МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

*Клиент-серверное Android приложения с использованием аппаратно-программного средства Arduino для чтения и записи ключей типа iButton*

*Дипломная работа*

09.03.02 *Информационные системы и технологии*

*Программная инженерия в информационных системах*

Допущен к защите

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_*С.Д. Махортов д.ф.- м.н., доцент* \_\_.\_\_.20\_\_

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*П.Н Парамонов, 4 курс*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.И. Чекмарев*

Воронеж 2021

**Содержание**

[Введение 3](#_Toc72930354)

[1. Постановка задачи 4](#_Toc72930355)

[2.1 Цель 4](#_Toc72930356)

[2.2 Сфера использования 4](#_Toc72930357)

[2.3 Требования 4](#_Toc72930358)

[2. Анализ предметной области 6](#_Toc72930359)

[2.1 Целевая аудитория 6](#_Toc72930360)

[2.2 Описание предметной области 6](#_Toc72930361)

[2.3 Анализ задачи 6](#_Toc72930362)

[2.3.1 Диаграмма вариантов использования 6](#_Toc72930363)

[2.3.2 Диаграмма классов 11](#_Toc72930364)

[2.3.3 Диаграмма состояний 12](#_Toc72930365)

[2.3.4 Диаграмма развертывания 13](#_Toc72930366)

[2.3.5 Схема Базы данных 14](#_Toc72930367)

[2.4 Архитектура приложения 18](#_Toc72930368)

[2.5 Архитектура аппаратно-программного средства Arduino 20](#_Toc72930369)

[3. Практическая часть 24](#_Toc72930370)

[3.1 Начальный экран 24](#_Toc72930371)

[3.2 Получение списка ключей из БД 25](#_Toc72930372)

[3.3 Добавление ключа в БД 27](#_Toc72930373)

[3.4 Выбор ключа 29](#_Toc72930374)

[3.5 Избранные ключи 30](#_Toc72930375)

[Заключение 33](#_Toc72930376)

[Список литературы 34](#_Toc72930377)

# Введение

Для людей, которые работают в сфере обслуживания, управления домов, такие как управляющие дома, ЖЭК, почтальоны, уборщики и другие, не составляет особой проблемой открыть ту или иную нужную им дверь подъезда, каких у них может быть не малое количество в закрепленных за ними. У них у всех есть связка универсальных ключей от дверей, но зачастую эта связка может быть настолько большой, что не всегда удобно будет найти нужный ключ от конкретной двери. Да и носить их с собой тоже не удобно.

Теперь давайте рассмотрим простых жителей домов. У каждого из них есть ключ от своего подъезда. А что если он его потеряет? Я думаю многие теряли ключи от домофона, и возможно ни раз. Тогда житель будет заказывать у мастера новый ключ, который обойдется около 200 рублей. Так же у жителей домов в собственности может быть ни один дом, а несколько.

Конечно же можно ввести код от домофона и открыть дверь. Но для каждой фирмы домофона, каких довольно немалое количество, свой код, да и далеко не все люди знают код даже от своего домофона. Поэтому появилась идея, о создании приложения на телефоне (базе Android), которое может скопировать и прошить ключ для домофона нужным кодом. Этот код может быть выбран как от нужного дома и подъезда, так и универсальный ключ от конкретного домофона. Для этого всего лишь надо небольшое Arduino устройство со считывателем ключа и телефон с установленным приложением. Тогда любой человек, сможет без труда сделать дубликат своего ключа от нужного дома, или сделать универсальный ключ.

# Постановка задачи

* 1. Цель

Разработать клиент-серверное приложение на базе OC Android с использованием аппаратно-программного средства Arduino для чтения записи ключей типа iButton, которое имеет такой функционал:

* Передать, достав из базы данных, код ключа Arduino
* Получить код ключа с помощью Arduino
* Добавить ключ в базу данных
* Получить информацию о городах, улицах, домах, подъездах и им принадлежащим ключам
* Получить информацию о домофонах и принадлежащим им универсальных ключах
* Добавить код ключа в избранное

Так же разработать само аппаратно-программное средство Arduino, которое может:

* Считать код поднесенного ключа к считывателю и отправить его приложению
* Записать в ключ, переданный через приложение код нужный код
  1. Сфера использования

Повседневная жизнь

* 1. Требования
* Возможность достать из базы данных код ключа и передать его Arduino
* Возможность получить код ключа, считав его с помощью Arduino
* Возможность добавления кода в базу данных
* Возможность получить информацию о городах, улицах, домах, подъездах и им принадлежащим ключам
* Возможность получить информацию о домофонах и принадлежащим им универсальных ключах
* Возможность добавить ключ в избранное

# Анализ предметной области

* 1. Целевая аудитория

Приложение подойдет для обычных жителей домов, для обслуживающего персонала дома, почтальонов, курьеров и т.д.

