МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет физики, математики, информатики

Кафедра программного обеспечения и администрирования информационных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине

Технологии и методы программирования

*на тему: МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ УХОДА ЗА ДОМАШНИМИ РАСТЕНИЯМИ*

Обучающегося 2 курса

очной формы обучения

направления подготовки

10.03.01 Информационная безопасность

Направленность (профиль)

Безопасность компьютерных систем (в сфере техники и технологии)

ФИО

Руководитель:

доцент кафедры ПОиАИС

Макаров Константин Сергеевич

ассистент кафедры ПОиАИС

Овсянников Александр Владимирович

Допустить к защите:

/

« »

20 г

Курск, 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc200022024)

[1 ПОСТАНОВКА ЗАДАЧИ 10](#_Toc200022025)

[1.1 Формулировка условия задачи 10](#_Toc200022026)

[1.2 Анализ и исследование задачи 12](#_Toc200022027)

[1.2.1 Краткие теоретические сведения об объекте исследования 12](#_Toc200022028)

[1.2.2 Обзор и анализ возможных альтернатив 13](#_Toc200022029)

[1.2.3 Разработка требований к приложению 15](#_Toc200022030)

[1.2.4 Анализ инструментальных средств 16](#_Toc200022031)

[1.3 Методические ограничения 17](#_Toc200022032)

[1.3.1 Стандарты 17](#_Toc200022033)

[1.3.2 Программная совместимость 18](#_Toc200022034)

[1.3.3 Требования к составу и параметрам технических средств 18](#_Toc200022035)

[1.3.4 Входные данные 18](#_Toc200022036)

[1.3.5 Выходные данные 19](#_Toc200022037)

[1.3.6 Безопасность и секретность 19](#_Toc200022038)

[1.3.7 Мобильность 20](#_Toc200022039)

[2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ 20](#_Toc200022040)

[2.1 Разработка структуры данных 20](#_Toc200022041)

[2.2 Разработка модели поведения объекта 23](#_Toc200022042)

[2.3 Формализация расчетов 29](#_Toc200022043)

[2.3.1 Основные алгоритмы 29](#_Toc200022044)

[2.3.2 Алгоритмы, обеспечивающие функциональность приложения 33](#_Toc200022045)

[2.4 Разработка интерфейса приложения 35](#_Toc200022046)

[2.5 Описание структуры приложения 42](#_Toc200022047)

[ЗАКЛЮЧЕНИЕ 44](#_Toc200022048)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc200022049)

[ПРИЛОЖЕНИЕ А 49](#_Toc200022050)

[ИСХОДНЫЙ КОД ПРОГРАММЫ 49](#_Toc200022051)

# ВВЕДЕНИЕ

В последние годы уход за комнатными растениями стал одним из наиболее востребованных направлений в сфере хобби и озеленения жилых и офисных помещений. Регулярный контроль за состоянием растений, своевременный полив, внесение удобрений и учёт оптимальных условий выращивания требуют от владельца обширных знаний и систематичности в ведении записей. Традиционные методы — бумажные дневники, разрозненные справочники и постоянный поиск информации в Интернете — часто оказываются неудобными и трудоёмкими. В связи с широким распространением мобильных устройств возникает необходимость создания единого инструмента, который позволил бы централизовать справочную информацию о растениях и обеспечить персонализированный учёт «зелёных питомцев» непосредственно на смартфоне.

Объектом исследования в данной работе является мобильное приложение для платформы Android, реализующее функции учёта и ухода за домашними растениями. Предметом исследования являются методы и средства организации локальной базы данных, взаимодействия пользовательского интерфейса и алгоритмы обработки информации о растениях (название, фотография, описание, условия выращивания), а также функции добавления, просмотра и редактирования пользовательских записей с указанием даты посадки.

Цель работы — разработать и обосновать структуру, алгоритмы и интерфейс приложения на языке Java в Android Studio, обеспечивающие хранение и управление справочной информацией о растениях, а также персонализированный учёт «зелёных питомцев» посредством локальной базы данных Room. Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить предметную область мобильных приложений для учёта домашних растений и проанализировать существующие аналоги.
2. Разработать функциональные требования к приложению и обосновать выбор инструментальных средств (Java, Android SDK, Room, Glide).
3. Спроектировать структуру данных и модель поведения объектов (таблицы «Plant», «UserPlant» и их взаимосвязи).
4. Формализовать основные алгоритмы взаимодействия Activity и DAO-интерфейсов, а также алгоритмы загрузки и отображения изображений.
5. Создать прототип интерфейса приложения с учётом требований эргономики: экран списка растений (MainActivity), экран добавления нового растения (CreateActivity), экран просмотра деталей (DescriptionActivity) и экран редактирования записи (EditActivity).
6. Провести тестирование приложения, оценить устойчивость к ошибкам пользователя и корректность работы с датами посадки и фотографиями.

Практическая ценность работы заключается в создании прототипа мобильного клиента для учёта и ухода за домашними растениями, доступного офлайн и не требующего постоянного подключения к сети. Полученные результаты могут использоваться цветоводами-любителями для систематизации справочной информации и контроля за состоянием каждого растения, а также станут основой для дальнейших расширений: добавления напоминаний о поливе, интеграции с облачным хранилищем и расширения справочной базы.

Структура работы соответствует утверждённому плану и включает следующие разделы:

• Постановка задачи, в которой даётся формулировка проблемы, проводится анализ объекта исследования, рассматриваются альтернативные подходы, формулируются требования и методические ограничения.

• Проектирование и разработка приложения, включающее описание структуры данных (ER-диаграмма, модели Plant.java, UserPlant.java), модели поведения, алгоритмов взаимодействия (DAO, ViewModel) и пользовательского интерфейса (макеты экранов, адаптер PlantRecyclerAdapter.java).

• Анализ проекта и тестирование, где рассматриваются методы защиты от ошибок пользователя и результаты функциональных и стресс-тестов (добавление, отображение, редактирование, удаление записей).

• Заключение, подведение итогов проделанной работы и рекомендации по дальнейшему развитию приложения (добавление напоминаний о поливе, синхронизация с облаком, мультиязычность и расширение справочной базы).

• Список использованных источников и приложения, содержащие исходный код приложения и подробное описание основных алгоритмов.

Таким образом, выполнение данной работы позволит получить целостное решение задачи создания мобильного приложения для ухода за домашними растениями с использованием современных технологий локального хранения данных и разработки под Android.

Аннотация: в данной пояснительной записке изложены этапы разработки мобильного приложения для ухода за домашними растениями на платформе Android, реализованного на языке Java с использованием локальной базы данных Room. Приложение обеспечивает функции хранения справочной информации о растениях, добавления пользовательских записей с указанием даты посадки, просмотра детальной информации и редактирования записей.

Ключевые слова: мобильное приложение, уход за растениями, Android, Java, Room.

Annotation: This explanatory note describes the development of a mobile application for plant care implemented as an Android app written in Java, using the Room library for local data storage. The application provides functions for storing reference information about plants, adding user-specific plant entries with a planting date, viewing detailed information, and editing records.

Keywords: mobile application, plant care, Android, Java, Room.

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Формулировка условия задачи

В мобильном приложении для ухода за домашними растениями должны быть реализованы следующие функции:

* хранение справочной информации о растениях: название, фотография, описание, оптимальные условия выращивания;
* добавление нового растения в личный каталог пользователя с указанием даты посадки, загрузкой фотографии и возможностью задать заметки о текущем состоянии;
* редактирование и удаление записей о растениях, включая изменение даты посадки, обновление фотографии и примечаний;
* отображение списка всех добавленных пользователем растений в виде прокручиваемых карточек (RecyclerView), где каждая карточка содержит название, миниатюру фотографии и дату посадки;
* просмотр детальной информации о выбранном растении: расширенное описание, условия полива и освещения, календарь ухода (при необходимости);
* вывод напоминаний о запланированных агротехнических мероприятиях (полив, подкормка, пересадка) на основе даты посадки и заданных интервалов;
* поиск и фильтрация растений в личном каталоге по названию, дате посадки или растущим условиям;
* просмотр истории действий над каждой записью (дата создания, редактирование, удаления) с указанием времени операции.

При запуске приложения пользователь получает доступ к главному экрану со списком добавленных растений. В нижней части экрана располагается кнопка «Добавить растение», при нажатии на которую открывается форма CreateActivity с полями:

• «Название» (текстовое поле);

• «Фотография» (кнопка загрузки или съёмки фото);

• «Описание» (многострочное текстовое поле);

• «Условия выращивания» (выбор из справочных шаблонов или ввод вручную);

• «Дата посадки» (календарь для выбора даты);

• «Заметки» (необязательное текстовое поле).

После сохранения новой записи пользователь возвращается на главный экран, где обновлённый список растений отображается с учётом добавленного элемента.

На экране DescriptionActivity при выборе карточки растения выводится:

• полная информация о растении (фото в большом размере, название, описание, условия выращивания);

• дата посадки и автоматически рассчитанный возраст растения (в днях);

• кнопки «Редактировать» (переход в EditActivity) и «Удалить» (удаление записи с подтверждением).

Экран EditActivity повторяет поля CreateActivity, но предварительно заполнен текущими данными выбранного растения. После сохранения изменений пользователь возвращается к экрану с подробностями растения.

