

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА  
ЗВФ

Кафедра «Вычислительные системы и технологии»  
«Информационно-управляющие вычислительные системы»

Выпускная квалификационная работа  
на тему:

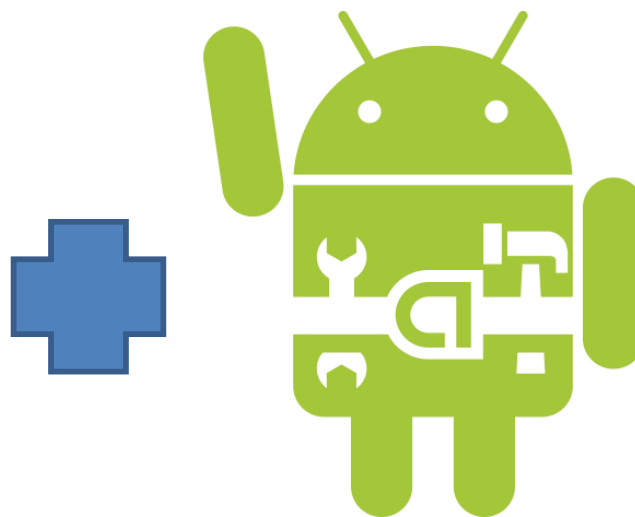
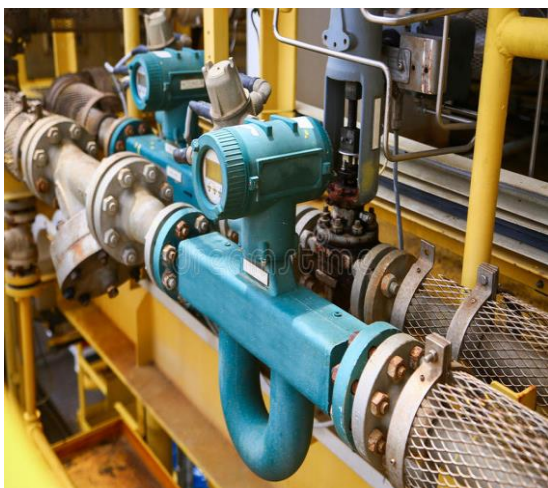
Приложение на платформе Android для системы  
мобильного обхода

**Выполнил:** студент гр. 14 ВМ Думин М. В.

**Научный руководитель:** Викулова Е.Н.

# Актуальность разработки

- Время
- Качество
- Без рутинных операций
- Использование современных технологий



# Цели и задачи работы

Цель разработки — создание программной системы для автоматизации контроля за проведением обходов на предприятиях, повышения качества их осуществления, освобождение персонала от рутинных операций по заполнению журналов обхода.

Задачи работы:

1. Сформулировать функциональные требования к приложению;
2. Выбрать средств для реализации;
3. Определить основные компоненты системы;
4. Разработать алгоритм работы приложения;
5. Разработать макет дизайна приложения;
6. Разработать базу данных;
7. Реализовать приложение на устройстве;
8. Смоделировать тестовый сервер;
9. Протестировать приложение;

# Функциональные требования

- Организация идентификации датчиков при совершении обхода;
- Реализация подключения к серверу с данными о показаниях датчиков;
- Организация автоматического получения данных о показаниях идентифицированных датчиков;
- Сохранение информации, полученной с датчиков, в памяти устройства;
- Отображение информации полученной с датчиков в виде списка датчиков и значений их показаний.
- Необходимо предоставить пользователю возможность выборочного удаления записей или полную очистку приложения от данных с датчиков;
- Требуется организовать автоматическую отправку данных о совершении обхода, для поддержания трудовой дисциплины.
- Реализация интуитивно понятного пользовательского интерфейса.

