

ЛИСТ ДЛЯ ЗАМЕЧАНИЙ

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc169778171)

[1 Средства разработки 7](#_Toc169778172)

[2 Прототип приложения 9](#_Toc169778173)

[3 База данных 10](#_Toc169778174)

[4 Интерфейс пользователя 12](#_Toc169778175)

[5 Разработка функционала приложения 15](#_Toc169778176)

[6 Тестирование 16](#_Toc169778177)

[Заключение 17](#_Toc169778178)

# ВВЕДЕНИЕ

Актуальность темы разработки мобильного приложения для управления умными теплицами или же аппаратным комплексом связано с уменьшением огромной доступностью мобильных телефонов, уменьшением расходов на создание панели управления аппаратными комплексами, так же позволяет производителям аппаратных комплексов заниматься ими, а не панелями управления. Разработка мобильных приложений требует особого подхода и знаний, так как они обеспечивают работу и взаимодействие клиентских устройств с аппаратными комплексами, а клиентские устройства могут быть разных производителей и соответственно мощностей.

Современные мобильные приложения должны быть надёжными, безопасными и эффективными. Они должны обеспечивать быстрый отклик на запросы пользователей, обрабатывать большие объёмы данных и поддерживать работу с различными разрешениями экранов и различным качеством связи, в частности и ее отсутствия.

Для разработки мобильного приложения используются различные языки программирования и фреймворки. Выбор языка программирования зависит от требований проекта, опыта разработчика и доступных ресурсов.

Цель работы – разработать мобильное приложение для управления умными теплицами.

В связи с поставленной целью были сформулированы следующие задачи:

– выбрать необходимые средства для разработки (язык, среду);

– разработать прототип приложения;

– спроектировать базу данных,

– разработать интерфейс пользователя;

– разработать функционал;

– произвести тестирование.

Ожидаемые результаты: рабочее мобильное приложения для управления умными теплицами.

# 1 Средства разработки

Для разработки Android–приложений используются следующие языки программирования:

Java — официальный и самый популярный язык программирования для Android.

Kotlin — современный язык программирования, совместимый с Java и имеющий больше синтаксического сахара.

C/C++ — языки программирования, которые используются через Android NDK для создания нативных библиотек.

Также для разработки Android–приложений можно использовать языки веб–разработки, такие как HTML, CSS и JavaScript, через инструменты PhoneGap, Adobe Cordova и React Native от Facebook.

Среды разработки для Android включают:

Android Studio — официальная интегрированная среда разработки (IDE) от Google, предоставляющая полный набор инструментов для создания приложений Android с использованием Java или Kotlin.

Kotlin Multiplatform Mobile (KMM) — общая платформа кодовой базы для создания собственных приложений Android и iOS с использованием языка Kotlin.

React Native — платформа с открытым исходным кодом на основе JavaScript для создания высокопроизводительных нативных приложений для Android и iOS.

Flutter — набор пользовательского интерфейса от Google для создания высокопроизводительных приложений, скомпилированных в собственном коде, для нескольких платформ.

Xamarin — кроссплатформенная среда разработки приложений от Microsoft, использующая C# для создания нативных приложений для Android, iOS и Windows.

Исходя из рекомендаций Google и содержания курса были выбраны:Kotlin и Android Studio.

# 2 Прототип приложения

Прототип должен осуществлять следующий функционал:

– совершать запросы к удаленному серверу по средствам HTTP;

– обрабатывать результаты запросов, парсить JSON;

– вести локальную базу данных, позволять добавлять, удалять и изменять сущности;

– информативно отображать данные от умных теплиц в различных представлениях;

– интерфейс должен быть понятен и интуитивен;

– должен быть свой стиль оформления, минималистичный и приятный на глаз.

# 3 База данных

База данных состоит из сущностей: теплица, сенсор, переключатель, свойство (характеристика), данные от сенсора или переключателя, рисунок 1.

