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 домофона*

*Выпускная квалификационная 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, группа 2.2*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.И. Чекмарев, ст. преподаватель*

Воронеж 2021

**Содержание**

[Введение 3](#_Toc74192609)

[1. Постановка задачи 4](#_Toc74192610)

[2.1 Цель 4](#_Toc74192611)

[2.2 Сфера использования 4](#_Toc74192612)

[2.3 Требования 4](#_Toc74192613)

[2. Анализ предметной области 6](#_Toc74192614)

[2.1 Целевая аудитория 6](#_Toc74192615)

[2.2 Описание предметной области 6](#_Toc74192616)

[2.3 Анализ задачи 7](#_Toc74192617)

[2.3.1 Схема Базы данных 7](#_Toc74192618)

[2.3.2 Диаграмма вариантов использования 10](#_Toc74192619)

[2.3.3 Диаграмма последовательности 14](#_Toc74192620)

[2.3.4 Диаграмма IDEF3 22](#_Toc74192621)

[2.3.5 Диаграмма пакетов 27](#_Toc74192622)

[2.3.6 Диаграмма классов 29](#_Toc74192632)

[2.3.7 Диаграмма состояний 34](#_Toc74192633)

[2.3.8 Диаграмма развертывания 34](#_Toc74192634)

[2.4 Архитектура приложения 35](#_Toc74192635)

[2.5 Архитектура аппаратно-программного средства Arduino 38](#_Toc74192636)

[3. Практическая часть 41](#_Toc74192637)

[3.1 Подключение Arduino к телефону 41](#_Toc74192638)

[3.2 Начальный экран 41](#_Toc74192639)

[3.3 Получение списка ключей из БД 43](#_Toc74192640)

[3.4 Добавление ключа в БД 45](#_Toc74192641)

[3.5 Запись ключа 47](#_Toc74192642)

[3.6 Избранные ключи 49](#_Toc74192643)

[Заключение 52](#_Toc74192644)

[Список использованных источников 53](#_Toc74192645)

# 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
  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. Схема Базы данных

На рисунке 1 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:

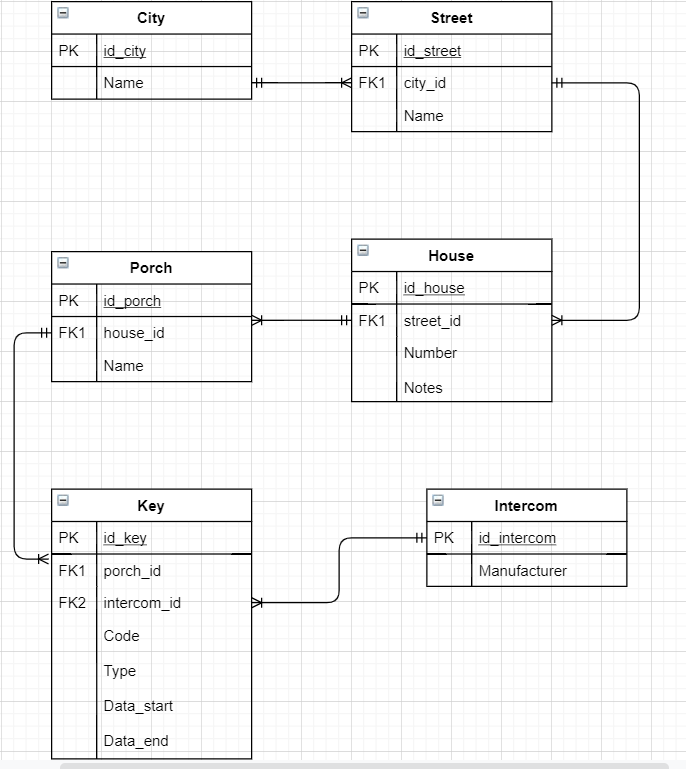
* + - * + id\_porch – PK
        + house\_id – FK ссылающийся на таблицу House
        + Number – номер подъезда. Тип Integer. NOT NULL
    - **Key** – ключ

Атрибуты:

* + - * + id\_key – PK
        + porch\_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.