Раздел «История действий» доступен из бокового меню или навигационной панели и содержит хронологический список операций (добавление, редактирование, удаление) с указанием даты и времени, а также кратким описанием действия (например, «Добавлено: Фикус Бенджамина – 12.04.2025», «Отредактировано: Орхидея – обновлено описание»).

Интерфейс приложения должен обеспечить интуитивно понятный доступ ко всем функциям:

• главный экран с прокручиваемым списком растений и кнопкой «Добавить»;

• экран просмотра деталей с понятными пиктограммами для редактирования и удаления;

• форма добавления/редактирования с отдельными секциями для фотографии, описания и условий выращивания;

• раздел «История действий» с возможностью фильтрации по типу операции и датам.

Необходимо также реализовать:

• проверку корректности ввода данных (например, обязательное поле «Название», дата посадки не в будущем);

• обработку ошибок при работе с базой данных (отказ сохранения, ошибка чтения) и отображение информативных сообщений пользователю;

• адаптивность интерфейса под различные размеры экранов (телефоны и планшеты) и поддержку Android 8.0 (API 24) и выше;

• эффективную загрузку и кеширование изображений (с использованием библиотеки Glide или аналога), чтобы уменьшить задержки при отображении фотографий растений.

Все данные о растениях и истории действий хранятся локально в базе данных Room. При необходимости в будущих версиях приложения предусмотрена возможность синхронизации с облачным хранилищем, однако в рамках текущей задачи достаточно обеспечить полноценную офлайн-работу и сохранность данных при закрытии приложения.

## 1.2 Анализ и исследование задачи

### 1.2.1 Краткие теоретические сведения об объекте исследования

Мобильное приложение для ухода за домашними растениями представляет собой программный продукт справочно-учётного типа, работающий на платформе Android. Основная его задача — хранение и обработка информации о растениях (название, фотография, описание, условия выращивания) и регистрация пользовательских записей с указанием даты посадки и прочих параметров. В качестве локального хранилища данных используется встроенная реляционная СУБД Room, позволяющая работать с объектно-реляционными моделями (классы Entity, DAO-интерфейсы, ViewModel и LiveData). Пользовательский интерфейс реализован средствами Android UI (Activity, RecyclerView, CardView, фрагменты), обеспечивающими отображение списка растений, форм ввода/редактирования и экран детального просмотра. Логика напоминаний (полив, подкормка) может быть реализована с использованием WorkManager или AlarmManager для планирования периодических уведомлений на основе сохранённой даты посадки и заданных интервалов.

### 1.2.2 Обзор и анализ возможных альтернатив

В качестве наиболее близких по функциональности альтернатив рассматриваются следующие приложения:

• PlantNet – справочник и идентификатор растений на основе анализа фотографий. Позволяет распознавать растения, но не предоставляет полного учёта пользовательских посадок и напоминаний.

• Gardenize – приложение для ведения садового дневника, позволяющее сохранять фотографии и заметки о растениях. Имеет облачную синхронизацию, но требует постоянного соединения с интернетом и не всегда оптимизировано для работы с комнатными растениями.

• Florish (Anya) – приложение с функциями напоминаний о поливе и советами по уходу. Ограничения: платная подписка на расширенный контент, нет возможности самостоятельно добавлять справочные описания растений и хранить подробную информацию локально без интернета.

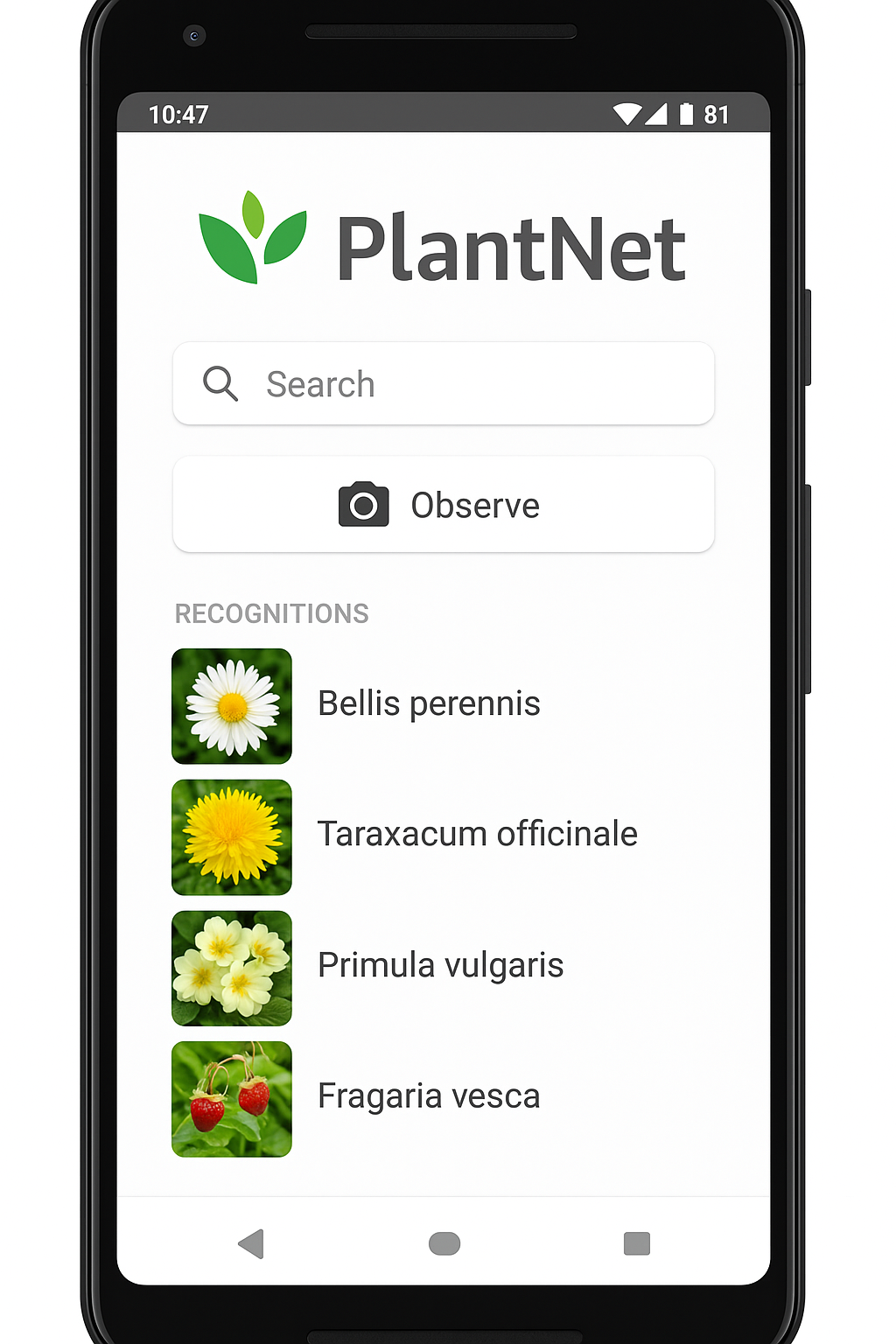


Рисунок 1 – альтернативный сервис «PlantNet»

Недостатки перечисленных решений:

• Зависимость от облака – большинство функций работает только при подключении к интернету, что неудобно при ограниченном трафике или отсутствии связи.

• Ограниченный набор справочной информации – приложения фокусируются либо на распознавании видов, либо на базе платного контента, без открытой структуры для пополнения локальной БД.

• Отсутствие гибкой настройки пользовательских записей – нельзя задать собственные параметры (дата посадки, заметки, индивидуальные интервалы полива) в полном объёме.

Ключевое отличие нашего решения заключается в том, что:

• используется локальная СУБД Room, позволяющая хранить справочную и пользовательскую информацию офлайн;

• пользователь может самостоятельно добавлять новые виды растений с подробным описанием и фотографией;

• реализована функция напоминаний о поливе и подкормке на базе заданных интервалов и даты посадки без зависимости от интернета;

• интерфейс адаптирован для различных размеров экранов и версий Android (8.0+), что расширяет охват аудитории.

### 1.2.3 Разработка требований к приложению

Функциональные требования:

• Хранение справочной информации о растениях: название, фотография, описание, оптимальные условия выращивания (освещённость, влажность, температура).

• Добавление нового растения пользователем с указанием следующих данных: название, фотография, дата посадки, описание текущего состояния (примечания).

• Редактирование и удаление записей растения: изменение даты посадки, обновление фотографии, корректировка описания и примечаний.

• Отображение списка всех пользовательских растений в виде прокручиваемых карточек (RecyclerView), содержащих название, миниатюру фотографии и дату посадки.

• Просмотр детальной информации о выбранном растении: расширенное описание, условия ухода, возраст растения (в днях), календарь запланированных напоминаний.

• Планирование и отображение напоминаний о поливе, подкормке и пересадке на основе даты посадки и заданных интервалов.

• Поиск и фильтрация списка растений по названию, дате посадки и условиям выращивания.

• Отображение истории действий над каждой записью (дата добавления, редактирования, удаления) с указанием времени операции.

Нефункциональные требования:

• Надёжность и устойчивость – корректная работа с базой данных при одновременном доступе, проверка ввода обязательных полей (название, дата посадки).

• Юзабилити – интуитивно понятный интерфейс, минимальное количество действий для добавления/редактирования записи, визуальные подсказки по состоянию растения.

• Производительность – быстрое отображение списка (до 200 мс при загрузке до 100 записей), эффективное кеширование изображений для плавного скроллинга.