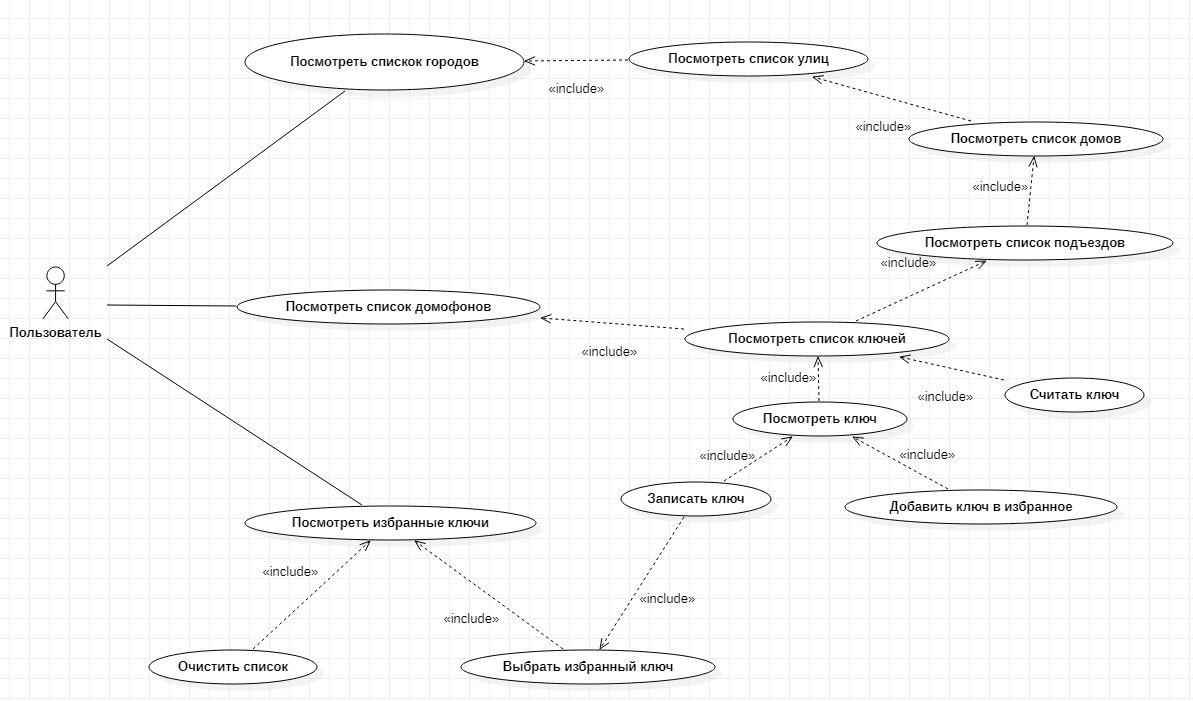
* 1. Описание предметной области

Как вообще работает домофон и ключ от него? В домофоне забит некий 64 битный - iButton код (может и 32 битный). В ключе точно так же забит какой-то код. Идентификатор состоит из восьми бит, которые отображаются в шестнадцатиричном виде: 01 XX XX XX XX XX XX YY. Здесь первый бит - это Family code, для ключей ibutton он всегда будет равен 1. Следующие шесть бит - это, собственно, уникальный идентификатор ключа. А восьмой бит - это, так называемый, "избыточный код" CRC или, другими словами, контрольная сумма, вычисляемая по специальному алгоритму из предыдущих семи бит. При поднесении ключа к домофону, последний считывает код с ключа, и если он совпадает с тем, что забит в домофоне, то дверь открывается. Этот самый код с оригинального ключа можно считать и записать в приложение, для последующей записи в новый ключ. Или же использовать для записи уже существующие универсальные ключи для конкретных домофонов.

Поэтому и возникла идея сделать приложение “DomofonMaster”, которая будет реализовывать данные задачи.

* 1. Анализ задачи
     1. Диаграмма вариантов использования

Ниже (рисунок 1) представлена диаграмма вариантов использования для данного проекта.

Актер в ней, только сам пользователь системы.

*Рис. 1. Диаграмма прецедентов*

Варианты использования:

* **Посмотреть список городов**

Описание: Позволяет пользователю просмотреть список доступных городов

Основной поток событий: Пользователь просматривает список городов

* **Посмотреть список улиц**

Описание: Позволяет пользователю просмотреть список доступных улиц

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список городов”

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города

* **Посмотреть список домов**

Описание: Позволяет пользователю просмотреть список доступных домов

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список улиц”

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города, а затем выбирает нужный дом улицы

* **Посмотреть список подъездов**

Описание: Позволяет пользователю просмотреть список доступных подъездов

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список домов**”**

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города, нужный дом, и в итоге нужный подъезд

* **Посмотреть список ключей**

Описание: Позволяет пользователю просмотреть список доступных ключей

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список подъездов**”**

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города, нужный дом, подъезд, и в конце список ключей данного подъезда

* **Считать ключ**

Описание: Позволяет пользователю считать ключ и добавить его в базу данных

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список ключей**”**

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города, нужный дом, подъезд, список ключей данного подъезда, и считывает ключ. Или до этого пункта можно добраться путем просмотра домофонов и просмотра ключей данного домофона

* **Посмотреть ключ**

Описание: Позволяет пользователю посмотреть нужный ему ключ

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть список ключей**”**

Основной поток событий: Пользователь просматривает список городов, затем список улиц выбранного города, нужный дом, подъезд, список ключей данного подъезда, и выбирает ключ. Или до этого пункта можно добраться путем просмотра домофонов и просмотра ключей данного домофона и выбор последнего

* **Посмотреть список домофонов**

Описание: Позволяет пользователю посмотреть список домофонов

Основной поток событий: Пользователь просматривает список домофонов.

* **Записать ключ**

Описание: Позволяет пользователю записать выбранный ключ

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть ключ**”**

Основной поток событий: Пользователь просматривает нужный ему ключ и записывает его

* **Добавить ключ в избранное**

Описание: Позволяет пользователю записать выбранный ключ

Предусловия: Во время варианта использования должно быть выполнено “Посмотреть ключ**”**

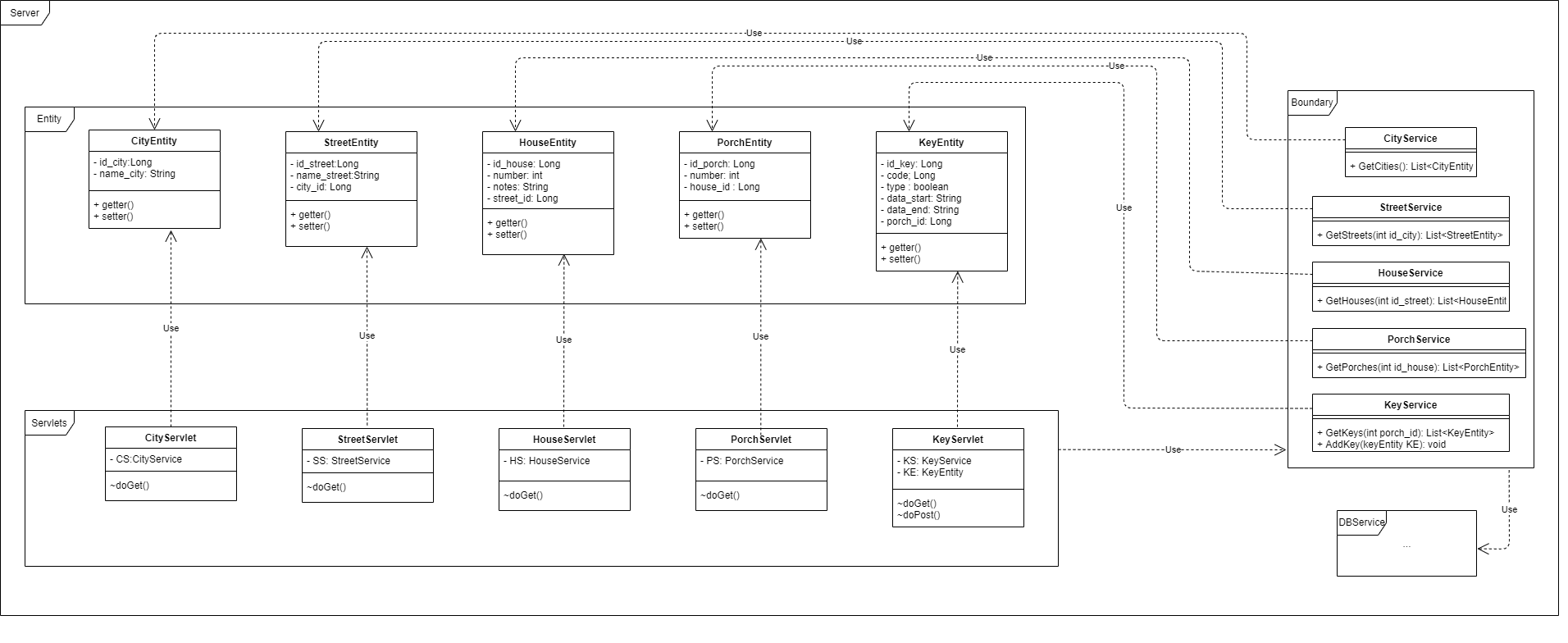
Основной поток событий: Пользователь добавляет просмотренный ключ в избранное