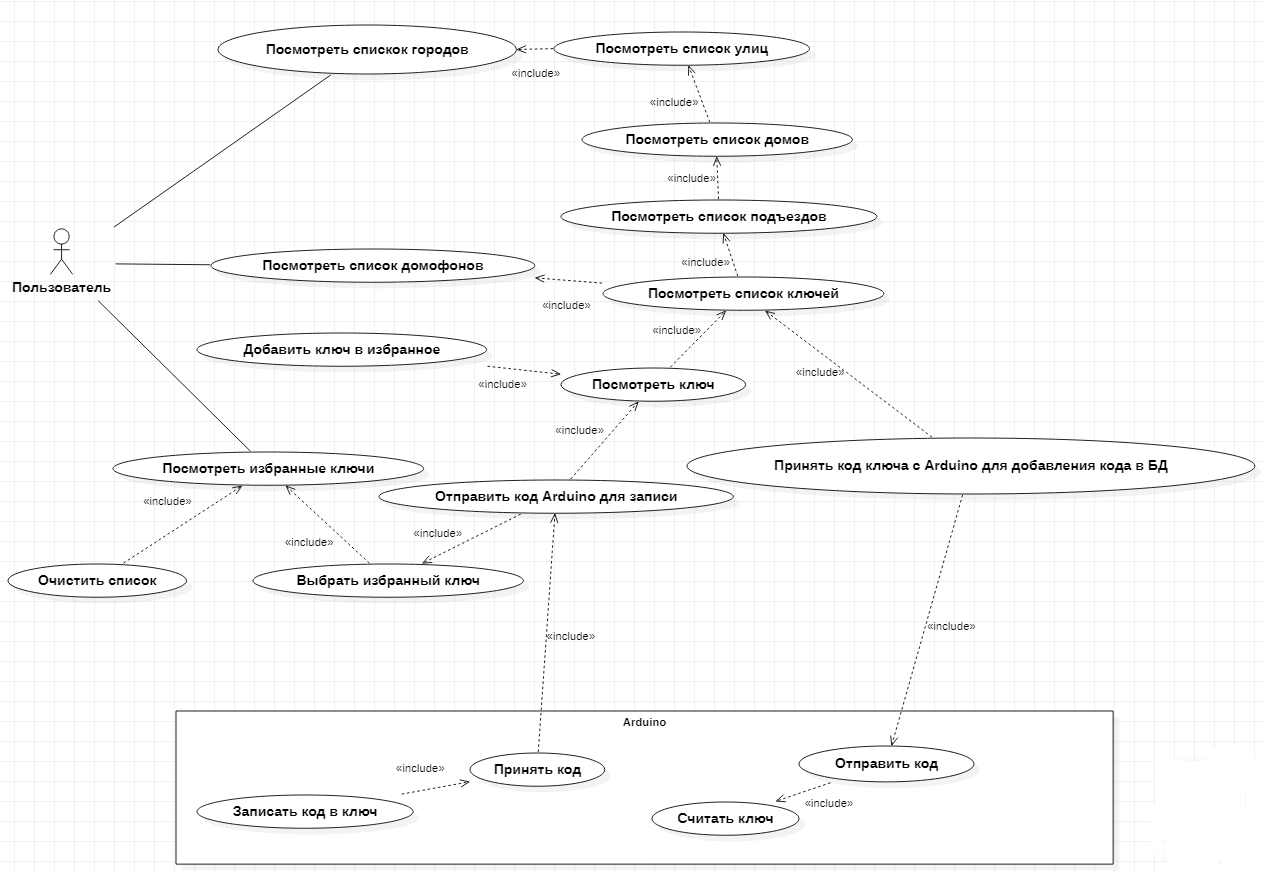


*Рис.1 Схема базы данных*

* + 1. Диаграмма вариантов использования

Ниже (рисунок 2) представлена диаграмма вариантов использования для данного проекта.