• Мобильность – поддержка Android 8.0 (API 24) и выше, адаптивный дизайн для смартфонов и планшетов с учётом разных разрешений экранов.

• Безопасность – хранение всех данных локально, отсутствие критичных разрешений (только доступ к файлам для загрузки фото и календарю для установки уведомлений).

• Масштабируемость – возможность в будущем расширить приложение функционалом синхронизации с облачным хранилищем или добавления аналитики (например, сбор статистики по времени жизни растений).

### 1.2.4 Анализ инструментальных средств

Среда разработки: Android Studio (Arctic Fox или новее), JDK 11, Gradle.

Язык программирования: Java 11 (официально поддерживается Android SDK).

Локальная база данных: Room Library (Google) для работы с SQLite через аннотации (@Entity, @Dao, @Database), LiveData и ViewModel для реактивного обновления UI при изменении данных.

UI-компоненты:

• RecyclerView + CardView – отображение списка растений;

• ConstraintLayout – гибкое расположение элементов на экранах;

• Material Components (AndroidX) – стилизация кнопок, полей ввода, диалоговых окон;

• DatePickerDialog – выбор даты посадки;

• WorkManager или AlarmManager – планирование локальных уведомлений о поливе и подкормке.

Загрузка и обработка изображений:

• Glide (или Picasso) – загрузка, масштабирование и кеширование фотографий растений в списке и детальном просмотре.

Тестирование:

• JUnit 4/5 – написание модульных тестов для DAO и ViewModel;

• Espresso – UI-тесты основных сценариев: добавление, редактирование, удаление записи, отображение списка;

• Android Profiler – анализ производительности при работе с изображениями и базой данных.

Системы контроля версий:

• Git + GitHub – управление исходным кодом, ветвления для новых функций, ревью изменений.

Таким образом, выбранный стек технологий и инструментальных средств обеспечивает полное покрытие требований по разработке, хранению данных, отображению интерфейса и тестированию мобильного приложения для ухода за домашними растениями.

## 1.3 Методические ограничения

### 

### 1.3.1 Стандарты

Разработка программной документации и кода приложения должна вестись в соответствии с требованиями ЕСПД и стандартами ГОСТ 19.701–90 «Система разработки и постановки на производство программ для ЭВМ. Термины и определения» и ГОСТ 2.304–88 «Правила выполнения чертежей схем».

### 1.3.2 Программная совместимость

Приложение должно корректно работать под управлением операционных систем Android версии 8.0 (API 24) и выше.

### 1.3.3 Требования к составу и параметрам технических средств

Для работы приложения желательно иметь мобильное устройство со следующими минимальными техническими характеристиками:

* процессор архитектуры ARM;
* тактовая частота не менее 1,2 ГГц;
* встроенная память: 16 ГБ или более;
* экран: минимальное разрешение 1280x720 пикселей;
* интернет-соединение: обязательно наличие доступа к мобильному интернету 3G/4G или WiFi.

### 1.3.4 Входные данные

Входными данными приложения для ухода за домашними растениями являются:

•  данные о растении, вводимые пользователем при добавлении/редактировании записи:

– «Название» (строка);

– «Фотография» (файл изображения, загружаемый из галереи или снятый камерой);

– «Описание» (многострочная строка с текстом);

– «Условия выращивания» (строка)

– «Дата посадки» (дата, выбранная в календаре);

– «Заметки» (необязательное многострочное текстовое поле).

•  действия пользователя над записями:

– команда «Добавить растение» (нажатие на кнопку, переход в CreateActivity);

– команда «Редактировать запись» (передача ID выбранного растения в EditActivity);

– команда «Удалить запись» (передача ID и подтверждение удаления).

### 1.3.5 Выходные данные

Выходные данные отображаются в графическом интерфейсе и включают:

•  список добавленных растений в виде карточек (RecyclerView), каждая карточка содержит: название, миниатюру фотографии, дату посадки;

•  детальную информацию о выбранном растении (в DescriptionActivity): полноразмерная фотография, название, описание, условия выращивания, возраст растения (вычисляется как разность текущей даты и даты посадки), примечания;

•  список запланированных и сработавших напоминаний (полив, подкормка, пересадка) с указанием даты и времени;

### 1.3.6 Безопасность и секретность

•  Все данные о растениях и напоминаниях хранятся локально в базе данных Room и не передаются в сеть, что исключает утечку личной информации.

•  Файлы изображений (фотографии растений) сохраняются в защищённом внутреннем хранилище приложения; доступ к ним извне невозможен без спецразрешений.

•  При возникновении ошибок доступа к базе данных или файловой системе приложение выводит информативное сообщение, но без раскрытия технических деталей (например, «Не удалось сохранить изображение»).

### 1.3.7 Мобильность

1. Собрать APK-файл в Android Studio (Build → Build Bundle(s)/APK(s) → Build APK(s)).
2. Скопировать готовый APK-файл на мобильное устройство (через USB, облачный диск или ADB).
3. Установить приложение (при необходимости разрешить установку из неизвестных источников).
4. Запустить приложение, нажав на его иконку в списке установленных программ на устройстве.
5. При обновлении версии достаточно повторить шаги 1–4, после чего локальные данные (база Room) сохранятся и будут доступны в новой сборке.

# 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

## 2.1 Разработка структуры данных

Для приложения «Уход за домашними растениями» предлагается следующая основная структура классов (DTO/Entity) и вспомогательных моделей для локального хранения и отображения данных в интерфейсе:

•  Справочная запись о растении (PlantDTO)

public class PlantDTO {

    private int id;                // автогенерируемый первичный ключ (Room @PrimaryKey autoGenerate = true)

    private String name;           // название растения

    private String description;    // описание (краткое/полное)

    private String conditions;     // условия выращивания (свет, температура, влажность)

    private String photoUri;       // URI фотографии (например, content://… или file://…)

}

Перед сохранением в локальную базу (Room) объект сериализуется автоматически через аннотации (@Entity, @ColumnInfo). Используется для формирования справочной базы, если приложение предусматривает предзаполненный каталог.

•  Пользовательская запись о растении (UserPlantDTO)

public class UserPlantDTO {

private int id; // автогенерируемый первичный ключ

private int plantRefId; // внешний ключ на PlantDTO (или 0, если растение добавлено «с нуля»)

private String customName; // пользовательское название (если отличается от справочного)

private String photoUri; // URI загруженной/сделанной пользователем фотографии

private long plantingDate; // дата посадки в миллисекундах (System.currentTimeMillis())

private String notes; // заметки пользователя (Entry: «точка размножения», «проблемы с листьями» и пр.)

}

При создании новой записи пользователь выбирает справочное растение (PlantDTO) или вводит всё вручную, после чего формируется объект UserPlantDTO и сохраняется в таблицу user\_plants.

•  Напоминание (Reminder)

public class Reminder {

private int id; // первичный ключ

private int userPlantId; // внешний ключ – ссылка на UserPlantDTO

private long remindTime; // время срабатывания (Unix-время в ms)

private String type; // тип напоминания: "WATER", "FERTILIZE", "REPOT"

private boolean isDone; // флаг: сработало ли (true/false)

}

Объекты Reminder создаются при добавлении или редактировании UserPlantDTO. Например, при добавлении растения со сроками полива каждые N дней создаются несколько записей в таблице напоминаний. Для планирования локальных уведомлений используется WorkManager или AlarmManager.

•  Запись истории действий (HistoryEntry)

public class HistoryEntry {

private int id; // первичный ключ

private int userPlantId; // внешний ключ – ссылка на UserPlantDTO

private long timestamp; // время операции (ms)

private String action; // тип действия: "ADD", "EDIT", "DELETE", "WATERED", "FERTILIZED"

private String description; // текстовое описание (например, "Полив: Орхидея — 05.05.2025")

}

Каждая операция над UserPlantDTO (создание, изменение, удаление) и факт выполнения напоминания (полив, подкормка) добавляет запись в history\_entries. Эти данные отображаются в разделе «История действий».

•  Пользовательский профиль (UserProfile)

public class UserProfile {

private String uid; // UUID или другой идентификатор пользователя

private String username; // имя пользователя (для титула/шапки приложения)

private String email; // email пользователя (если требуется регистрация)

}

Если функционал предполагает учетную запись, данные UserProfile загружаются при запуске приложения (например, из локального SharedPreferences или удаленного сервера) и кэшируются в памяти.

•  Параметры поиска/фильтрации (FilterParams)

public class FilterParams {

    private String queryText;         // текст поиска по названию

    private long dateFrom;            // начало диапазона даты посадки (ms)

    private long dateTo;              // конец диапазона даты посадки (ms)

    private String conditionsFilter;  // фильтр по условиям выращивания (например, "тень", "яркий свет")

}

Объект FilterParams формируется на основе ввода пользователя в поле поиска или диалоговом окне фильтрации и передается в ViewModel/DAO для получения отфильтрованного списка UserPlantDTO.

## 2.2 Разработка модели поведения объекта

Модель поведения системы формализуется как последовательность состояний и переходов при взаимодействии пользователя, локальных компонентов приложения (Activity, ViewModel, Repository) и базы данных Room.

1. Инициализация приложения

При запуске MainActivity проверяется наличие авторизованной сессии (по UserProfile в SharedPreferences). Если профиль существует, сразу инициализируется PlantRepository или экземпляр RoomDatabase для получения DAO. Затем выполняется запрос к userPlantDao методом getAll(), возвращающему LiveData списка объектов UserPlantDTO; на этот LiveData подписывается адаптер RecyclerView, чтобы автоматически обновлять список при изменении данных. Если таблица справочной информации PlantDTO пуста, в методе onCreate проверяется, нужно ли заполнить её предустановленными данными из ресурсов приложения.