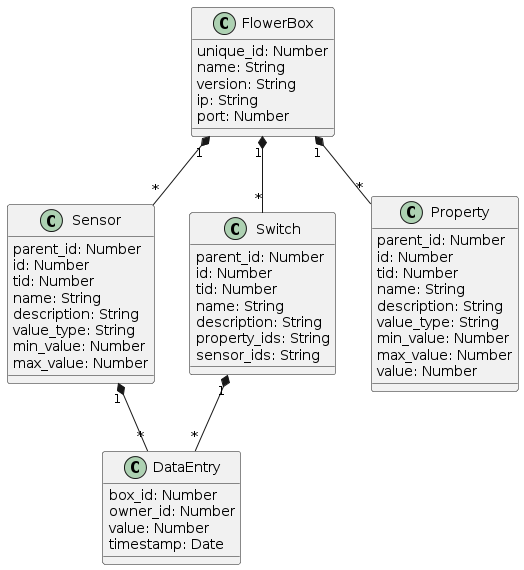


Рисунок 1 – Схема базы данных

Теплица состоит из полей:

– name – имя;

– version – версия ПО;

– ip – IP адрес;

– port – сетевой порт;

– unique\_id – уникальный ИД (каждый микроконтроллер одного семейства нумеруется на заводе).

Сенсор состоит из полей:

– name – имя;

– description – описание, что делает, в чем измеряет;

–value\_type – какие данные возвращает (int, float, string, пока только float);

– parent\_id – ID теплице к которой принадлежит;

– id – уникальный ИД внутри теплицы, по сути теплица считает от 0 до N;

– tid – уникальный идентификатор, что это за сенсор, по сути имея базу данных с их описаниями и названиями, по нему можно доставать необходимую информацию, пример термометр, барометр;

– min\_value – минимальное значение данных;

– max\_value – максимальное значение данных.

Переключатель состоит из тех же самых полей, что и сенсор, но вдобавок имеет следующие:

– properties\_ids – ИД свойств которые непосредственно влияют на работу переключателя;

– sensors\_ids – ИД сенсоров от которых зависит работа переключателя.

Данные состоит из полей:

– value – значение данных;

– timestamp – UNIX время, когда были получены;

– box\_id – ID теплице к которой принадлежит;

– owner\_id – ИД сенсора или переключателя.

# 4 Интерфейс пользователя

Исходя из специфики данных и самого приложения, много где используется отображения списками, ибо данные динамические, а именно Recycle View, пример на рисунке 1.

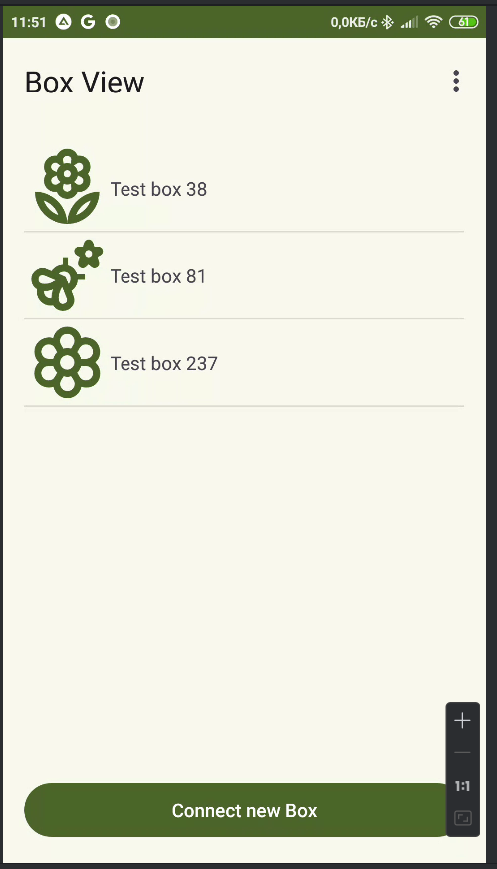


Рисунок 1 – Главный экран

Для ожидания используются диалоговые окна с отсутствием возможности из закрыть, пример рисунке 2.