Актор в ней, только сам пользователь системы.



*Рис. 2. 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

**Принять код ключа с Arduino для добавления кода в БД**

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.

**Отправить код Arduino для записи**

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 код в ключ

* **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

* **Очистить список**

Описание: Позволяет пользователю очистить список избранных ключей

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

Основной поток событий: Пользователь просматривает список избранных ключей, а затем очищает его

* **Считать ключ**

Описание: Arduino считывает ключ, который подносит пользователь

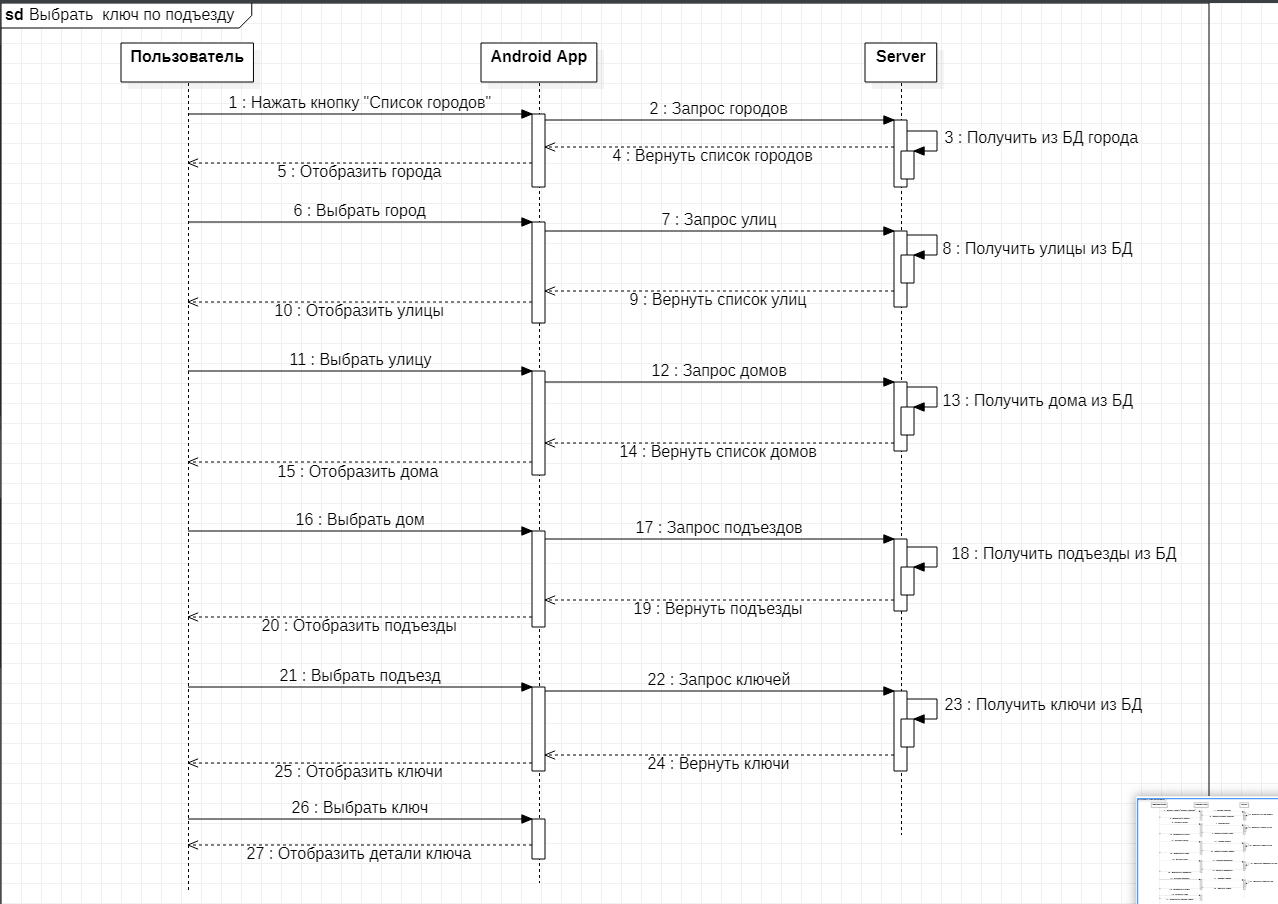
* **Записать код в ключ**

Описание: Arduino записывает код в ключ, который подносит пользователь

* + 1. Диаграмма последовательности

Далее представлены диаграммы последовательности для вышеуказанных вариантов использования:

* **Выбрать ключ по подъезду**



*Рис.3 Диаграмма последовательности “Выбрать ключ по подъезду”*

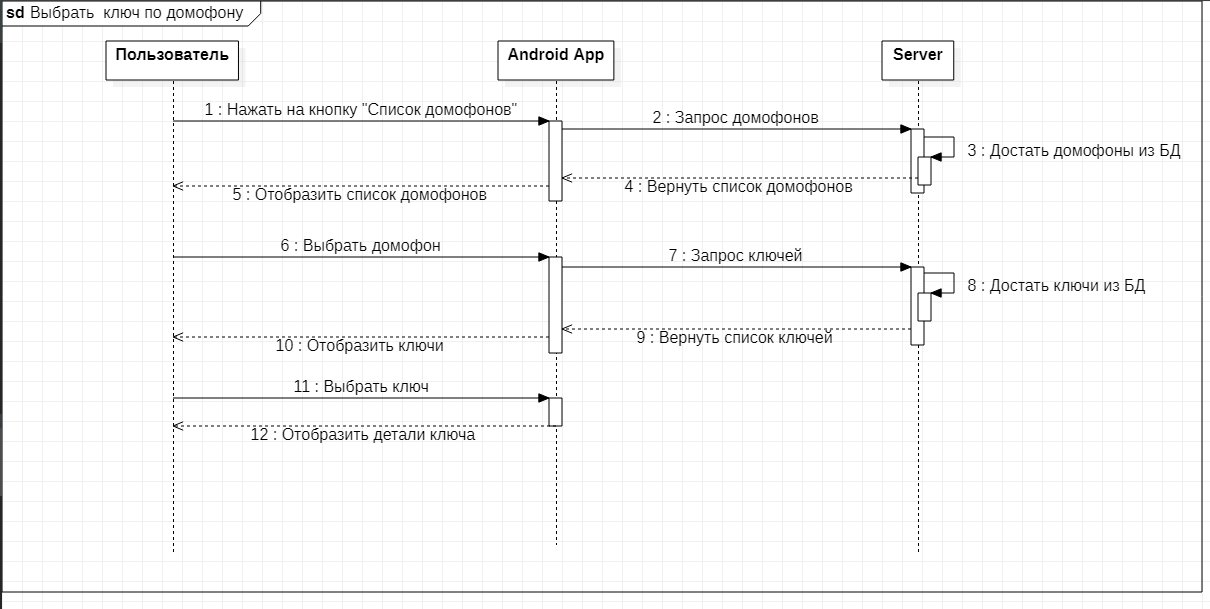
Объекты:

* Пользователь
* Android App
* Server

Сообщения между объектами:

1. Нажать кнопку "Список городов"
2. Запрос городов
3. Получить из БД города
4. Вернуть список городов
5. Отобразить города
6. Выбрать город
7. Запрос улиц
8. Получить улицы из БД
9. Вернуть список улиц
10. Отобразить улицы
11. Выбрать улицу
12. Запрос домов
13. Получить дома из БД
14. Вернуть список домов
15. Отобразить дома
16. Выбрать дом
17. Запрос подъездов
18. Получить подъезды из БД
19. Вернуть подъезды
20. Отобразить подъезды
21. Выбрать подъезд
22. Запрос ключей
23. Получить ключи из БД
24. Вернуть ключи
25. Выбрать ключ
26. Отобразить детали ключа

