финансовых расходов пользователя.</p>
<u>OnBackButtonListener</u> 	<pre>interface OnBackButtonListener</pre> <p>Интерфейс для обработки пользовательских действий при нажатии кнопки "Назад" внутри фрагментов.</p>
<u>OnItemClickListener</u> 	<pre>interface OnItemClickListener</pre> <p>Интерфейс обработчика нажатий на элементы списка категорий.</p>
<u>OnOperationClickListener</u>	<pre>interface OnOperationClickListener</pre> <p>Интерфейс обработчика нажатий на элементы списка операций.</p>
<u>Operation</u>	<pre>class Operation(var amount: Int, var category: String, var date: LocalDate, var id: Int)</pre> <p>Модель финансовой операции (дохода или расхода), используемая в приложении UWasting.</p>
<u>OperationsList</u>	<pre>class OperationsList(var item: ArrayList<Operation>)</pre> <p>Класс-обёртка над списком операций пользователя. Предоставляет методы фильтрации, агрегации и анализа данных о расходах и доходах.</p>

Рисунок 9 – Документация разработчика




OperationsRecyclerView 	<pre>class OperationsRecyclerView(data: ArrayList<Triple<LocalDate, Category, Int>>, onOperationClickListener: OnOperationClickListener, mainActivity: MainActivity) : RecyclerView.Adapter<OperationsRecyclerView.OperationViewHolder></pre> <p>Адаптер для отображения списка операций в RecyclerView. Используется на главном экране приложения для визуализации истории расходов и доходов.</p>
SelectingCategoryRecyclerView 	<pre>class SelectingCategoryRecyclerView : RecyclerView.Adapter<SelectingCategoryRecyclerView.SelectingCategoryViewHolder></pre> <p>Адаптер для RecyclerView, используемый при выборе категории операции. Позволяет пользователю выбрать одну из доступных категорий с помощью радиокнопки.</p>
User 	<pre>class User</pre> <p>Модель пользователя приложения UWasting.</p>

Рисунок 10 – Документация разработчика

app / com.example.uwasting.data.remote	
<h1>Package-level declarations</h1>	
<u>Types</u>	
StatBureauApi	<pre>interface StatBureauApi</pre> <p>Интерфейс для взаимодействия с API сайта statbureau.org. Предоставляет доступ к данным по инфляции (ИПЦ) через HTTP-запросы.</p>
StatBureauData	<pre>class StatBureauData(var inflationRate: Float, var inflationRateRounded: Float, var inflationRateFormatted: String, var month: String, var monthFormatted: String, var country: Int)</pre> <p>Модель данных, получаемая с сайта statbureau.org, содержащая информацию об инфляции.</p>
UWastingApi	<pre>interface UWastingApi</pre> <p>Интерфейс взаимодействия с серверной частью приложения UWasting. Предоставляет методы для регистрации, авторизации, изменения данных пользователя и работы с операциями.</p>