* **Просмотреть список избранных ключей**

Описание: Позволяет пользователю посмотреть избранные ключи

Основной поток событий: Пользователь просматривает список избранных ключей

* + 1. Диаграмма классов



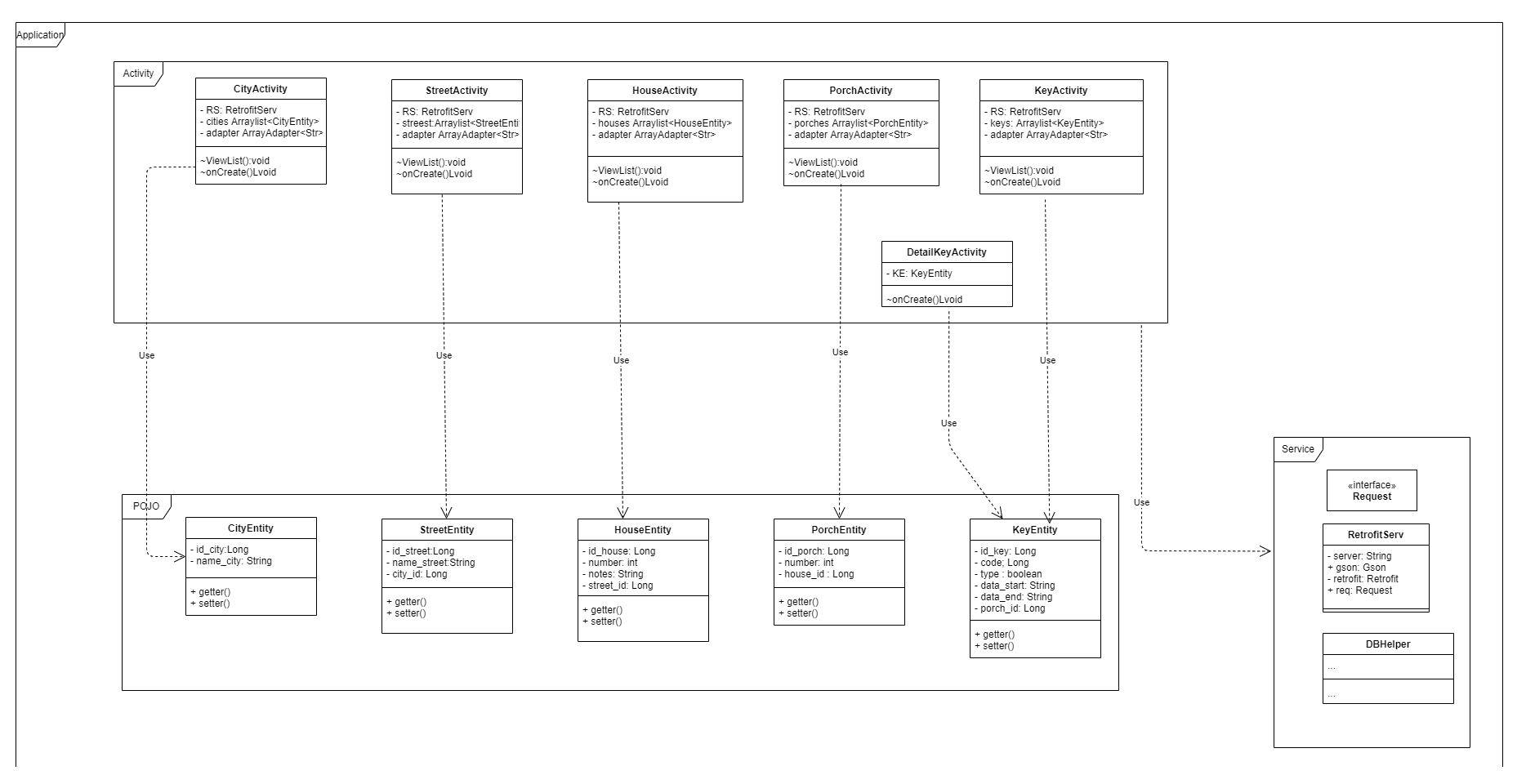
*Рис.2 Диаграмма классов сервера*

На рисунке 2 представлена диаграмма классов сервера:

В пакете Entity содержатся сущности модели базы данных

В пакете Servlet содержатся сервлеты для всех сущностей, который выполняют Get/Post запросы

В пакете Boundary содержатся классы сервиса управления, которые делают SQL запросы (с помощью пакета DBService) базе данных.



*Рис. 3 Диаграмма классов приложения*

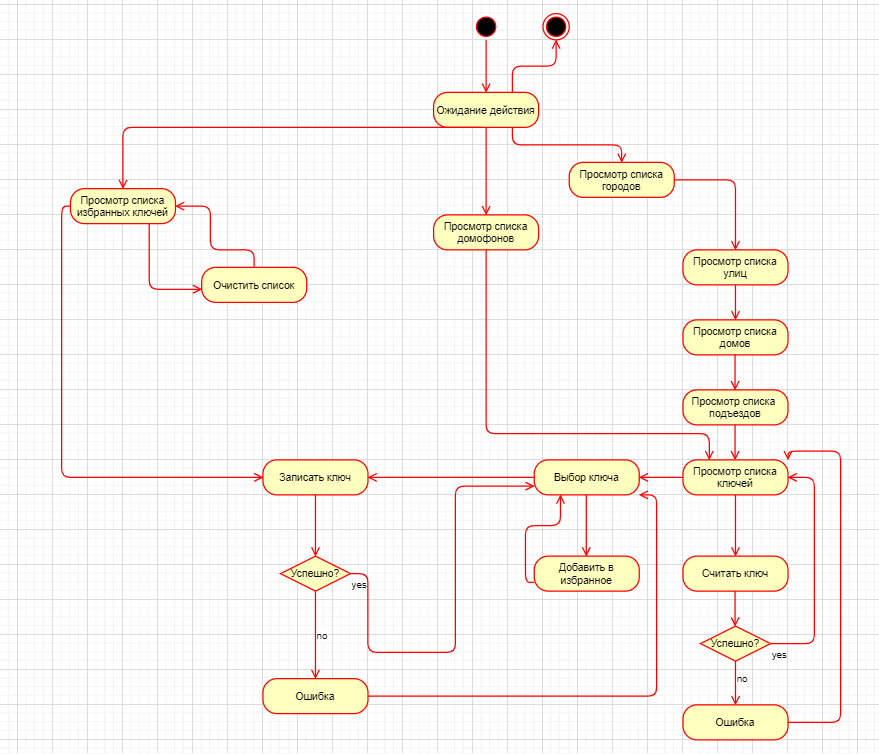
На рисунке 3 представлена диаграмма классов приложения:

В пакете POJO содержатся сущности модели базы данных

В пакете Activity содержатся активности для всех сущностей, который выполняют запросы на сервер и отображают разметку layout.xml

В пакете Service содержатся классы сервиса.