# Выбор средств для реализации

## Целевое устройство

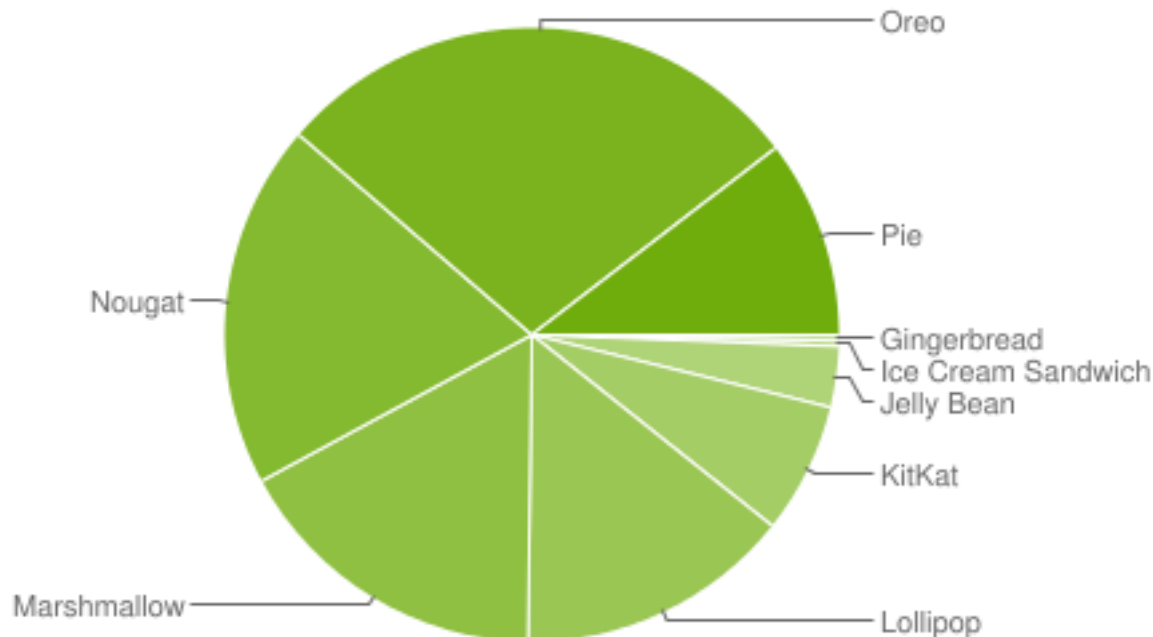
В качестве устройства был выбран смартфон под управлением ОС Android, так он предоставляет широкий спектр возможностей для выполнения различных задач, при этом существует множество моделей от простых и относительно дешевых до взрывозащищённых и ударопрочных



# Выбор средств для реализации

## Версия операционной системы

В результате проведенного анализа было решено поддерживать версии начиная с 4.1 для охвата большого количества платформ Android.



# Выбор средств для реализации

## Идентификация датчиков

Так как датчиков может быть огромное множество, а их типы и назначение повторяются, целесообразно их структурировать в виде каталогов. Имена вложенных каталогов указывать через «слеш» ( / ), последним будет указано название датчика, например:

'Название\_системы / название\_подсистемы / имя\_датчика' - это будет полное имя датчика. Полное имя датчика будет использоваться для получения информации с него.

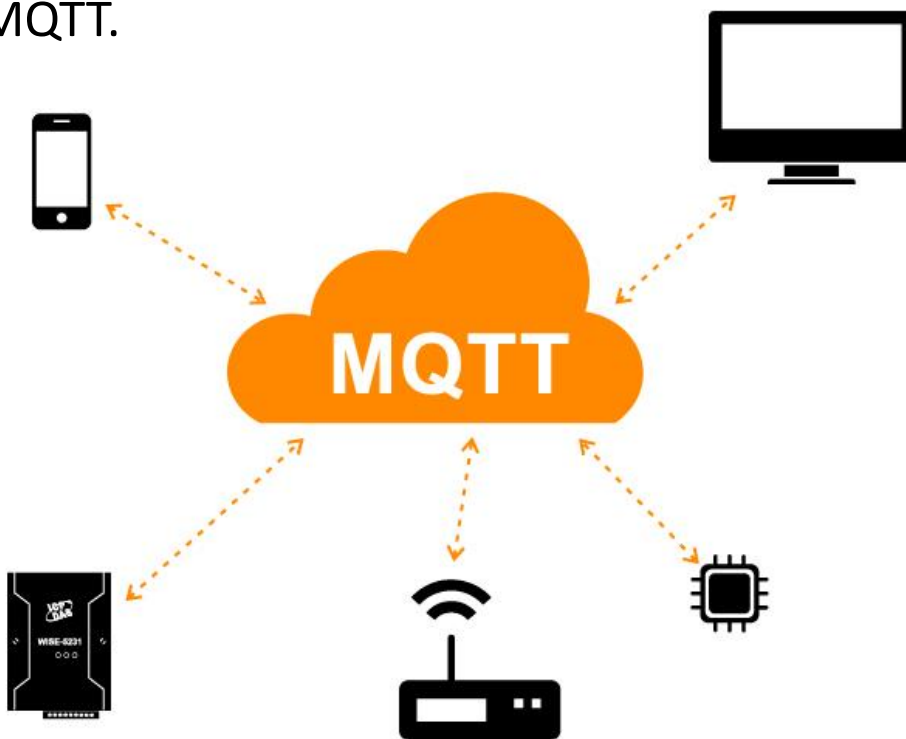
Для того чтобы быстро и точно передать имя датчика в приложение можно использовать **Bluetooth-маяки** и **NFC-метки**, закодировав его в соответствующий сигнал. В рамках дипломной работы я заменил эти два способа передачи данных считыванием QR-кодов. Данная функция работает аналогично, только используется не чип NFC или bluetooth модуль, а камера смартфона.