Рисунок 11 – Документация разработчика

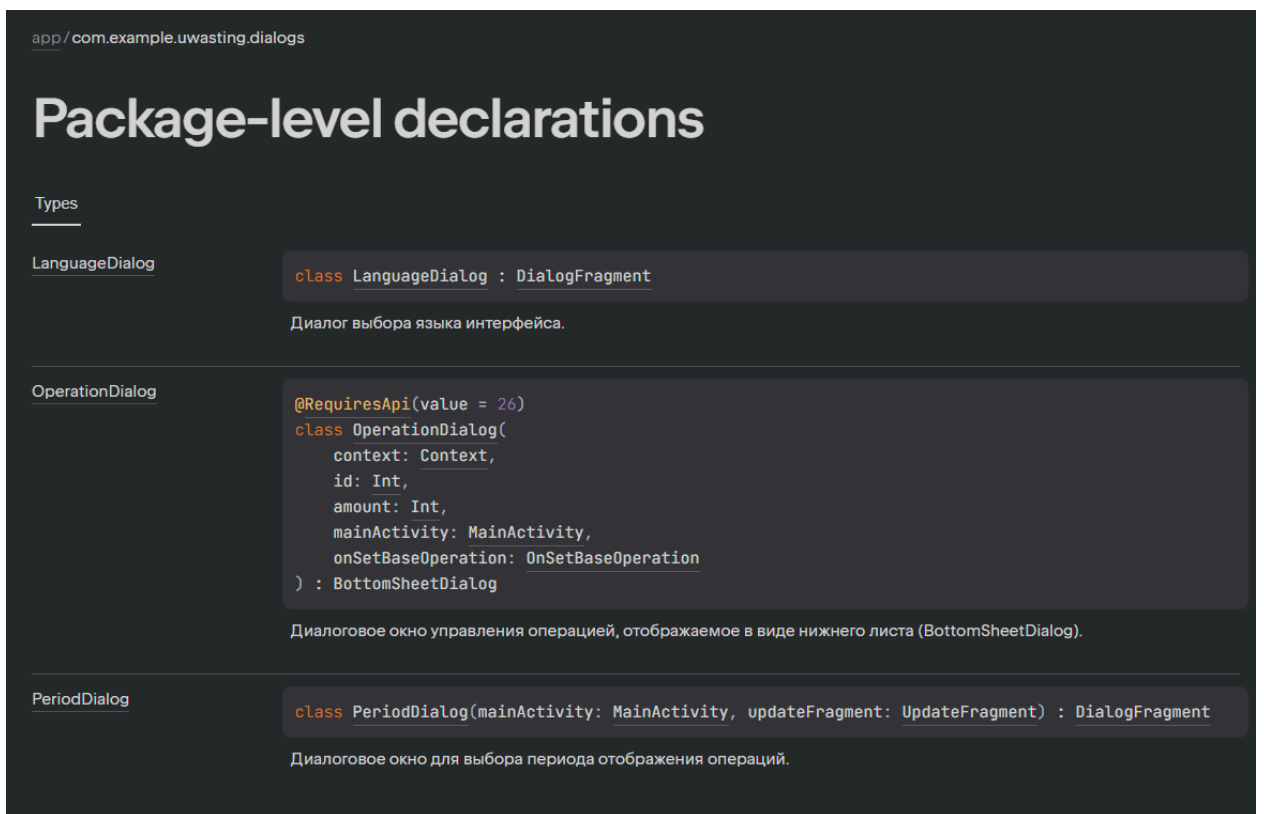


Рисунок 12 – Документация разработчика

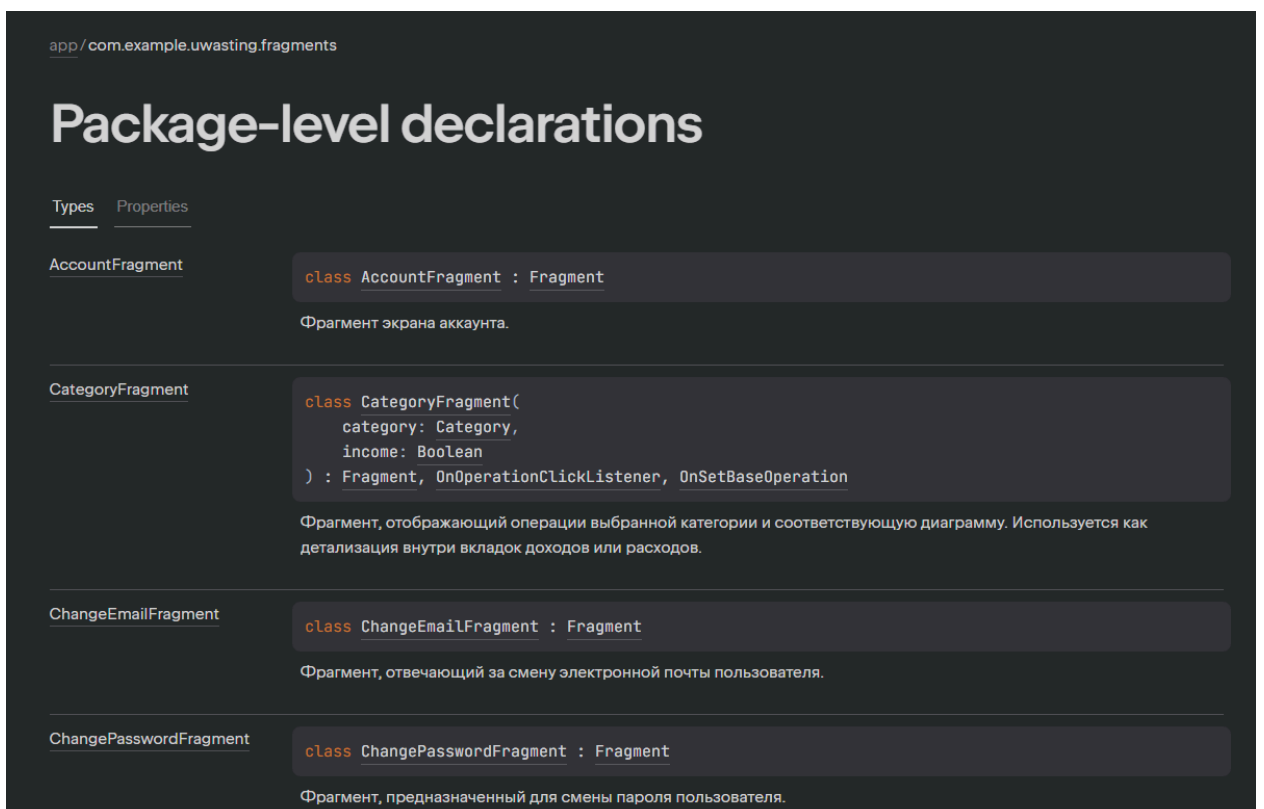


Рисунок 13 – Документация разработчика

EmailFragment	<pre>class EmailFragment : Fragment</pre> <p>Фрагмент ввода электронной почты при регистрации.</p>
ExpensesFragment	<pre>class ExpensesFragment : Fragment, OnItemClickListener, UpdateFragment</pre> <p>Фрагмент, отображающий статистику расходов пользователя.</p>
IncomesFragment	<pre>class IncomesFragment : Fragment, OnItemClickListener, UpdateFragment</pre> <p>Фрагмент, отображающий информацию о доходах пользователя.</p>
NameFragment	<pre>class NameFragment : Fragment</pre> <p>Фрагмент для ввода имени и фамилии пользователя при регистрации.</p>
NavigationHeaderFragment	<pre>class NavigationHeaderFragment : Fragment</pre> <p>Фрагмент, отображающий заголовок бокового меню (Navigation Drawer).</p>
NewExpenseFragment	<pre>class NewExpenseFragment : Fragment, SetCategory</pre> <p>Фрагмент для добавления новой расходной операции.</p>
NewIncomeFragment	<pre>class NewIncomeFragment : Fragment, SetCategory</pre> <p>Фрагмент для добавления новой доходной операции.</p>

Рисунок 14 – Документация разработчика

OnSetBaseOperation	<pre>interface OnSetBaseOperation</pre> <p>Интерфейс обратного вызова, вызываемого после установки новой базовой операции. Используется, чтобы обновить UI после изменения валюты (у.е.) или удаления операции.</p>