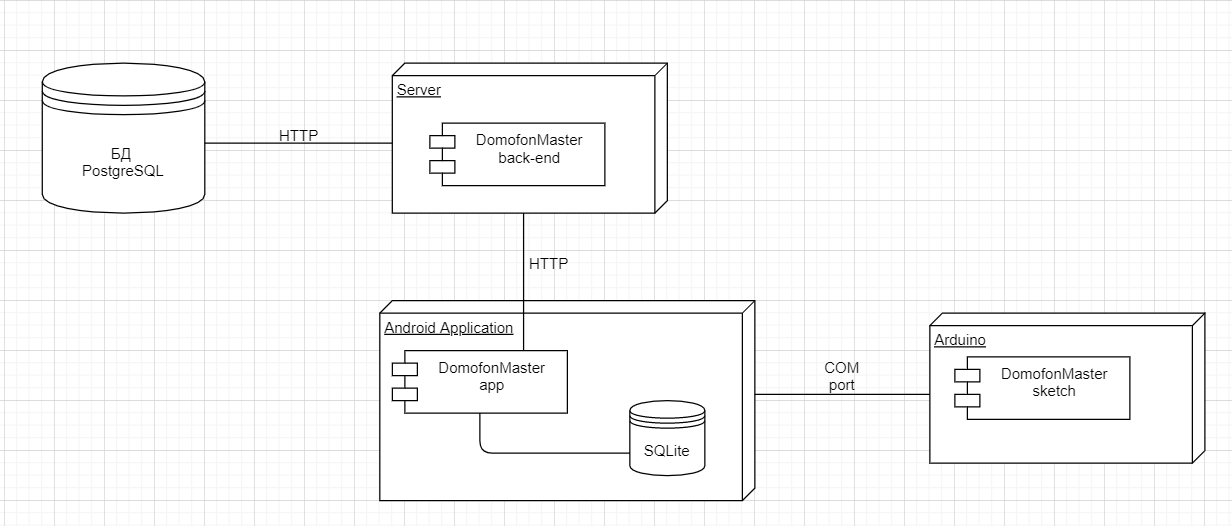
* + 1. Диаграмма состояний



*Рис. 4 Диаграмма состояний*

На рисунке 4 изображена диаграмма состояний, которая дает понять состояния программы во время действий пользователя и их последовательность.

* + 1. Диаграмма развертывания



*Рис. 5 Диаграмма развертывания*

На рисунке 5 изображена диаграмма развертывания приложения. По ней мы видим какие узлы существуют, какие компоненты в этих узлах и как все связано.

Отдельно развернута База Данных. На серверной части развернуто back-end приложение. В Android App узле развернуто само приложение, а также локальная база данных. Сервер и приложение взаимодействуют через HTTP протокол. В Arduino части прошит скетч для записи и чтения ключей, который взаимодействует с приложение по COM порту.

* + 1. Схема Базы данных

На рисунке 6 изображена схема базы данных. На ней мы видим таблицы:

* + - **City** - Город

Атрибуты:

* + - * + *id\_city* – PK
        + *Name* – название города. Тип String. NOT NULL
    - **Street** – улица

Атрибуты:

* + - * *id\_street* – PK
      * *city\_id* – FK ссылающийся на таблицу *City*
      * *Name –* название улицы. Тип String. NOT NULL
    - **House** – Дом

Атрибуты:

* + - * + *id\_house* – PK
        + *street\_id* – FK ссылающийся на таблицу *Street*
        + *Number* – номер дома. Тип Integer. NOT NULL
        + *Notes –* примечания. Тип String.
    - **Porch** – подъезд

Атрибуты:

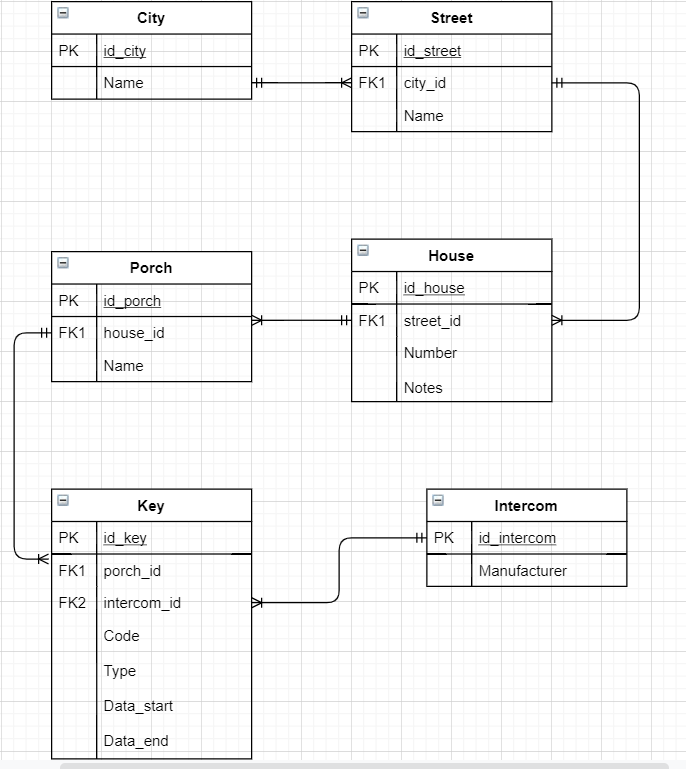
* + - * + *id\_porch* – PK
        + *house\_id* – FK ссылающийся на таблицу *House*
        + *Number* – номер подъезда. Тип Integer. NOT NULL
    - **Key** – ключ

Атрибуты:

* + - * + *id\_ke*y – PK
        + *porc*h*\_id* – FK ссылающийся на таблицу *Porch*
        + *intercom\_id* – FK ссылающийся на таблицу *Intercom*
        + *Code* – код ключа. Тип Long. NOT NULL
        + *Data\_start* – дата регистрации ключа. Тип String. NOT NULL
        + *Data\_end* – дата окончания действия ключа. Тип String.

Отношения между таблицами:

* + - City – Street. Отношение 1: m. В одном городе может быть много улиц. Но конкретная улица может быть только в одном городе.
    - Street – House. Отношение 1: m. В одной улице может быть много домов. Но конкретный дом может быть только на одной улице.
    - House – Porch. Отношение 1: m. В одном доме может быть много подъездов. Но конкретный подъезд может быть только в одном доме.
    - Porch – Key. Отношение 1: m. Для одного подъезда подходит несколько ключей. Но конкретный ключ подходит для одного подъезда.
    - Intercom – Key. Отношение 1: m. Для одного производителя домофона подходит несколько ключей. Но конкретный ключ подходит для одного домофона.



*Рис.6 Схема базы данных*

* 1. Архитектура приложения

В разработке приложения были выбраны такие основные технологии как:

* ЯП Java
* Jakarta EE (Java EE)
* Сервер Apache Tomcat 9
* Фреймворк Apache Maven
* Gradle
* Библиотека Retrofit
* БД PostgreSQL

И сам сервер, и приложение написаны на языке программирования Java. А именно сервер написан на Jakarta EE - набор спецификаций и соответствующей документации для языка Java, описывающей архитектуру серверной платформы. Jakarta EE во многом ориентирована на использование её через веб, как в интернете, так и в локальных сетях. Вся спецификация создаётся и утверждается через JCP (Java Community Process).

Tomcat - контейнер сервлетов с открытым исходным кодом. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat позволяет запускать веб-приложения. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server.

Apache Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML. Maven используется для построения и управления проектами, написанными на Java, C#, Ruby, Scala, и других языках.

С помощью Maven был собран back-end часть проекта. Были подтянуты такие библиотеки как:

* javax.servlet
* com.google.code.gson
* javax
* junit
* org.postgresql

Среди примечательных альтернатив Maven — система автоматической сборки Gradle.