1. Отображение списка растений

В MainActivity в методе onCreate настраивается RecyclerView с адаптером PlantRecyclerAdapter. Когда LiveData<List> изменяется, observer вызывает adapter.submitList() или notifyDataSetChanged(), и интерфейс обновляется. Кнопка «Добавить растение» (FloatingActionButton) создаёт Intent для перехода к CreateActivity с кодом REQUEST\_ADD\_PLANT.

1. Добавление нового растения

В CreateActivity в onCreate инициализируются поля ввода: текстовое поле для названия, текстовое поле для описания, текстовое поле для условий выращивания, кнопка выбора фотографии, текстовое поле для отображения даты посадки, кнопка вызова DatePickerDialog и текстовое поле для заметок.

При нажатии на кнопку «Выбрать фото» открывается Intent для выбора изображения из галереи или съёмки камерой; результат сохраняется в переменной photoUri. При нажатии на кнопку «Выбрать дату» открывается DatePickerDialog, и выбранная пользователем дата преобразуется в целое значение plantingDate типа long (System.currentTimeMillis()).

После заполнения обязательных полей (название и дата посадки) пользователь нажимает «Сохранить»: сначала проверяется, что название не пустое, иначе разблокируется ошибка в текстовом поле. Затем создаётся объект UserPlantDTO, где поле id устанавливается в 0 (чтобы Room автоматически сгенерировал первичный ключ), поле plantRefId устанавливается в 0 (если растение добавлено вручную, без справочной записи), поле customName получает введённое пользователем название, поле photoUri получает строку URI фотографии, поле plantingDate получает сохранённое значение даты посадки, а поле notes получает пользовательский текст заметок. Этот объект передаётся в userPlantDao.insert(newPlant), и после вставки возвращается с индексом, присвоенным Room.

Далее для каждого типа напоминания (например, полив, подкормка, пересадка) вычисляется время следующего срабатывания. Допустим, для напоминания о поливе вычисляется значение nextWaterTime = calculateNextTime(plantingDate, WATER\_INTERVAL). Создаётся объект Reminder, где id равен 0 (чтобы Room сгенерировал ключ), поле userPlantId равно идентификатору вновь вставленного UserPlantDTO, поле remindTime установлено в значение nextWaterTime, поле type задано строкой «WATER», а поле isDone установлено в false. Этот объект сохраняется через reminderDao.insert(waterReminder), и для nextWaterTime планируется выполнение задачи WorkManager или AlarmManager.

После этого создаётся запись истории: объект HistoryEntry с id = 0 (Room сгенерирует ключ), поле userPlantId = полученный идентификатор растения, поле timestamp = System.currentTimeMillis(), поле action = «ADD», поле description = «Добавлено растение: « + введённое название. Эту запись сохраняют через historyDao.insert(entry). Затем вызывается setResult(RESULT\_OK) и finish(), возвращая пользователя в MainActivity, где список автоматически обновляется.

1. Просмотр детальной информации

В MainActivity при нажатии на элемент списка через интерфейс OnPlantClickListener создаётся Intent к DescriptionActivity, и в него кладётся целочисленное значение userPlantId, равное методу getId() у объекта UserPlantDTO.

В DescriptionActivity в onCreate извлекается userPlantId через getIntent().getIntExtra(“USER\_PLANT\_ID”, -1). Затем загружается объект UserPlantDTO plant, полученный методом userPlantDao.getById(userPlantId) или через LiveData с observe. Элементы интерфейса заполняются данными: ImageView устанавливает изображение по строке URI через метод setImageURI(Uri.parse(plant.getPhotoUri())), TextView устанавливает текст названия через plant.getCustomName(), TextView устанавливает условия выращивания через plant.getConditions(), TextView устанавливает описание растения через plant.getDescription(), TextView отображает дату посадки через форматирование plant.getPlantingDate(), а возраст рассчитывается как возраст в днях: возраст = (System.currentTimeMillis() минус plant.getPlantingDate()) делить на (1000 \* 60 \* 60 \* 24). Далее TextView для возраста получает значение в формате «N дн.», а TextView для заметок устанавливается plant.getNotes(). После заполнения основных полей запрашивается список напоминаний reminderDao.getByUserPlantId(userPlantId) и отображается в соответствующем RecyclerView или ListView.

1. Редактирование записи о растении

В DescriptionActivity кнопка «Редактировать» создаёт Intent к EditActivity и кладёт туда прежний userPlantId. В EditActivity в onCreate выполняется запрос userPlantDao.getById(userPlantId) (или через LiveData) для загрузки объекта UserPlantDTO plant. Поля формы редактирования: текстовое поле для названия, текстовое поле для описания, текстовое поле для условий, ImageView для фотографии, текстовое поле для даты посадки и текстовое поле для заметок заполняются текущими значениями: метод setText(plant.getCustomName()), setText(plant.getDescription()), setText(plant.getConditions()), установка изображения через setImageURI(Uri.parse(plant.getPhotoUri())), установка текста даты через formatDate(plant.getPlantingDate()), установка текста заметок через plant.getNotes().

После того как пользователь внесёт изменения и нажмёт «Сохранить», сначала проверяется, что название не пустое. Затем вызываются методы plant.setCustomName(newName), plant.setPhotoUri(newPhotoUri), plant.setPlantingDate(newPlantingDate), plant.setNotes(newNotes), и после этого plant передаётся в userPlantDao.update(plant). Далее удаляются старые напоминания методом reminderDao.deleteByUserPlantId(userPlantId), после чего создаются новые напоминания: сначала вычисляется новое remindTime (например, calculateNextTime на основании изменённой даты посадки), затем создаётся объект Reminder с id = 0, userPlantId = plant.getId(), remindTime = вычисленное время, type равен соответствующему типу, isDone = false, и сохраняется через reminderDao.insert(nextReminder). После этого планируется задача WorkManager через scheduleWork(nextReminder). Затем создаётся запись истории: объект HistoryEntry с полями id = 0, userPlantId = идентификатор, timestamp = System.currentTimeMillis(), action = «EDIT», description = «Отредактировано растение: « + новый текст названия, и сохраняется через historyDao.insert(entry). Наконец, вызывается setResult(RESULT\_OK) и finish(), возвращая пользователя в DescriptionActivity или MainActivity, где данные автоматически обновляются.

1. Удаление записи о растении

В DescriptionActivity при нажатии на кнопку «Удалить» показывается диалог подтверждения (AlertDialog). Если пользователь подтверждает удаление, сначала вызывается reminderDao.deleteByUserPlantId(userPlantId) для удаления всех связанных напоминаний, затем userPlantDao.delete(plant) удаляет сам объект UserPlantDTO. После этого создаётся запись истории: HistoryEntry entry с id = 0, userPlantId = идентификатор, timestamp = System.currentTimeMillis(), action = «DELETE», description = «Удалено растение: « + plant.getCustomName(), и сохраняется через historyDao.insert(entry). Наконец, вызывается finish(), возвращая пользователя в MainActivity.

1. Обработка напоминаний и обновление UI

Когда наступает время remindTime, запланированная задача WorkManager или AlarmManager выдаёт локальное уведомление с текстом «Пора полить: [название растения]». При взаимодействии пользователя с уведомлением (например, нажатие на действие «Отметить как выполнено» в NotificationAction) приложение получает объект Reminder, устанавливает его поле isDone в true и сохраняет через reminderDao.update(reminder). Затем создаётся новая запись истории: HistoryEntry entry с id = 0, userPlantId = идентификатор, timestamp = System.currentTimeMillis(), action = «WATERED», description = «Полив: « + plant.getCustomName(), и сохраняется через historyDao.insert(entry). Далее вычисляется время следующего напоминания: nextTime = calculateNextTime(currentTime, WATER\_INTERVAL), создаётся новый объект Reminder с id = 0, userPlantId = идентификатор, remindTime = nextTime, type = «WATER», isDone = false, сохраняется через reminderDao.insert(nextReminder), и планируется следующая задача по scheduleWork(nextReminder). В MainActivity или в отдельном RemindersActivity отображается список предстоящих напоминаний, получаемый через reminderDao.getPending(). При изменении данных LiveData автоматически обновляет RecyclerView.

1. Поиск и фильтрация списка растений

В MainActivity присутствует компонент SearchView или EditText, а также кнопка «Фильтр», вызывающая DialogFragment с полями, соответствующими объекту FilterParams: текстовый поиск по названию (queryText), диапазон дат посадки (dateFrom и dateTo), фильтр по условиям выращивания (conditionsFilter). После того как пользователь нажмёт кнопку «Применить», приложение вызывает метод userPlantDao.getFiltered(”%” + queryText + “%”, dateFrom, dateTo, conditionsFilter), возвращающий LiveData<List>, на который подписывается адаптер методом submitList, что приводит к обновлению отображаемого списка.

1. Просмотр истории действий

В HistoryActivity в методе onCreate загружается LiveData<List> через historyDao.getAll(). Затем RecyclerView настраивается адаптером HistoryAdapter. Каждый элемент списка отображает отформатированное время из поля timestamp (например, «dd.MM.yyyy HH:mm»), строку action (тип операции: ADD, EDIT, WATERED, DELETE) и текст из поля description (описание операции). При необходимости реализована фильтрация по типу действия или по дате с помощью дополнительных запросов DAO, принимающих параметры даты или списка типов.