PasswordFragment	<pre>class PasswordFragment : Fragment</pre> <p>Фрагмент для завершения регистрации пользователя: ввод и подтверждение пароля.</p>
SelectCategoryFragment	<pre>class SelectCategoryFragment(setCategory: SetCategory, categoriesType: Int) : Fragment</pre> <p>Фрагмент для выбора категории дохода или расхода.</p>
SetCategory	<pre>interface SetCategory</pre>
SignInFragment	<pre>class SignInFragment : Fragment</pre> <p>Фрагмент входа в приложение (авторизация).</p>
StartFragment	<pre>class StartFragment : Fragment, OnBackButtonListener</pre> <p>Стартовый фрагмент приложения.</p>
TabFragment	<pre>class TabFragment : Fragment, OnBackButtonListener</pre> <p>Главный фрагмент после авторизации, отображающий две вкладки:</p>

Рисунок 15 – Документация разработчика


UpdateFragment	<code>interface UpdateFragment</code>
ValueFormatter	<code>class ValueFormatter(xValsDateLabel: ArrayList<LocalDate>) : ValueFormatter</code> Кастомный форматтер оси X для отображения дат на BarChart.
VerifyPasswordFragment 	<code>class VerifyPasswordFragment(mode: Int) : Fragment</code> Фрагмент, предназначенный для проверки текущего пароля пользователя перед выполнением чувствительных действий.

Рисунок 16 – Документация разработчика

app / com.example.uwasting.preferences

Package-level declarations

Types Properties

MyPreference

`class MyPreference(context: Context)`
 Класс-обёртка над `SharedPreferences`, используемый для хранения пользовательских настроек и данных.