Gradle — система автоматической сборки, построенная на принципах Apache Ant и Apache Maven, но предоставляющая DSL на языках Groovy и Kotlin вместо традиционной XML-образной формы представления конфигурации проекта. Gradle был разработан для расширяемых многопроектных сборок, и поддерживает инкрементальные сборки, определяя, какие компоненты дерева сборки не изменились и какие задачи, зависимые от этих частей, не требуют перезапуска. Gradle собирал само Android приложение.

Для реализации клиент-серверных приложений, коммуницирующих по протоколу HTTP, удобно использовать специально разработанные для таких целей пакеты классов. Одной из наиболее популярных библиотек является Retrofit. К ее достоинствам относят:

* Нет нужды делать запросы к HTTP API в отдельном потоке
* Сокращается длина кода и, соответственно, ускоряется разработка
* Возможно подключение стандартных пакетов для конвертации JSON в объекты и обратно (например, пакета Gson)
* Динамическое построение запросов
* Обработка ошибок
* Упрощенная передача файлов

Логика работы библиотеки основывается на аннотациях. Благодаря их использованию можно описывать динамические запросы на сервер.

Для описания запросов к серверу необходимо объявить интерфейс, который впоследствии будет использоваться при генерации запросов. Перед каждым методом интерфейса должна стоять аннотация, основываясь на которой, Retrofit определяет, какого типа запрос обрабатывается данным методом. Также с помощью аннотаций можно указывать параметры запроса.

В качестве СУБД была выбрана PostgreSQL в силу открытого доступа и высокой производительности.  PostgreSQL можно программно расширить за счёт хранимых процедур.

С клиентской стороны остается лишь иметь Android устройство не ниже Android 6 Marshmallow.

Общение между back-end и application происходит методом REST API. Обмен данными происходит путем передачи их в файлах JSON.

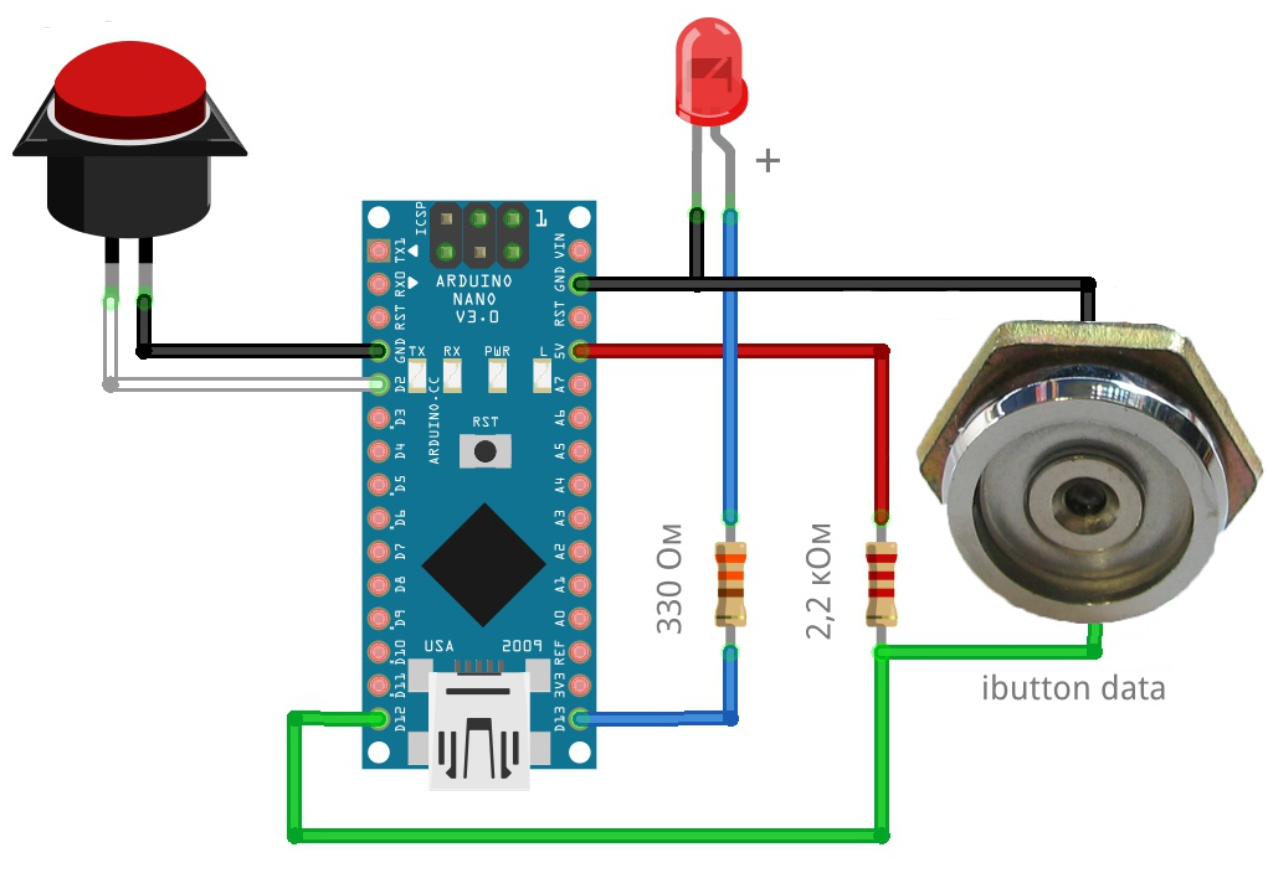
* 1. Архитектура аппаратно-программного средства Arduino

Данное устройство было собранно на Arduino Nano. Подключение с телефоном происходит через USB-кабель.

В схеме присутствуют такие элементы:

* + Arduino Nano
  + Кнопка
  + RGB светодиод
  + Два резистора с номинальным сопротивлением 2,2 кОм и 330 Ом
  + Считыватель ключа типа iButton

Схема изображена на рисунке 7.



*Рис.7 Схема устройства*

Линия данных iButton обязательно подтягивается к питанию (+5В) через резистор 2,2 кОм. Светодиод подключается через ограничительный резистор подходящего номинала, в данном случае 330 Ом. Так как в скетче используем вход кнопки (D2) со встроенным подтягивающим (PULLUP) резистором, то кнопку устанавливаем на GND.