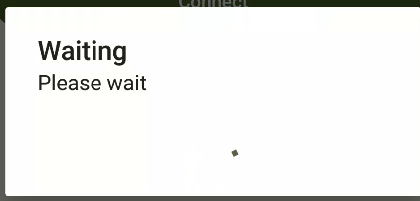


Рисунок 2 – Диалоговое окно ожидания

Так же для упрощения логики, используются кнопки для совершения запросов, а не автоматически при открытии фрагментов и активностей, пример на рисунке 3.



Рисунок 3 – Окно сенсоров теплицы

Была сгенерирована светлая и темная тема, со своим основным цветом оттенком зеленого, как бы обозначающая то для чего оно надо, пример рисунок 4.

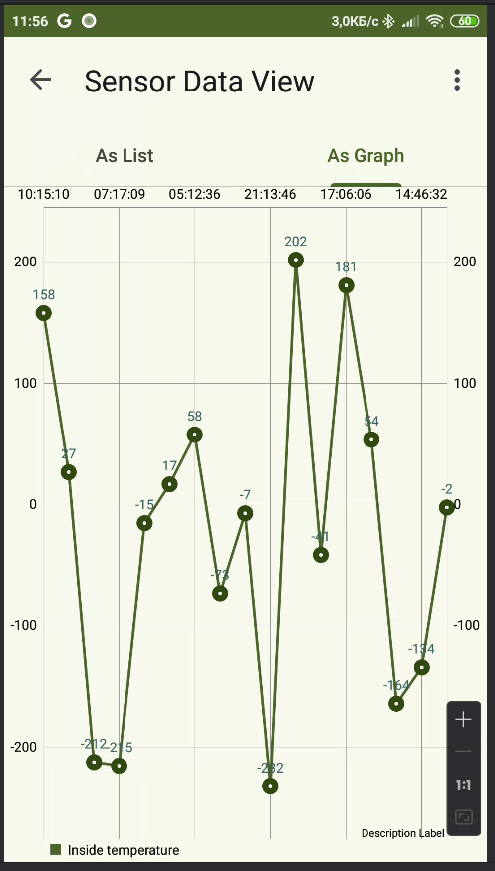


Рисунок 4 – График данных сенсора в светлой теме

Для различия теплиц, сенсоров, переключателей и свойств, были подобраны иконки.

# 5 Разработка функционала приложения

Исходя из необходимого функционала были выбраны следующие библиотеки:

– Room – библиотека для взаимодействия с базой данных по средствам object relation mapping (ORM);

– Retroofit2 – библиотека для асинхронного сетевого взаимодействия;

– Gson – библиотека для сериализации и десериализации объектов в и из JSON формата;

– MPAndroidChart – библиотека для отображения данных как различные виды диаграмм.

# 6 Тестирование

Тестирование производилось по средством «потыкать» и «подумать» исходя из простоты рабочего прототипа и сжатых сроков.

Сетевое взаимодействие, JSON и базу данных можно протестировать по средствами Junit.

Основные ошибки найденные при тестирование были:

– непредсказуемое поведение (краши) при обработки ответов от запросов (активность или фрагмент могут быть закрыты);

– не корректное описание классов для парсинга из JSON.

### Заключение

В ходе курса цель была достигнута, все поставленные задачи были выполнены:

– получены навыки мобильной разработки;

– изучены основы языка Kotlin;

– изучены основные возможности Android;

– выбраны необходимые средства для разработки (язык, среду);

– разработан прототип приложения;

– спроектирована базу данных,

– разработан интерфейс пользователя;

– разработан функционал приложения;

– произведено тестирование.

Из достигнутых задач можно сделать вывод, что было создано рабочее мобильное приложение.

Дальнейшие перспективы:

- сервер с базой данных о клиентах и возможных сенсорах, переключателях, характеристиках (мета программирование);

- доработка логики ожидания запросов;

- скрытие логики по синхронизации;

- разработка универсального API для использования данного приложения с семейством устройств;

- уведомления о критических параметрах от сенсоров и сбоях работы.