Таким образом, описанная модель поведения обеспечивает предсказуемое взаимодействие между пользователем, компонентами приложения и локальной базой данных Room, гарантирует своевременное появление напоминаний, корректное обновление интерфейса при изменениях данных и хранение полного журнала операций.

## 2.3 Формализация расчетов

### 2.3.1 Основные алгоритмы

1. Добавление нового растения

При нажатии кнопки «Сохранить» в CreateActivity формируется объект UserPlantDTO с полями id = 0, plantRefId = справочный идентификатор или 0, customName = введённое название, photoUri = строка URI выбранного фото, plantingDate = значение даты в миллисекундах, notes = текст заметок. Вызывается метод userPlantDao.insert(newPlant) для сохранения в таблицу user\_plants, который возвращает автоматически сгенерированный идентификатор newPlantId.

Далее для каждого типа напоминания (полив, подкормка, пересадка) вычисляется время первого напоминания: nextWaterTime = calculateNextTime(plantingDate, WATER\_INTERVAL). Создаётся объект Reminder с id = 0, userPlantId = newPlantId, remindTime = nextWaterTime, type = «WATER», isDone = false; сохраняется через reminderDao.insert(waterReminder). Для nextWaterTime планируется задача WorkManager с initialDelay = nextWaterTime – System.currentTimeMillis().

После этого формируется запись истории: объект HistoryEntry с id = 0, userPlantId = newPlantId, timestamp = System.currentTimeMillis(), action = «ADD», description = «Добавлено растение: « + customName; сохраняется через historyDao.insert(entry). Затем вызывается setResult(RESULT\_OK) и finish(), и MainActivity автоматически обновляет список.

1. Отображение списка растений

В MainActivity при запуске подписываются на LiveData<List> через userPlantDao.getAll(). Когда LiveData оповещает об изменении, RecyclerView адаптер получает новый список и вызывает submitList(plants) или notifyDataSetChanged(). При прокрутке списка изображения загружаются через Glide.load(photoUri).into(ivThumbnail) с кешированием для быстрого отображения.

1. Просмотр и выполнение напоминаний

В отдельном сервисе (ForegroundService или Worker) вызываются reminderDao.getDueReminders(System.currentTimeMillis()) для поиска просроченных напоминаний. Для каждого найденного напоминания формируется Notification с текстом «Пора выполнить: [тип напоминания] для [customName]». При взаимодействии пользователя с уведомлением (нажатие «Отметить как выполнено») приложение получает reminderId, выполняет reminderDao.update(reminder.setDone(true)).

Затем создаётся новая запись истории: объект HistoryEntry с id = 0, userPlantId = reminder.userPlantId, timestamp = System.currentTimeMillis(), action = «WATERED» или «FERTILIZED», description = «Напоминание выполнено: « + customName; сохраняется через historyDao.insert(entry). После этого вычисляется время следующего напоминания: nextTime = calculateNextTime(currentRemindTime, WATER\_INTERVAL), создаётся новый Reminder с id = 0, userPlantId = тот же, remindTime = nextTime, type = тот же, isDone = false; сохраняется через reminderDao.insert(nextReminder). Для него планируется новая задача WorkManager с initialDelay = nextTime – System.currentTimeMillis().

1. Редактирование записи о растении

В EditActivity загружается объект UserPlantDTO plant = userPlantDao.getById(userPlantId). Поля формы редактирования заполняются текущими значениями:

etName.setText(plant.getCustomName()), etDescription.setText(plant.getDescription()), etConditions.setText(plant.getConditions()), ivPhoto.setImageURI(Uri.parse(plant.getPhotoUri())), tvPlantingDate.setText(formatDate(plant.getPlantingDate())), etNotes.setText(plant.getNotes()).

После внесения изменений и нажатия «Сохранить» проверяется, что customName не пустое, затем вызываются plant.setCustomName(newName), plant.setDescription(newDescription), plant.setConditions(newConditions), plant.setPhotoUri(newPhotoUri), plant.setPlantingDate(newPlantingDate), plant.setNotes(newNotes); после этого userPlantDao.update(plant). Старые напоминания удаляются методом reminderDao.deleteByUserPlantId(userPlantId).

Создаются новые напоминания аналогично этапу «Добавление нового растения»: вычисляется remindTime, формируется объект Reminder с id = 0, userPlantId = plant.getId(), remindTime = новое значение, type = «WATER», isDone = false; сохраняется через reminderDao.insert(nextReminder) и планируется задача WorkManager. Дальше создаётся запись истории: HistoryEntry entry = new HistoryEntry(0, userPlantId, System.currentTimeMillis(), «EDIT», «Отредактировано растение: « + newName); сохраняется через historyDao.insert(entry). Наконец, вызывается setResult(RESULT\_OK) и finish(), возвращая пользователя в DescriptionActivity или MainActivity.

1. Удаление записи о растении

В DescriptionActivity при нажатии «Удалить» появляется диалог подтверждения. При согласии выполняется reminderDao.deleteByUserPlantId(userPlantId) для удаления всех связанных напоминаний, затем userPlantDao.delete(plant) удаляет сам объект UserPlantDTO.

Создаётся запись истории: объект HistoryEntry с id = 0, userPlantId = прежний идентификатор, timestamp = System.currentTimeMillis(), action = «DELETE», description = «Удалено растение: « + plant.getCustomName(); сохраняется через historyDao.insert(entry). После этого вызывается finish(), и MainActivity обновляет список.

1. Поиск и фильтрация списка растений

В MainActivity имеются SearchView или EditText и кнопка «Фильтр», которая вызывает DialogFragment с полями для объекта FilterParams: текстовый поиск по названию (queryText), диапазон дат посадки (dateFrom, dateTo), фильтр по условиям выращивания (conditionsFilter).

После нажатия «Применить» вызывается метод userPlantDao.getFiltered(”%” + queryText + “%”, dateFrom, dateTo, conditionsFilter), возвращающий LiveData<List>, на который подписывается адаптер через submitList(filteredPlants), что приводит к автоматическому обновлению отображаемого списка.

1. Расчет возраста растения

В DescriptionActivity после получения plantingDate вычисляется ageDays как (System.currentTimeMillis() – plantingDate) / (1000 \* 60 \* 60 \* 24). Поле tvAge получает текст ageDays + « дн.», отображая количество полных дней с момента посадки.

### 2.3.2 Алгоритмы, обеспечивающие функциональность приложения

• Обработчики кнопок

Для кнопки «Добавить растение»:

1. Вызов vibrate(50) для тактильного отклика.
2. Переход к CreateActivity через Intent.

Для кнопок «Сохранить» и «Удалить» в CreateActivity и EditActivity:

1. Валидация обязательных полей (customName, plantingDate).
2. Формирование и сохранение объекта UserPlantDTO или удаление через DAO.
3. Формирование и сохранение объектов Reminder и HistoryEntry.
4. Обновление UI через setResult(RESULT\_OK).
5. Возврат в MainActivity или DescriptionActivity и автоматическое обновление списка.

• Обновление состояния интерфейса

Метод loadPlantList() в MainActivity подписывается на LiveData<List>; при любом изменении данных адаптер PlantRecyclerAdapter обновляет RecyclerView. Метод loadReminderList() в RemindersActivity или DescriptionActivity подписывается на LiveData<List> и вызывает adapter.submitList(reminders). Функции formatDate(long ms) и formatAge(long ms) обеспечивают преобразование даты и возраста растения в читаемый формат.

• Планирование и отмена уведомлений

Метод scheduleReminder(Reminder reminder) вычисляет задержку delay = reminder.getRemindTime() – System.currentTimeMillis(); если delay меньше нуля, напоминание не планируется. В противном случае создаётся OneTimeWorkRequest с initialDelay = delay, а Worker содержит логику отправки Notification. Для отмены задачи вызывается WorkManager.cancelWorkById(reminder.getWorkId()), где workId хранится в таблице Reminder или состоянии ViewModel.

• Валидация ввода данных

Метод validatePlantInput(String name, long date) проверяет, что name не пустое и date меньше или равно System.currentTimeMillis(); в противном случае возвращает false и выводит сообщение об ошибке через Toast или setError в текстовом поле.

• Управление изображениями и кешированием

При загрузке фотографий в адаптере PlantRecyclerAdapter используется Glide.with(context).load(photoUri).centerCrop().placeholder(R.drawable.placeholder).into(ivThumbnail). Для полноразмерного просмотра в DescriptionActivity применяется Glide.with(this).load(photoUri).fitCenter().into(ivPhotoFull). Glide автоматически кеширует изображения во внутреннем хранилище приложения для повторного использования.

• Обработка ошибок базы данных

Все операции insert, update и delete обёрнуты в try–catch-блоки. В случае SQLException приложение отображает сообщение через Toast, логирует ошибку методом Log.e(TAG, «DB error», exception) и продолжает работу с оставшимися данными без критического сбоя.

В совокупности эти алгоритмы обеспечивают надёжную и предсказуемую работу приложения: от добавления новых растений и планирования напоминаний до корректного отображения данных и ведения полной истории действий.