* **Выбрать ключ по домофону**



*Рис.4 Диаграмма последовательности “Выбрать ключ по домофону”*

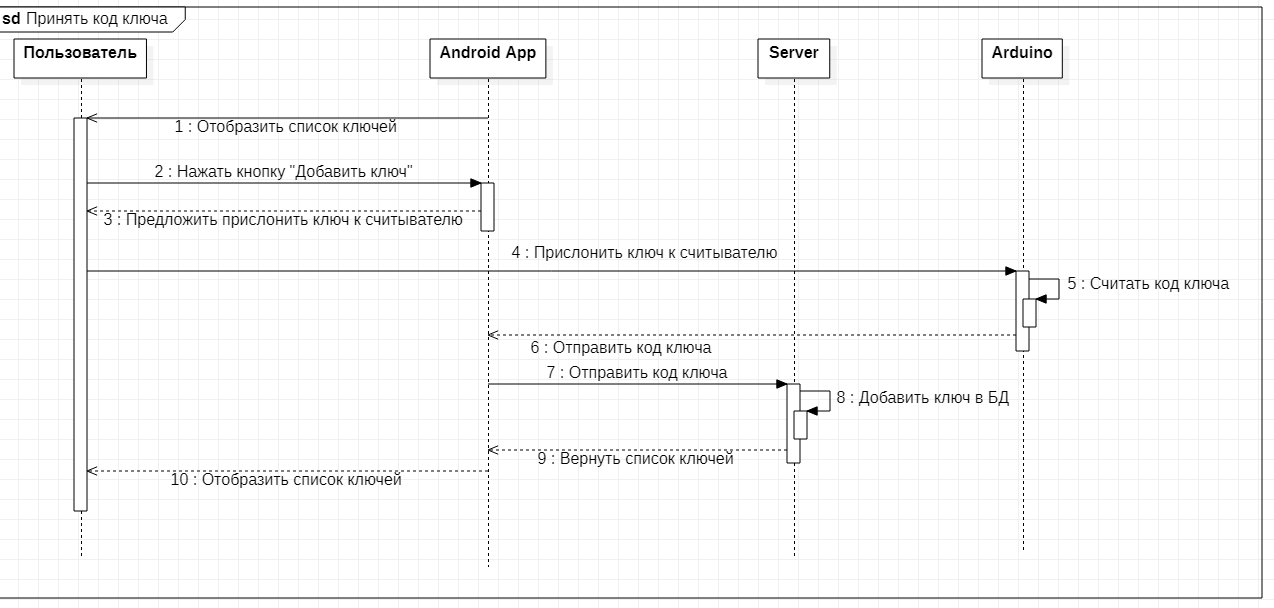
Объекты:

* Пользователь
* Android App
* Server

Сообщения между объектами:

1. Нажать на кнопку "Список домофонов"
2. Запрос домофонов
3. Достать домофоны из БД
4. Вернуть список домофонов
5. Отобразить список домофонов
6. Выбрать домофон
7. Запрос ключей
8. Достать ключи из БД
9. Вернуть список ключей
10. Отобразить ключи
11. Выбрать ключ
12. Отобразить детали ключа