# Выбор средств для реализации

## Протокол передачи данных

Для взаимодействия между собой устройства используют различные промышленные протоколы, одним из популярных протоколов для этой цели является MQTT.





# Выбор средств для реализации

## Хранение информации

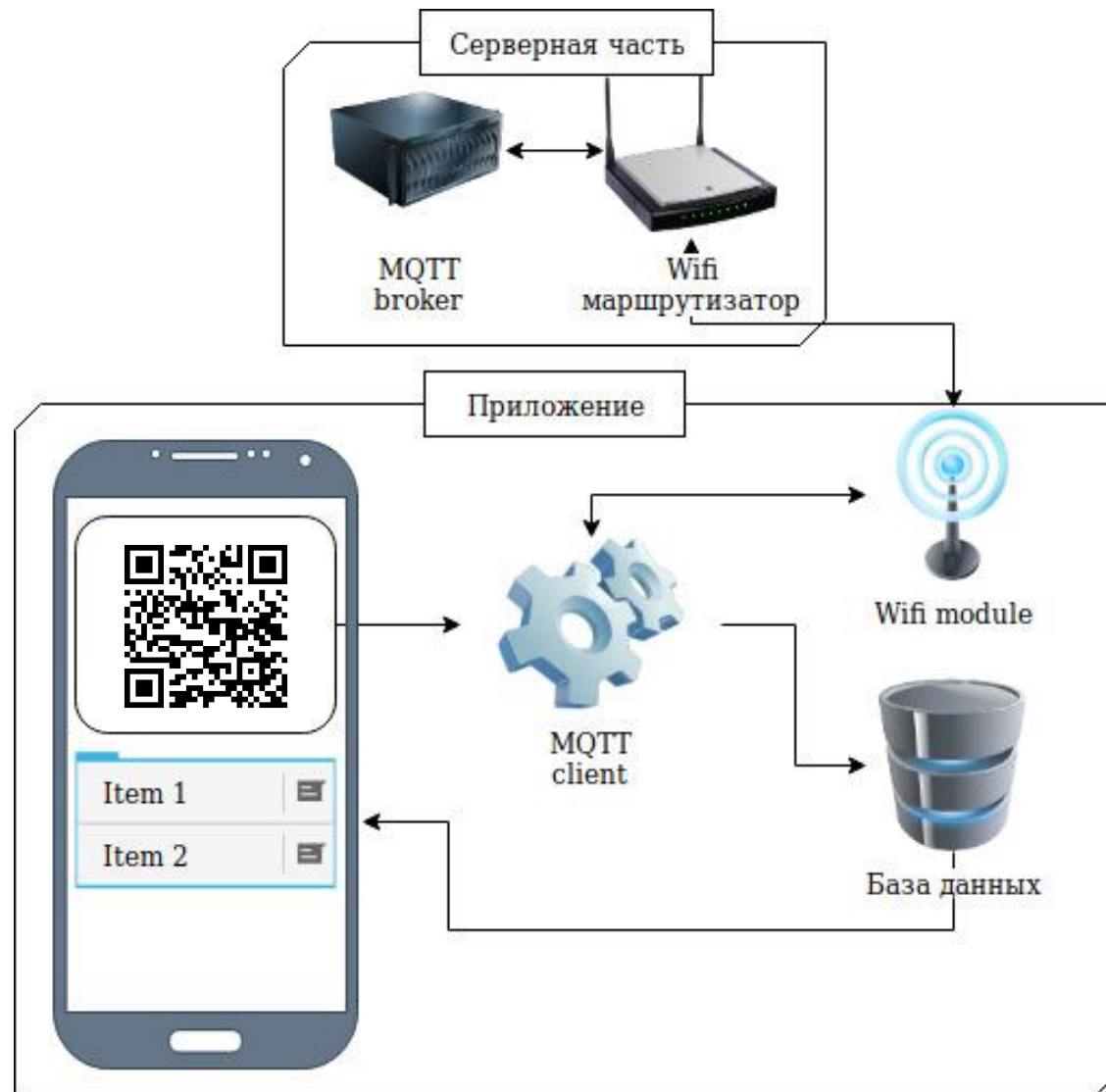
Информацию сохраняемая в приложении делится на два группы:

- Параметры камеры(подсветка, автофокусировка) и настройки соединения
- Показания датчиков

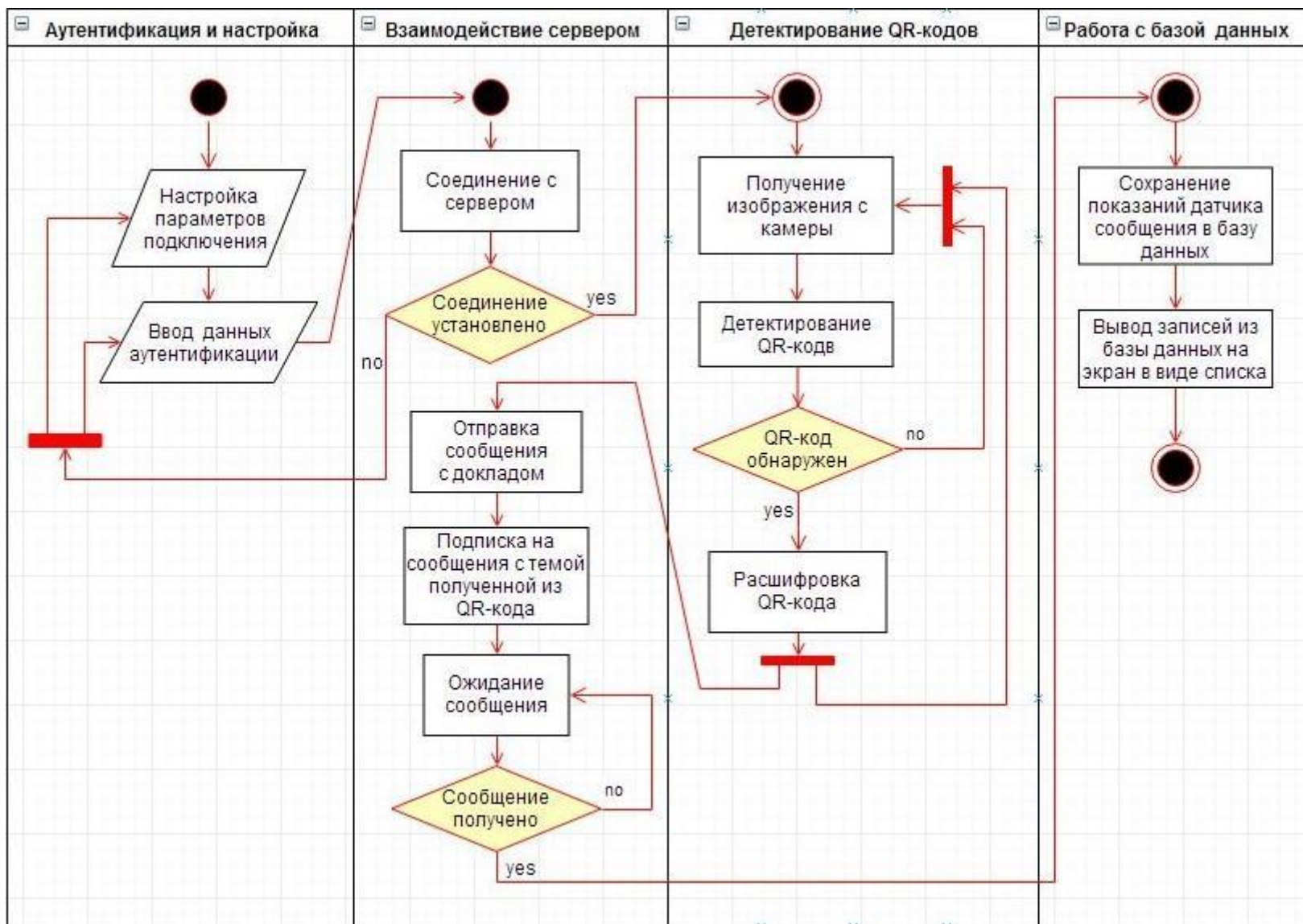
Первая группа характеризуется небольшим, но постоянным объёмом данных. Поэтому для неё в качестве хранилища будет использован `SharedPreferences` — постоянное хранилище на платформе Android, используемое приложениями для хранения своих настроек.