## 2.4 Разработка интерфейса приложения

Интерфейс приложения «Уход за домашними растениями» спроектирован с учётом простоты и наглядности, чтобы пользователь мог быстро добавлять новые растения, просматривать детальную информацию и редактировать существующие записи. Основные компоненты интерфейса выполнены на базе стандартных виджетов Android и оформлены в единой цветовой схеме. В приложении предусмотрены четыре ключевых экрана: список растений, добавление нового растения, просмотр деталей и редактирование записи.

1. Экран списка растений (MainActivity)

Проект экрана списка растений представлен на рисунке 2.

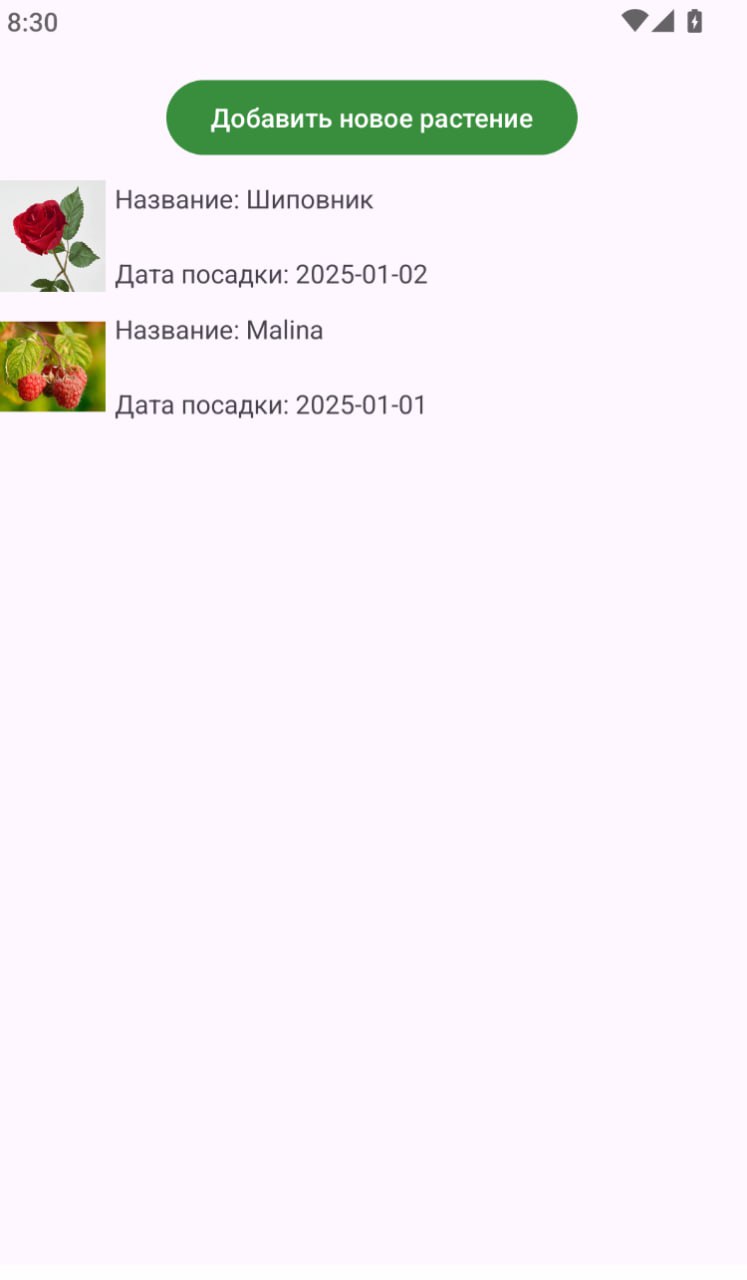


Рисунок 2 – Проект экрана MainActivity

Значения свойств компонентов экрана MainActivity представлены в таблице 1.

Таблица 1 – Свойства компонентов MainActivity

|  |  |  |
| --- | --- | --- |
| Имя  компонента | Свойства | Значение |
| RecyclerView | android:id | @+id/recyclerViewPlant |
| android:layout\_width | 409dp |
| android:layout\_height | 665dp |
| android:layout\_marginStart | 1 |
| app:layout\_constraintStart\_toStartOf | parent |
| app:layout\_constraintTop\_toBottomOf | @+id/createNewPlant |
| app:layout\_constraintEnd\_toEndOf | parent |
| app:layout\_constraintBottom\_toBottomOf | parent |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Имя  компонента | Свойства | Значение |
| Button | android:layout\_height | wrap\_content |
| android:layout\_width | wrap\_content |
| android:id | @+id/createNewPlant |
| android:layout\_marginStart | 160dp |
| android:text | «Добавить новое растение» |
| android:layout\_marginTop | 16dp |
| android:layout\_marginEnd | 161dp |
| android:onClick | createButtonOnClick |
| app:layout\_constraintStart\_toStartOf | parent |
| app:layout\_constraintEnd\_toEndOf | parent |
| app:layout\_constraintEnd\_toEndOf | parent |

1. Экран добавления нового растения (CreateActivity)

Проект экрана добавления нового растения представлен на рисунке 3.

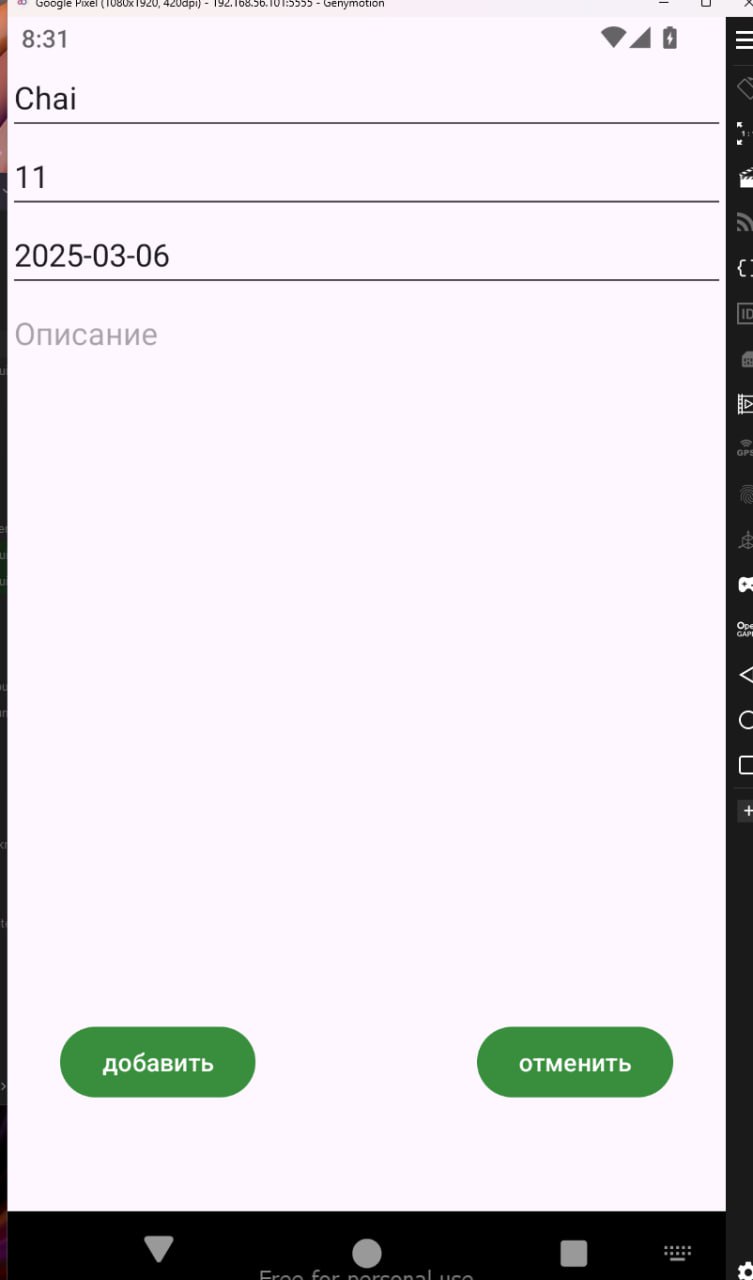


Рисунок 3 – Проект экрана CreateActivity

Значения свойств компонентов экрана CreateActivity представлены в таблице 2.

Таблица 2 – Свойства компонентов CreateActivity

|  |  |  |
| --- | --- | --- |
| Имя компонента | Свойства | Значение |
| ConstraintLayout | android:id | @+id/main |
| android:layout\_widеh | match\_parent |
| android:layout\_height | match\_parent |
| tools:context | “.CreateActivity” |
| EditText | android:id | @+id/create\_name\_editText |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| android:ems | 10 |
| android:hint | “Название” |
| android:inputType | text |
| app:layout\_constraintTop\_toTopOf | @+id/linearLayout |
| app:layout\_constraintStart\_toStartOf | parent |
| app:layout\_constraintEnd\_toEndOf | parent |
| EditText | android:id | @+id/create\_wateringDays\_editText |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| android:ems | 10 |
| android:hint | “Количество дней между поливами” |
| android:inputType | text |
| EditText | android:id | @+id/create\_data\_editText |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| android:ems | 10 |
| android:hint | date |
| EditText | android:id | @+id/create\_description\_editText |
| android:layout\_width | match\_parent |
| android:layout\_height | 386dp |
| android:hint | «Описание» |
| Button | android:id | @+id/create\_plant\_button |
| android:layout\_width | wrap\_content |
| android:layout\_height | wrap\_content |
| android:text | «добавить» |
| android:layout\_marginStart | 30dp |
| android:layout\_marginTop | 550dp |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Имя компонента | Свойства | Значение |
| Button | android:text | «отменить» |
| android:id | @+id/cancel\_create\_button |
| android:layout\_width | wrap\_content |
| android:layout\_height | wrap\_content |
| android:layout\_marginEnd | 30dp |
| android:layout\_marginTop | 550dp |

1. Экран просмотра деталей растений (DescriptionActivity)

Проект экрана просмотра деталей представлен на рисунке 4.