Возможности устройства:

* Чтение ID ключа с последующей записью в перезаписываемую "болванку" (RW1990);
* Защита от случайной записи некорректного значения ID;
* Переключение режимов работы записи/чтения по нажатию кнопки

Работа с устройством:

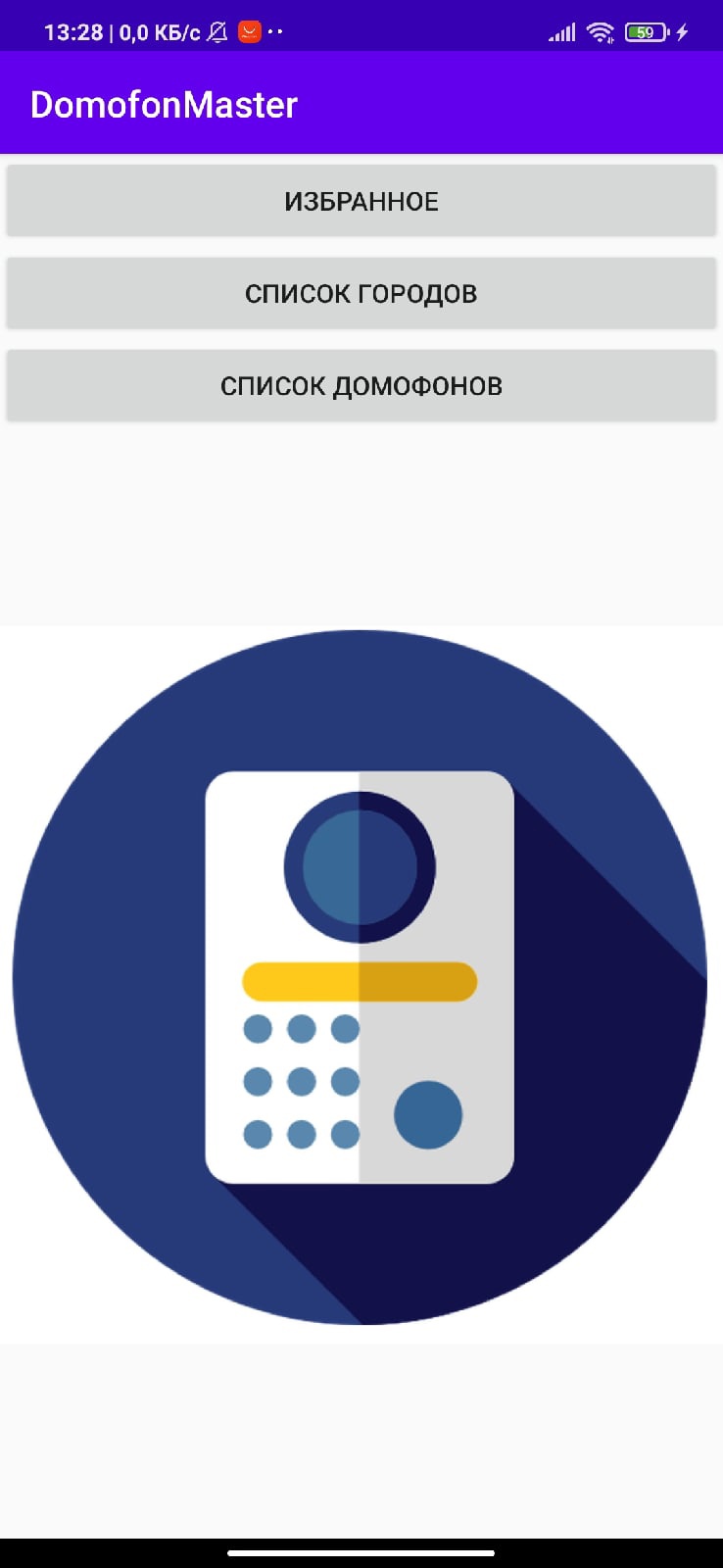
Подключаем устройство к телефону (при текущей реализации к компьютеру). Светодиод несколько раз мигает в процессе загрузки. Через пару секунд устройство готово к работе, светодиод при этом гаснет.

Чтобы считать ID ключа, прикладываем ключ к контактной площадке считывателя. Светодиод при этом начинает часто моргать, а в терминале отображается считанный ID (при текущей реализации), который сохраняется в буфере до тех пор, пока в неё не будет загружен другой ID.

Чтобы записать ID в перезаписываемый ключ, надо нажать кнопку на устройстве. При этом зажигается светодиод, что говорит о готовности устройства к записи. Прикладываем записываемый ключ к контактной площадке считывателя: светодиод при этом гаснет, а примерно через секунду начинает часто моргать, что говорит о завершении процесса записи и переключении устойства обратно в режим чтения. Если записываемый ID был некорректный (с неверным Family code, или CRC), в терминал выведется соответствующее сообщение и запись будет отменена. Таким образом, устройство предохраняняет ключ от записи в него некорректных данных. Тем не менее, всё же может случиться так, что данные запишутся с ошибками. Такое может произойти, например, если ключ будет недостаточно плотно приложен к контактной площадке при записи данных.

# Практическая часть

* 1. Начальный экран



*Рис.8 Начальный экран*

На рисунке 8 изображен начальный экран приложения. Здесь можно выполнить 3 действия:

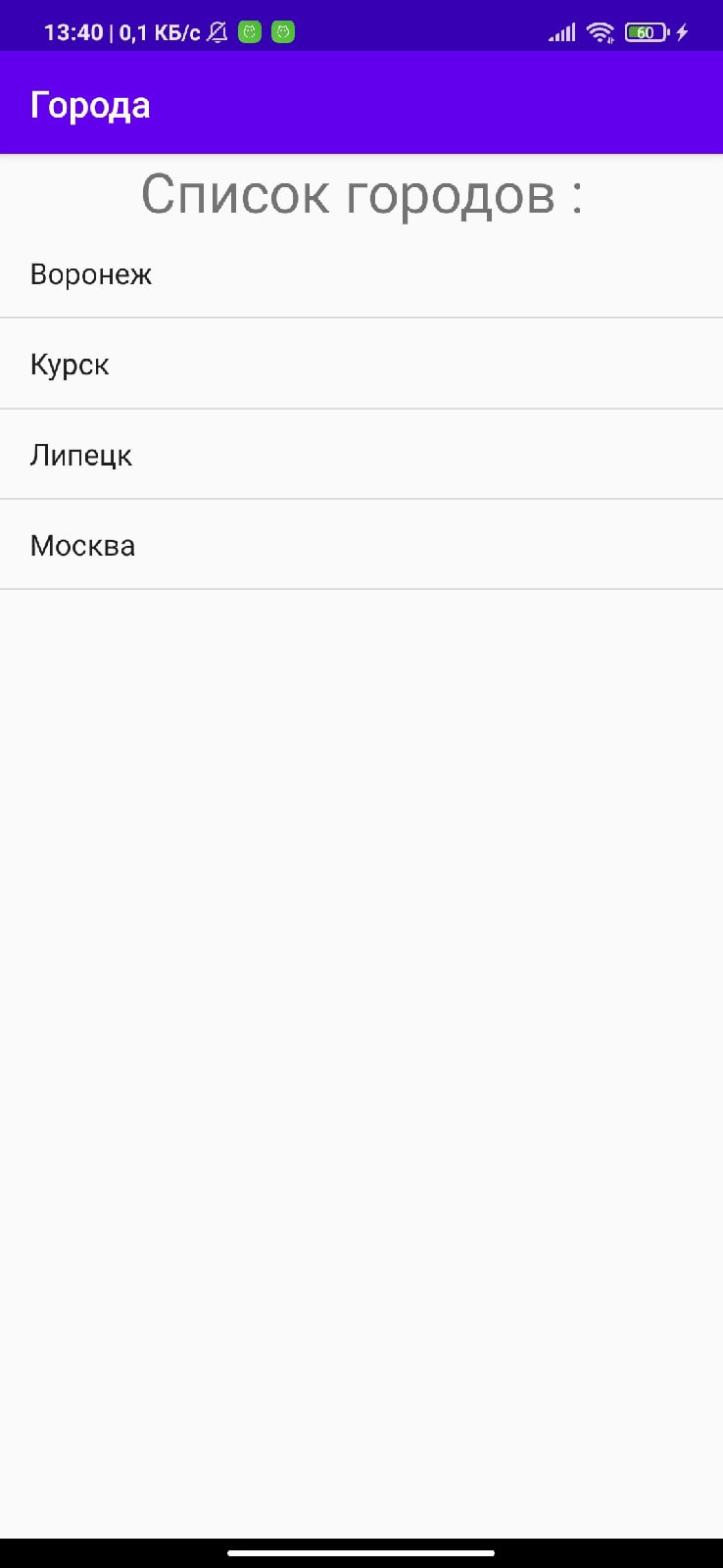
* Посмотреть список избранных ключей
* Посмотреть список городов
* Посмотреть список домофонов
  1. Получение списка ключей из БД