Вторая группа — показания датчиков — очень динамична в процессе работы и может как увеличивать объём данных, используемых в приложении, так и уменьшать. Следовательно для хранения информации больше подходит база данных. Стандартным решением для этих целей является база данных SQLite

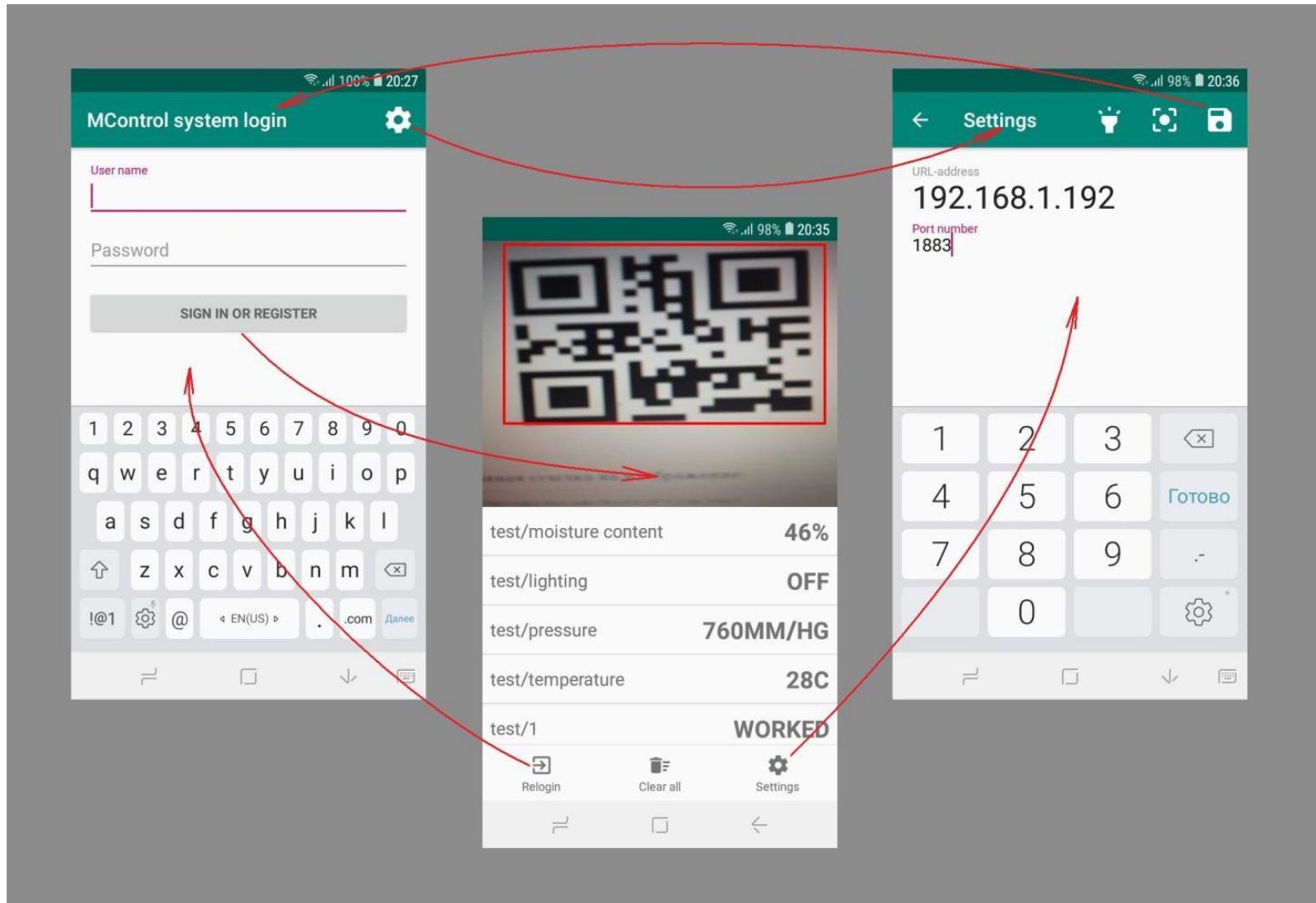
# Структурная схема системы



# Алгоритм работы приложения



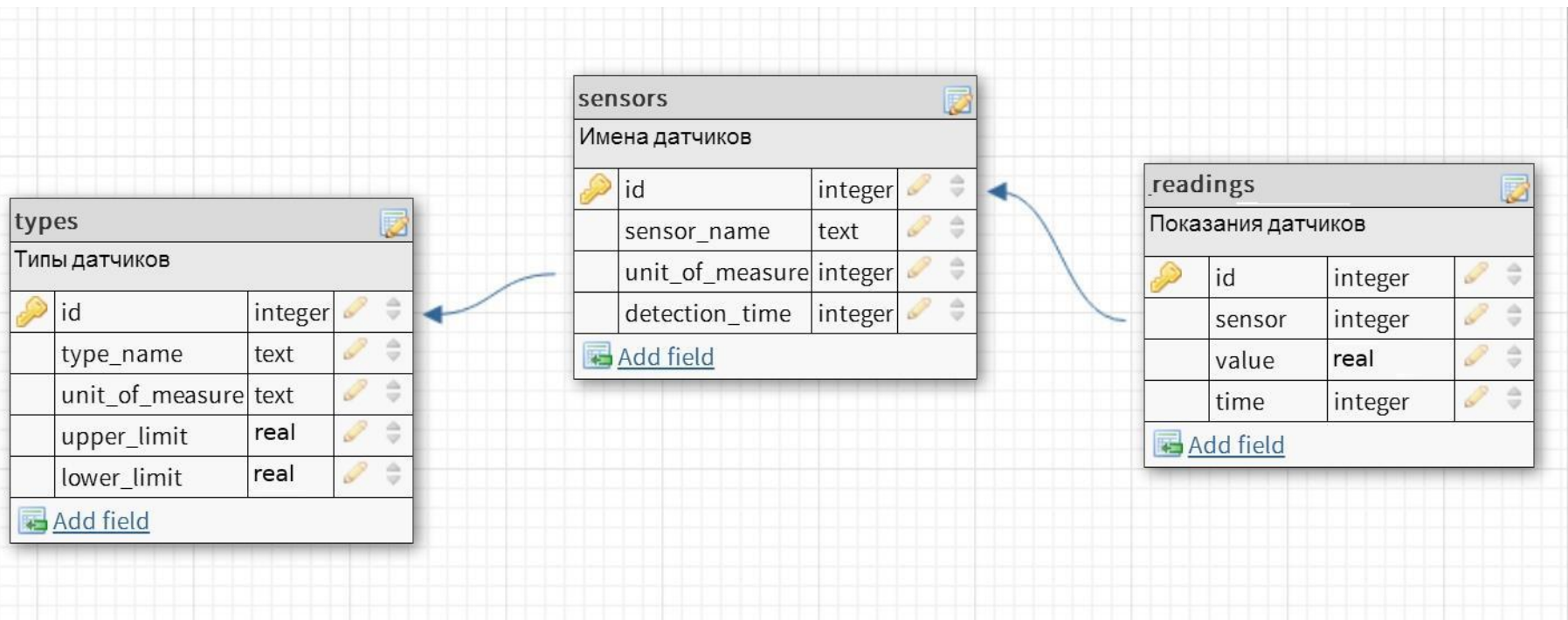
# Навигационная карта приложения



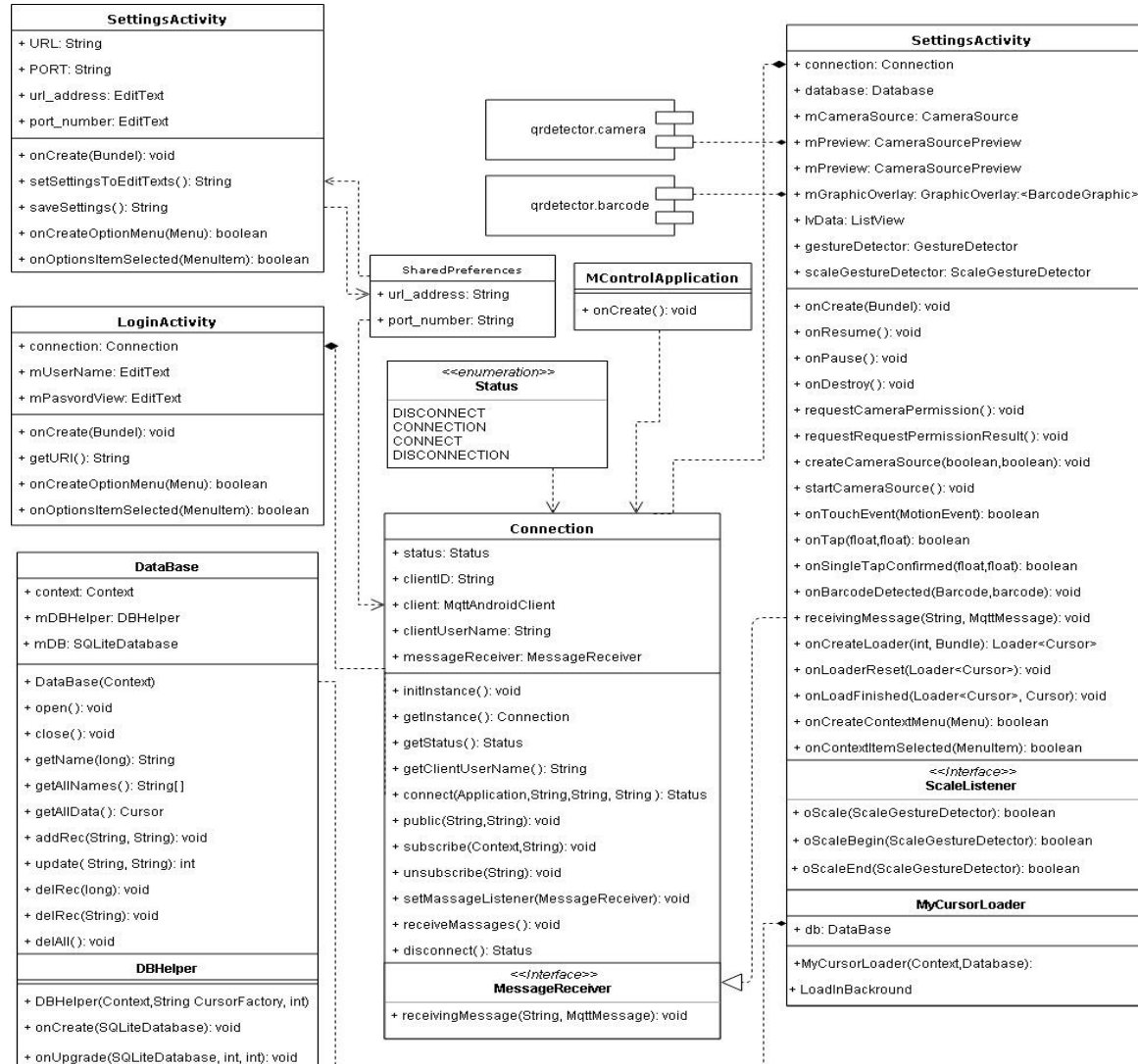
# База данных приложения

Для хранения информации о показаниях датчиков приложение использует базу данных SQLite. SQLite позволяет реализовать реляционную модель данных. Характеристики реляционных баз данных и реляционной модели данных:

- Использование ключей
- Отсутствие избыточности данных
- Ограничение ввода
- Поддержание целостности данных