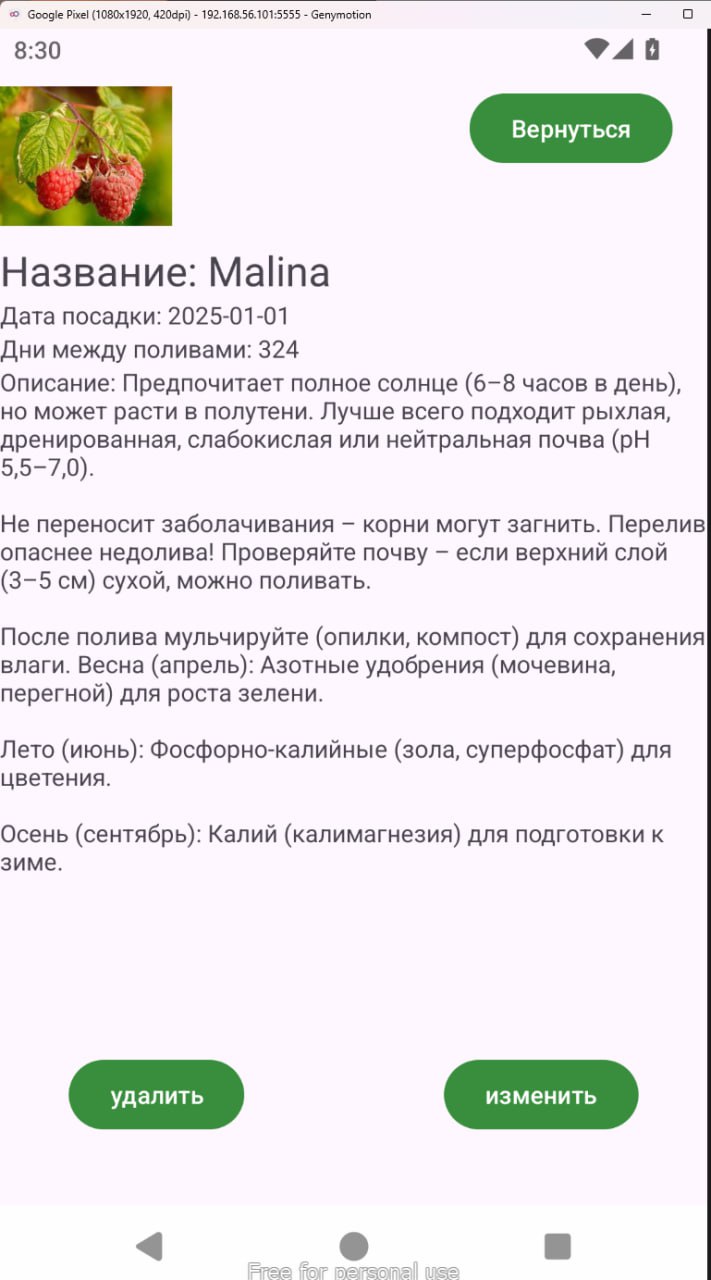


Рисунок 4 – Проект экрана (DescriptionActivity)

Значения свойств компонентов экрана LoginActivity представлены в таблице 3.

Таблица 3 – Свойства компонентов LoginActivity

|  |  |  |
| --- | --- | --- |
| Имя компонента | Свойства | Значение |
| ImageView | android:id | @+id/plant\_image\_description |
| android:layout\_width | 100dp |
| android:layout\_height | 100dp |
| android:srcCompat | @drawable/ic\_launcher\_background |
| TextView | android:id | @+id/textView\_plant\_name\_desciption |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| android:textSize | 24sp |
| TextView | android:id | @+id/textView\_plant\_date\_desciption |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| TextView | android:id | @+id/textView\_plant\_wateringDays\_desciption |
| android:layout\_width | match\_parent |
| android:layout\_height | wrap\_content |
| TextView | android:id | @+id/textView\_plant\_description\_desciption |
| android:layout\_width | match\_parent |
| android:layout\_height | 464dp |
| Button | android:id | @+id/createDate |
| android:layout\_width | wrap\_content |
| android:layout\_height | wrap\_content |
| android:text | «изменить» |
| android:layout\_marginEnd | 40dp |
| android:layout\_marginBottom | 40dp |
| Button | android:id | @+id/button2 |
| android:layout\_width | wrap\_content |
| android:layout\_height | wrap\_content |
| android:text | «Вернуться» |
| android:layout\_marginEnd | 10dp |
| android:layout\_marginBottom | 20dp |

1. Экран редактирования записи (EditActivity)

Проект экрана сервисных центров представлен на рисунке 5.



Рисунок 5 – Проект экрана (EditActivity)

Значения свойств компонентов экрана EditActivity представлены в таблице 4.

Таблица 4 – Свойства компонентов EditActivity

|  |  |  |
| --- | --- | --- |
| Имя компонента | Свойства | Значение |
| ImageView | android:id | @+id/imageEdit |
| android:layout\_width | 100dp |
| android:layout\_height | 100dp |
| android:layout\_marginStart | 30dp |
| android:layout\_marginTop | 52dp |
| EditText | android:id | @+id/nameEdit |
| android:layout\_width | 217dp |
| android:layout\_height | 40dp |
| android:hint | «Название» |
| EditText | android:id | @+id/editTextDate |
| android:layout\_width | 217dp |
| android:layout\_height | 40dp |
| android:hint | «Дата посадки» |
| EditText | android:id | @+id/textView\_plant\_wateringDays\_desciption |
| android:layout\_width | 217dp |
| android:layout\_height | 40dp |
| android:hint | «Дней между поливами» |
| EditText | android:id | @+id/editDescription |
| android:layout\_width | 350dp |
| android:layout\_height | 477dp |
| android:hint | «описание» |
| Button | android:id | @+id/backButtonEdit |
| android:layout\_width | wrap\_content |
| android:layout\_height | wrap\_content |
| android:text | «изменить» |
| android:layout\_marginEnd | 30dp |
| android:layout\_marginBottom | 35dp |

Такое оформление обеспечивает единообразие интерфейса, лёгкость восприятия и быстрый доступ ко ключевым функциям приложения.

## 2.5 Описание структуры приложения

Приложение состоит из следующих модулей, представленных на рис.6:

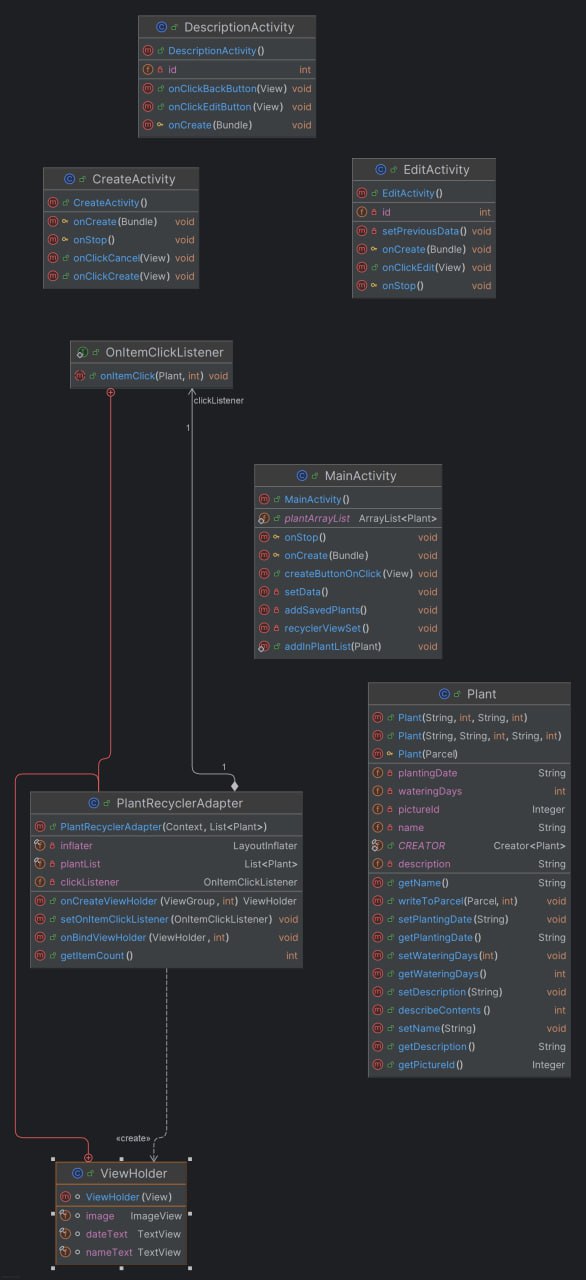


Рисунок 6 – Диаграмма классов

Такая модульная структура обеспечивает простой рост и модификацию приложения: добавление новых типов напоминаний, расширение справочной базы растений или внедрение синхронизации с облачным хранилищем сводится к расширению соответствующих моделей, репозиториев и экранов, не затрагивая фундаментальные слои.