Рисунок 17 – Документация разработчика

2.2 Документация пользователя

Описание приложения

Andrew edited this page 3 minutes ago · [1 revision](#)

UWasting — это мобильное Android-приложение для контроля за личными расходами и рационального потребления ресурсов. Приложение позволяет пользователю отслеживать траты, классифицировать их по категориям и анализировать динамику потребления через графики и таблицы. В связке с серверной частью обеспечивается синхронизация данных и хранение истории расходов.

Рисунок 18 – Документация пользователя

Установка и требования

Andrew edited this page 5 minutes ago · [1 revision](#)

Требования:

- ОС: Android 5.0 и выше
- Интернет: необходим для синхронизации
- Память: ~30MB на устройстве
- Разрешения:
 - Доступ в интернет
 - Доступ к внутреннему хранилищу (опционально для логов/резервных копий)

Установка:

- Скачайте .apk с [GitHub Releases](#) или соберите вручную из исходников через Android Studio.
- Установите APK и разрешите установку из неизвестных источников при необходимости.
- При первом запуске будет предложено ввести данные для подключения к серверу.

Рисунок 19 – Документация пользователя

Обзор интерфейса

Andrew edited this page 4 minutes ago · [1 revision](#)

Главный экран (MainActivity)

- Список всех добавленных трат.
- Кнопка "+" в нижнем правом углу — добавление новой траты.
- Верхняя панель: фильтры по дате и категориям, кнопка обновления данных, настройки.

Экран добавления траты (AddActivity)

Поля ввода:

- Категория (выбор из выпадающего списка)
- Описание
- Сумма
- Дата (по умолчанию текущая)
- Кнопка "Сохранить"

Экран статистики (StatsActivity)

- Гистограмма расходов по категориям.
- Фильтрация по временным диапазонам (день/неделя/месяц).
- Диаграммы, отображающие долю трат по категориям.

Настройки (SettingsActivity)

- Сервер: IP-адрес или URL API
- Токен авторизации
- Интервал синхронизации
- Уведомления

Рисунок 20 – Документация пользователя

Основные функции

Andrew edited this page 3 minutes ago · [1 revision](#)

1. Синхронизация с сервером

- Все расходы сохраняются на сервере, указанном в настройках.
- Используется API (описано в UWasting-Server).
- Данные синхронизируются при запуске приложения или вручную через кнопку "обновить".

2. Учёт трат

- Создание новых записей с категорией, датой, описанием и суммой.
- Поддержка категорий: еда, транспорт, развлечения, коммунальные, прочее.
- Локальное и серверное хранение данных.

3. Статистика и анализ

- Визуальные отчёты: столбчатые и круговые диаграммы.
- Распределение по категориям.
- Выбор периода анализа: текущий день, неделя, месяц.

4. Настройки

- Возможность вручную задать адрес API сервера.
- Тест соединения с сервером.
- Настройка автоматической синхронизации и уведомлений.
- Очистка локальных данных.

Рисунок 21 – Документация пользователя

Поддерживаемые категории расходов

Andrew edited this page 2 minutes ago · [1 revision](#)

(в коде определены в Categories.kt)

- Food
- Transport
- Entertainment
- Utilities
- Other

Рисунок 22 – Документация пользователя

Работа с сервером

Andrew edited this page 1 minute ago · [1 revision](#)

Приложение обращается к REST API, предоставляемому серверной частью (UWasting-Server):

- `/api/expenses` — получение/добавление/обновление трат
- `/api/categories` — получение списка категорий
- Авторизация реализована через API Token

Все запросы отправляются в формате JSON.

Рисунок 23 – Документация пользователя

ВЫВОД

В ходе выполнения практических работ №9-10 была организована разработка мобильного приложения UWasting, направленного на учёт расходов и анализ финансов. Командой успешно внедрена методология Scrum, что позволило гибко управлять задачами, оперативно корректировать требования и распределять нагрузку.

Для клиентской части использован стек технологий на базе Kotlin (AndroidX, Retrofit, RxJava), обеспечивающий стабильность и удобство разработки. Серверная часть реализована на C# (ASP.NET Core) с использованием PostgreSQL и Docker, что гарантирует масштабируемость и надёжность системы.

Документация разработчика и пользователя охватывает ключевые аспекты проекта: структуру пакетов, классы, интерфейсы, а также инструкции по установке и использованию приложения.