# Диаграмма классов



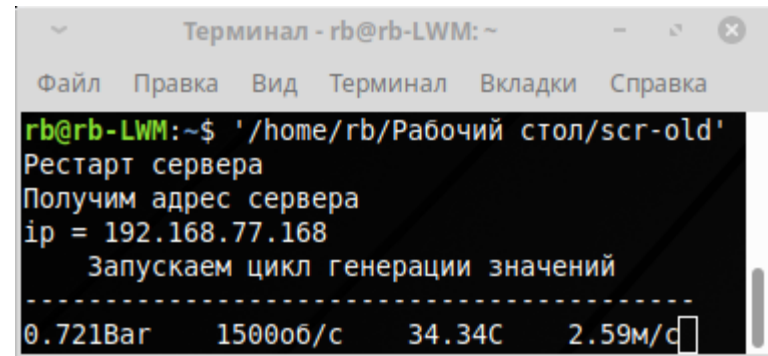


# Моделирование работы сервера

Для проверки работоспособности приложения потребуется:

- Компьютер с возможностью подключения к сети и сетевой ОС;
- Маршрутизатор WiFi;
- MQTT - брокер;
- Скрипт для моделирования генерации показаний датчиков;
- QR-коды с зашифрованными в них именами датчиков.

В бесконечном цикле скрипт генерирует случайные значения и посредством команды `mosquitto_pub` отправляет их на сервер, который перенаправляет эти сообщения клиентам, подписавшимся на соответствующие темы.



```
Терминал - rb@rb-LWM: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
rb@rb-LWM:~$ '/home/rb/Рабочий стол/scr-old'
Рестарт сервера
Получим адрес сервера
ip = 192.168.77.168
    Запускаем цикл генерации значений
-----
0.721Bar    1500об/с    34.34C    2.59м/с
```

# Тестирование

## Характеристики смартфона Samsung J2:

<b>Операционная система</b>	Android 8.1	<b>Процессор</b>	MSM8917
Количество ядер	4	Частота	1.4 ГГц
Объем оперативной памяти	1.5 ГБ	Объем встроенной памяти	16 ГБ

**Функциональное тестирование** - проверка соответствия программного продукта функциональным требованиям, указанным в техническом задании на создание этого продукта.

**Полученный результат:** приложение удовлетворяет всем функциональным требованиям

**Стресс тестирование** – проверка приложения с целью выявить предел нормального функционирования.

**Полученный результат:** общее количество датчиков в системе существенного влияния на функционирование системы не оказывают. При этом работа приложения существенно замедляется при уменьшении интервала между сообщениями, отправляемыми с сервера. Оптимальный интервал между сообщениями 0.1 секунды.

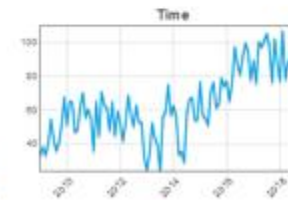
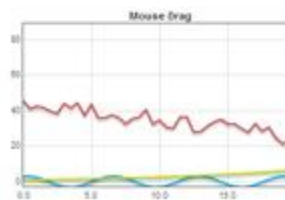
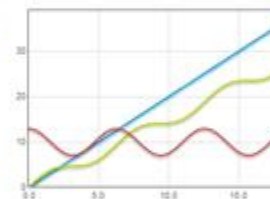
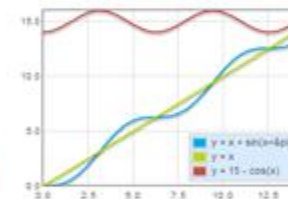
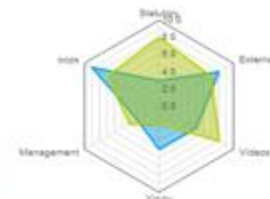
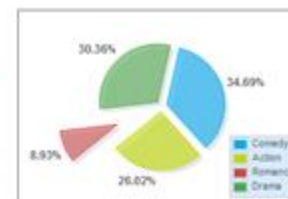
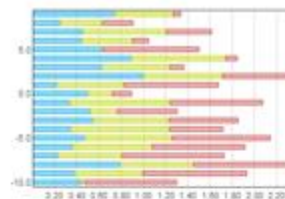
Допустимый: 0.01 секунды.



# Заключение

В результате проделанной работы было спроектировано и реализовано удобное и простое в использовании мобильное приложение для систематического обхода предприятия.

В случае дальнейшего развития приложения возможно добавление возможности считывать NFC-метки и Bluetooth-маяки существенно увеличит возможности приложения, а вывод статистики показаний в виде графиков положительно повлияет на восприятие данных.



Благодарю за  
внимание