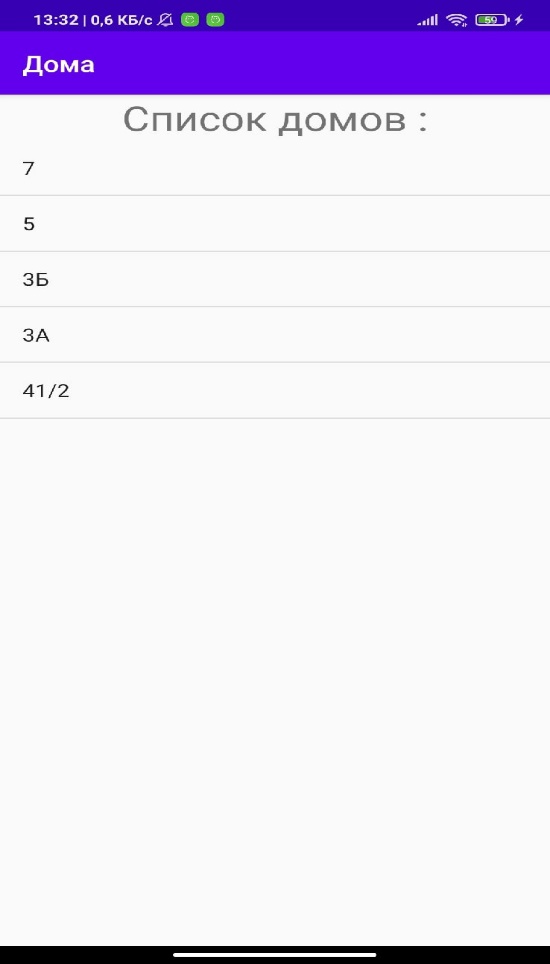
На следующих рисунках (9-13) мы выбираем нужный нам их списка город, затем улицу этого города, затем нужный нам дом, и его подъезд.

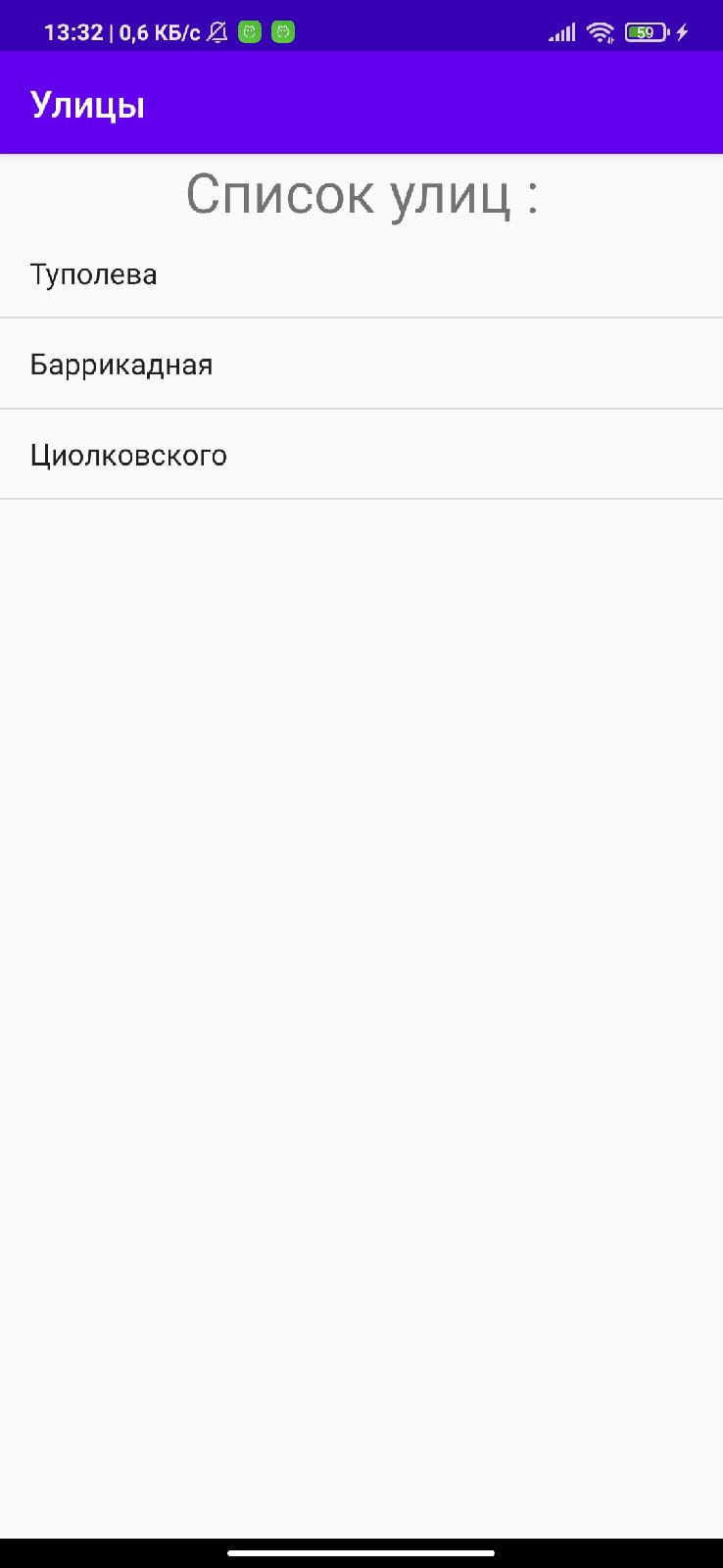
Итак, выберем следующие пункты. Воронеж -> Туполева -> 7 ->1

Для города, улицы, дома и подъезда соответственно.

*Рис. 9 Список городов*







*Рис. 10 Список улиц Рис. 11 Список домов*

**



*Рис. 13 Список подъездов Рис. 14 Список ключей*

Вот мы получили список доступных ключей для 1-его подъезда дома “7” по улице Туполева города Воронежа.

* 1. Добавление ключа в БД

Вернемся и выберем пустой подъезд без ключей. В базе данных на данный момент нет ключей (рисунок 15).



*Рис.15 Отсутствие ключей*

Нажмем кнопку “Добавить ключ”. Далее Arduino считывает код поднесенного ключа и передает его приложению, и последнее отправляет данные о ключе на сервер, который добавляет ключ в БД. И приложение выдают соответствующее сообщение (рисунок 16). В данном случае код берется рандомный. Далее обновим страницу и увидим, что для второго подъезда появился ключ, который мы только что добавили (рисунок 17).