* **Принять код ключа**



*Рис.5 Диаграмма последовательности “* *Принять код ключа”*

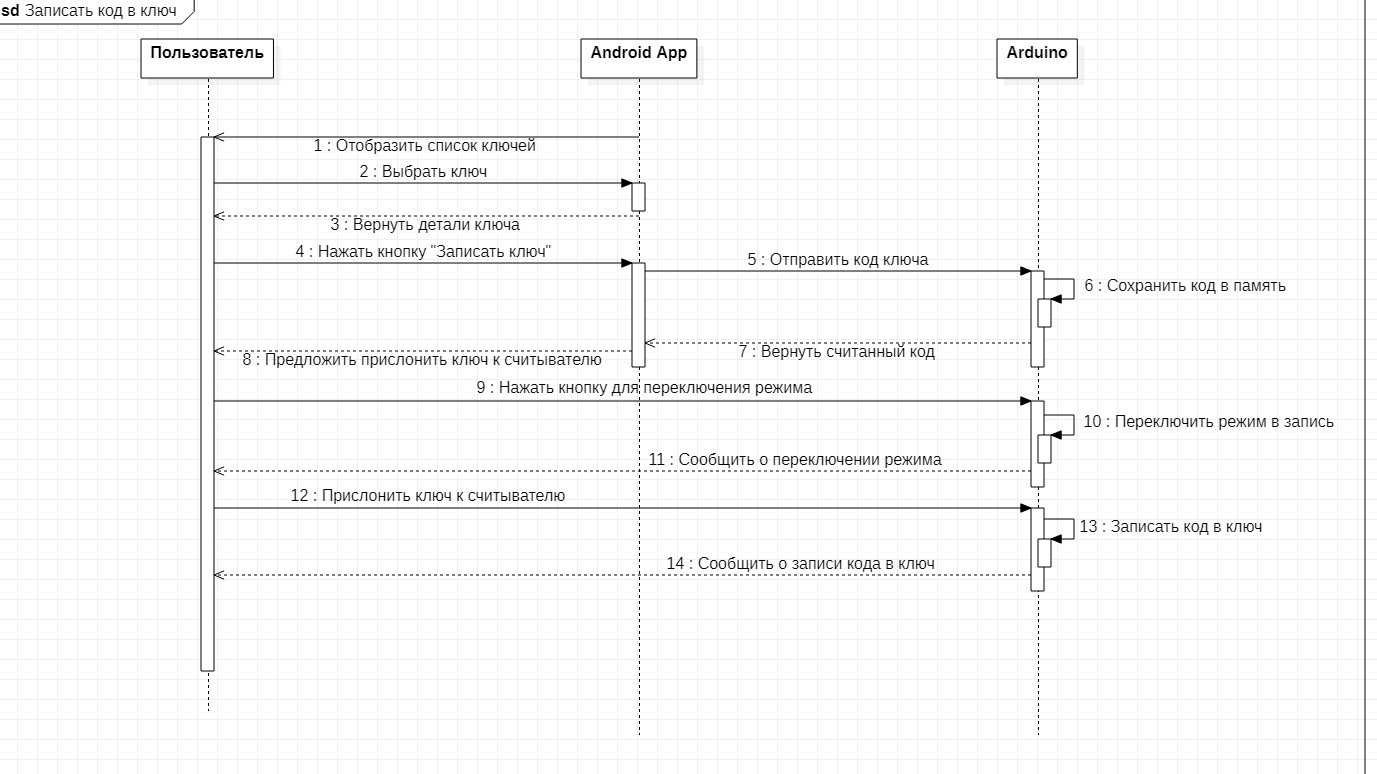
Объекты:

* Пользователь
* Android App
* Server
* Arduino

Сообщения между объектами:

1. Отобразить список ключей
2. Нажать кнопку "Добавить ключ"
3. Предложить прислонить ключ к считывателю
4. Прислонить ключ к считывателю
5. Считать код ключа
6. Отправить код ключа
7. Отправить код ключа
8. Добавить ключ в БД
9. Вернуть список ключей
10. Отобразить список ключей