# ЗАКЛЮЧЕНИЕ

В процессе разработки мобильного приложения для учёта и ухода за домашними растениями были изучены многочисленные источники: официальная документация Android Studio, материалы по мобильной разработке на языке Java, руководства по использованию библиотеки Room для локального хранения данных, руководство по работе с RecyclerView и ConstraintLayout, а также пособия по библиотеке Glide для загрузки и кеширования изображений.

В результате создано приложение, реализующее полный цикл работы с «зелёными питомцами» в офлайн-режиме:

* Пользователь может добавлять новое растение, вводя его название, количество дней между поливами, дату посадки и подробное описание.
* Каждая запись сохраняется локально в базе Room как объект UserPlantDTO, а в интерфейсе отображается посредством RecyclerView с использованием адаптера PlantRecyclerAdapter.
* При выборе конкретного растения открывается экран с детальной информацией: изображение, название, дата посадки, интервал полива и полное текстовое описание.
* С экрана просмотра можно отредактировать запись (передать объект Plant в EditActivity) или удалить её, после чего список автоматически обновляется.
* Пользовательская история операций (дата добавления, изменения, удаления) сохраняется в сущности HistoryEntry и может быть показана в виде отдельного экрана (опционально), тем самым обеспечивая возможность проследить динамику ухода за каждым растением.
* Напоминания о поливе и удобрениях (класс Reminder) рассчитываются на основе даты посадки и заданного интервала, и планируются через WorkManager; при срабатывании формируются локальные уведомления, помогающие своевременно ухаживать за растениями.
* Загрузка изображений реализована через библиотеку Glide, что обеспечивает быстрый рендеринг и кеширование миниатюр в списке, а также полноразмерных фотографий в экране описания.

Приложение может быть использовано любителями комнатных растений для систематизации знаний о каждом экземпляре, расчёта графика полива и хранения полной истории ухода. Благодаря архитектуре MVVM (модули ViewModel, Repository) и модульному разделению на слои (модель Plant, адаптер PlantRecyclerAdapter, активити MainActivity, CreateActivity, DescriptionActivity, EditActivity) приложение легко масштабируется и модифицируется.

Для повышения удобства и расширения функционала в будущих версиях рекомендуется:

* Интегрировать облачную синхронизацию данных (например, через Firebase Realtime Database или Firestore), чтобы пользователь мог просматривать свои записи с разных устройств и создавать резервные копии.
* Расширить систему напоминаний: добавить гибкую настройку уведомлений (выбор звука, повторяемые циклы, исключения по датам), а также возможность вручную отмечать события через уведомление («Отметить полив выполненным»).
* Реализовать функцию автоматического распознавания растений или получение справочной базы из внешних API (например, PlantNet API), чтобы пользователь мог вводить вид одним нажатием, а приложение подгружало стандартное описание и условия выращивания.
* Добавить раздел «Статистика»: графики по частоте полива, динамике роста (при наличии полей для фиксации высоты/размеров), а также напоминания о подкормке и пересадке в зависимости от времени жизни растения.
* Расширить поддержку фото: дать возможность загружать несколько изображений одного растения (для отслеживания состояния листьев), а также реализовать редактор фото (обрезка, поворот) непосредственно в приложении.
* Внедрить систему категорий и тегов (например, «Суккуленты», «Орхидеи», «Фиалки»), чтобы облегчить поиск/фильтрацию большого количества записей и сформировать отдельные коллекции.
* Обеспечить мультиязычность интерфейса для расширения аудитории и добавить встроенные подсказки (tooltips) по заполнению полей при первом запуске.

Таким образом, разработанное приложение демонстрирует практические приёмы работы с современными Android-компонентами (Room, LiveData, RecyclerView, WorkManager, Glide), а также полезное применение концепций хранениия данных и уведомлений в офлайн-режиме. Полученные результаты могут быть использованы как отправная точка для более сложных систем ухода за растениями, включающих аналитические модули, интерактивные справочники и облачную синхронизацию.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в библиотеку Room (GeeksforGeeks) [Электронный ресурс]. URL: https://www.geeksforgeeks.org/introduction-to-room-persistent-library-in-android/ (дата обращения: 05.04.2025).
2. Документация по работе в Android Studio [Электронный ресурс]. URL: https://developer.android.com/ (дата обращения: 12.03.2025).
3. Документация по библиотеке Room (Android Jetpack) [Электронный ресурс]. URL: https://developer.android.com/training/data-storage/room (дата обращения: 05.05.2025).
4. Описание работы с DAO в Room (Android Jetpack) [Электронный ресурс]. URL: https://developer.android.com/training/data-storage/room/accessing-data (дата обращения: 05.05.2025).
5. Полный обзор Room Persistence Library (Daily.dev) [Электронный ресурс]. URL: https://daily.dev/blog/android-room-persistence-library-complete-guide (дата обращения: 05.05.2025).
6. Руководство по внедрению Room Database (Medium, Amit Raikwar) [Электронный ресурс]. URL: https://amitraikwar.medium.com/getting-started-with-room-database-in-android-fa1ca23ce21e (дата обращения: 05.05.2025).
7. Руководство CodePath Android Cliffnotes: Room Guide [Электронный ресурс]. URL: https://guides.codepath.com/android/Room-Guide (дата обращения: 22.04.2025).
8. API-справочник по пакету androidx.room [Электронный ресурс]. URL: https://developer.android.com/reference/androidx/room/package-summary (дата обращения: 05.05.2025).
9. Glide v4: быстрое и эффективное загрузка изображений (официальный сайт) [Электронный ресурс]. URL: https://bumptech.github.io/glide/ (дата обращения: 05.04.2025).
10. Glide: Getting Started (официальная документация) [Электронный ресурс]. URL: https://bumptech.github.io/glide/doc/getting-started.html (дата обращения: 08.03.2025).
11. Glide — пример использования в Android (GeeksforGeeks) [Электронный ресурс]. URL: https://www.geeksforgeeks.org/image-loading-caching-library-android-set-2/ (дата обращения: 08.03.2025).

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

https://github.com/sascrystal/botanistGuide

------------------MainActivity.java------------------

package com.example.botanistguide;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import java.lang.reflect.Type;

import java.util.ArrayList;

import java.util.List;

public class MainActivity extends AppCompatActivity {

public static ArrayList<Plant> plantArrayList = new ArrayList<>();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

if(plantArrayList.isEmpty()){

addSavedPlants();

}

if(plantArrayList.isEmpty()){

setData();

}

recyclerViewSet();

}

@Override

protected void onStop() {

super.onStop();

Gson gson = new Gson();

String json = gson.toJson(plantArrayList);

SharedPreferences prefs = getSharedPreferences("my\_prefs", MODE\_PRIVATE);

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

}

private void recyclerViewSet(){

RecyclerView recyclerView = findViewById(R.id.recyclerViewPlant);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

PlantRecyclerAdapter adapter = new PlantRecyclerAdapter(this, plantArrayList);

recyclerView.setAdapter(adapter);

adapter.setOnItemClickListener((plant, position)->{

Intent intent = new Intent(MainActivity.this, DescriptionActivity.class);

intent.putExtra("id",position);

startActivity(intent);

});

}

public void createButtonOnClick(View v){

Intent intent = new Intent(MainActivity.this,CreateActivity.class);

startActivity(intent);

}

private void addSavedPlants(){

SharedPreferences prefs = getSharedPreferences("my\_prefs", MODE\_PRIVATE);

Gson gson = new Gson();

String json = prefs.getString("plants\_list", "");

Type type = new TypeToken<List<Plant>>() {}.getType();

List<Plant> savedPlants = gson.fromJson(json, type);

if(savedPlants != null&&!savedPlants.isEmpty()){

plantArrayList.addAll(savedPlants);

}

}

private void setData(){

plantArrayList.add(new Plant("Шиповник",6,"Предпочитает полное солнце (6–8 часов в день), но может расти в полутени. Лучше всего подходит рыхлая, дренированная, слабокислая или нейтральная почва (pH 5,5–7,0).\n" +

"\n" +

"Не переносит заболачивания – корни могут загнить. Перелив опаснее недолива! Проверяйте почву – если верхний слой (3–5 см) сухой, можно поливать.\n" +

"\n" +

"После полива мульчируйте (опилки, компост) для сохранения влаги. Весна (апрель): Азотные удобрения (мочевина, перегной) для роста зелени.\n" +

"\n" +

"Лето (июнь): Фосфорно-калийные (зола, суперфосфат) для цветения.\n" +

"\n" +

"Осень (сентябрь): Калий (калимагнезия) для подготовки к зиме.",R.drawable.images));

plantArrayList.add(new Plant("Малина", 8,"Семейство: Розовые (Rosaceae)\n" +

"\n" +

"Тип: Листопадный полукустарник\n" +

"\n" +

"Высота: 1,5–2,5 м\n" +

"\n" +

"Побеги: Двухлетние (первый год – зелёные, второй – одревесневшие, плодоносящие)\n" +

"\n" +

"Листья: Овальные, зубчатые, зелёные сверху, белесые снизу\n" +

"\n" +

"Цветки: Белые, собраны в кисти\n" +

"\n" +

"Плоды: Сборные костянки (красные, жёлтые или чёрные в зависимости от сорта)\n" +

"\n" +

"Период плодоношения:\n" +

"\n" +

"Обычная малина – июнь-июль,\n"

,R.drawable.rasberry));

}

public static void addInPlantList(Plant plant){

plantArrayList.add(plant);

}

}