*Рис 16 Добавление ключа Рис. 17 Добавленный ключ*

* 1. Выбор ключа

Выберем текущий ключ из второго подъезда. И появится экран, где можно прочитать этот ключ, нажав на кнопку “Прочитать ключ”, тогда мы передаем Arduino данных ключ и можем открыть дверь (рисунок 18).

Вторая функция – это добавление ключа в избранное. При нажатии на кнопку “Добавить в избранное”, данный ключ добавляется в избранные ключи пользователя (рисунок 19).



*Рис. 18 Считывание кода Рис. 19 Добавление ключа в избранное*

* 1. Избранные ключи

Вернемся на начальный экран и выберем “Избранные”. Мы увидим только что добавленный ключ (рисунок 20). Выберем этот ключ, и мы сможем считать его точно так же, как и в предыдущий раз (рисунок 21).

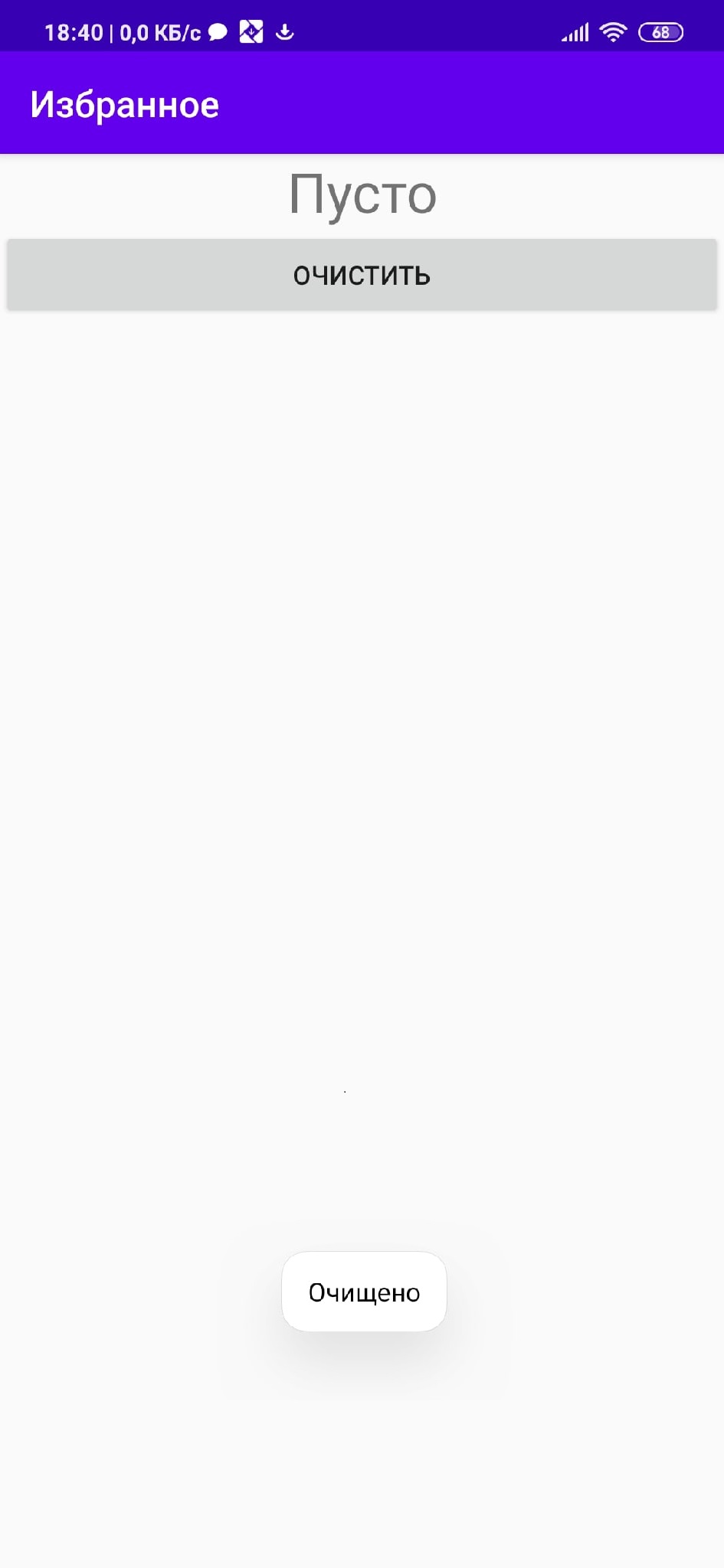


*Рис. 20 Избранные ключи*



*Рис. 21 Считывание избранного ключа*

Далее вернемся в список избранных ключей, и очистим его, нажав на соответствующую кнопку. Список очистится и высветится соответствующее сообщение об этом (рисунок 22).



*Рис. 22 Очистка списка избранного*

# Заключение

Данное приложение реализовано практически полностью, с учетом той архитектуры, которая была указана, за исключением реализации связи Arduino и Android приложения для передачи между ними данных.

Были выполнены следующие поставленные цели и требований:

* Разработана back-end часть приложения. Сервер, развернутый на локальном компьютере
* Разработано android приложение, находящееся на Android устройстве
* Была создана связь между android приложением и сервером с помощью REST
* Разработана база данных
* Разработано аппаратно-программное средство Arduino

# Список литературы

1. Помощь в решениях возникающих проблем: сайт – URL:

<https://stackoverflow.com> (дата обращения: 28.03.2021-20.05.2021). – Текст: электронный

1. Понимание как вернуть JSON из Java сервлета: сайт – URL:

<https://fooobar.com/questions/4711/how-do-you-return-a-json-object-from-a-java-servlet> (дата обращения: 14.04.2021). – Текст: электронный

1. Android справочник: сайт – URL:

<https://myitacademy.ru/edu/course/view.php> (дата обращения: 28.03.2021-25.04.2021).- Текст: электронный

1. Android адаптеры: сайт – URL:

<https://metanit.com/java/android/5.1.php> (дата обращения: 21.04.2021). – Текст: электронный

1. Клиент серверные приложение: сайт – URL:

<http://blog.harrix.org/article/7209> (дата обращения: 02.04.2021). – Текст: электронный

1. Справочник Retrofit: сайт – URL:

<https://square.github.io/retrofit/> (дата обращения: 15.04.2021). – Текст: электронный

1. Справочник Retrofit русскоязычная версия: сайт – URL:

<http://developer.alexanderklimov.ru/android/library/retrofit.php> (дата обращения: 15.04.2021). – Текст: электронный

1. Работа с базой данный SQLite на Android: сайт – URL:

<http://www.fandroid.info/urok-34-rabota-s-bazami-dannyh-sqlite-v-android/> (дата обращения: 23.04.2021). – Текст: электронный

1. Дубликатор домофонных ключей на Arduino: сайт – URL: [http://osboy.ru/blog/microcontrollers/ibutton-dublicator.html?jnac407e82=1#comments](http://osboy.ru/blog/microcontrollers/ibutton-dublicator.html?jnac407e82=1%23comments) (дата обращения: 25.05.2021). – Текст: электронный
2. Дубликатор домофонных ключей на Arduino Nano: сайт – URL: <https://4pda.to/forum/index.php?showtopic=953401> (дата обращения: 25.05.2021). – Текст: электронный