* **Записать код в ключ**



*Рис.6 Диаграмма последовательности “Записать код в ключ”*

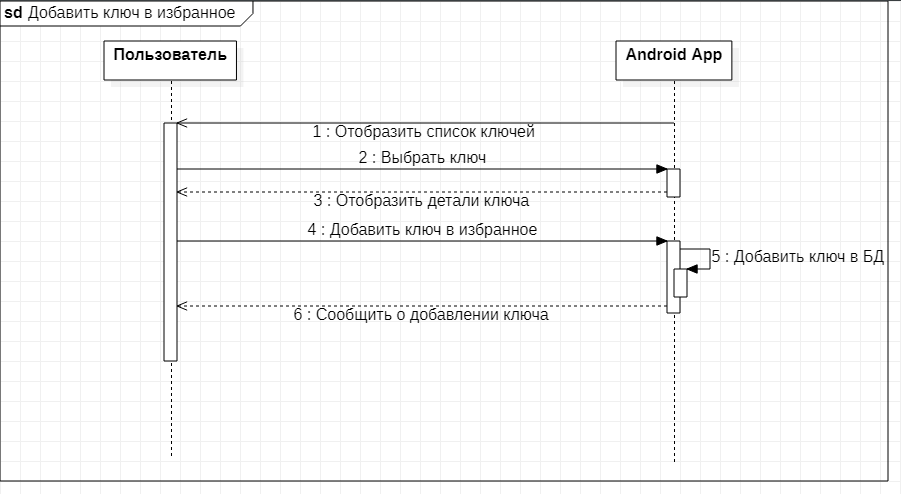
Объекты:

* Пользователь
* Android App
* Arduino

Сообщения между объектами:

1. Отобразить список ключей
2. Выбрать ключ
3. Вернуть детали ключа
4. Нажать кнопку "Записать ключ"
5. Отправить код ключа
6. Сохранить код в память
7. Вернуть считанный код
8. Предложить прислонить ключ к считывателю
9. Нажать кнопку для переключения режима
10. Переключить режим в запись
11. Сообщить о переключении режима
12. Прислонить ключ к считывателю
13. Записать код в ключ
14. Сообщить о записи кода в ключ

* **Добавить ключ в избранное**



*Рис.7 Диаграмма последовательности “Добавить ключ в избранное”*

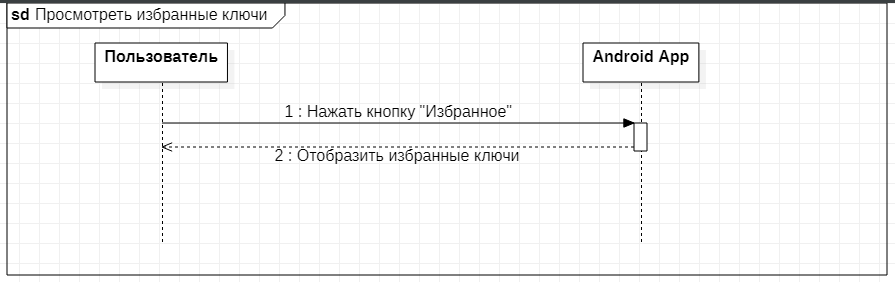
Объекты:

* Пользователь
* Android App

Сообщения между объектами:

1. Отобразить список ключей
2. Выбрать ключ
3. Отобразить детали ключа
4. Добавить ключ в избранное
5. Добавить ключ в БД
6. Сообщить о добавлении ключа

* **Просмотреть избранные ключи**



*Рис.8 Диаграмма последовательности “Просмотреть избранные ключи”*

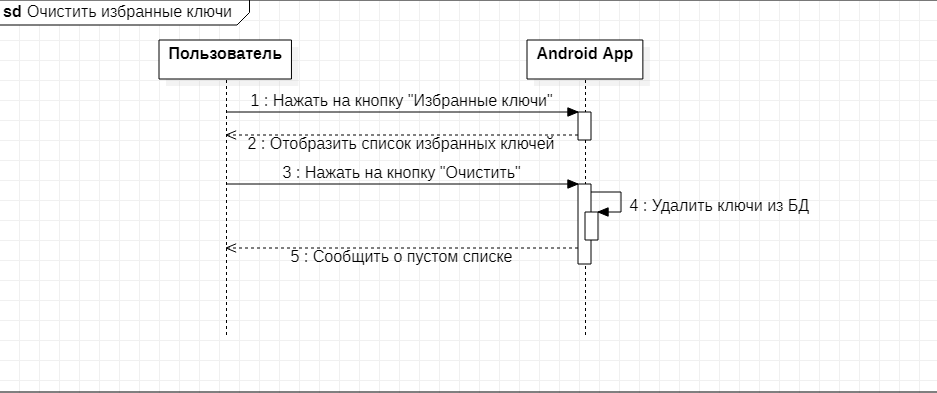
Объекты:

* Пользователь
* Android App

Сообщения между объектами:

1. Нажать кнопку "Избранное"
2. Отобразить избранные ключи

* **Очистить избранные ключи**



*Рис.9 Диаграмма последовательности “Очистить избранные ключи”*

Объекты:

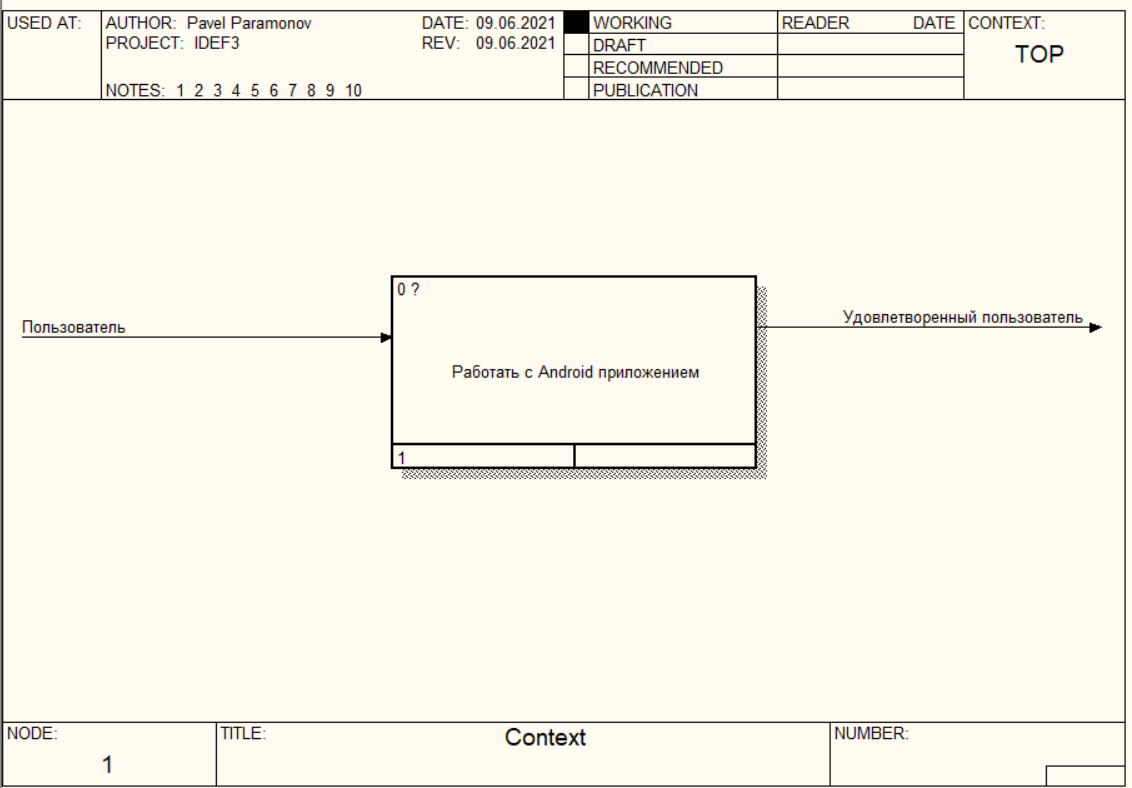
* Пользователь
* Android App

Сообщения между объектами:

1. Нажать на кнопку "Избранные ключи"
2. Отобразить список избранных ключей
3. Нажать на кнопку "Очистить"
4. Удалить ключи из БД
5. Сообщить о пустом списке
   * 1. Диаграмма IDEF3

Существуют два типа диаграмм в стандарте IDEF3, представляющие описание одного и того же сценария процесса в разных ракурсах: диаграмма Описания Последовательности Этапов Процесса - PFDD и диаграмма Состояния Объекта в Процессе Трансформации - OSTN. В данном случае для описания работы системы в целом была разработана диаграмма PFDD.

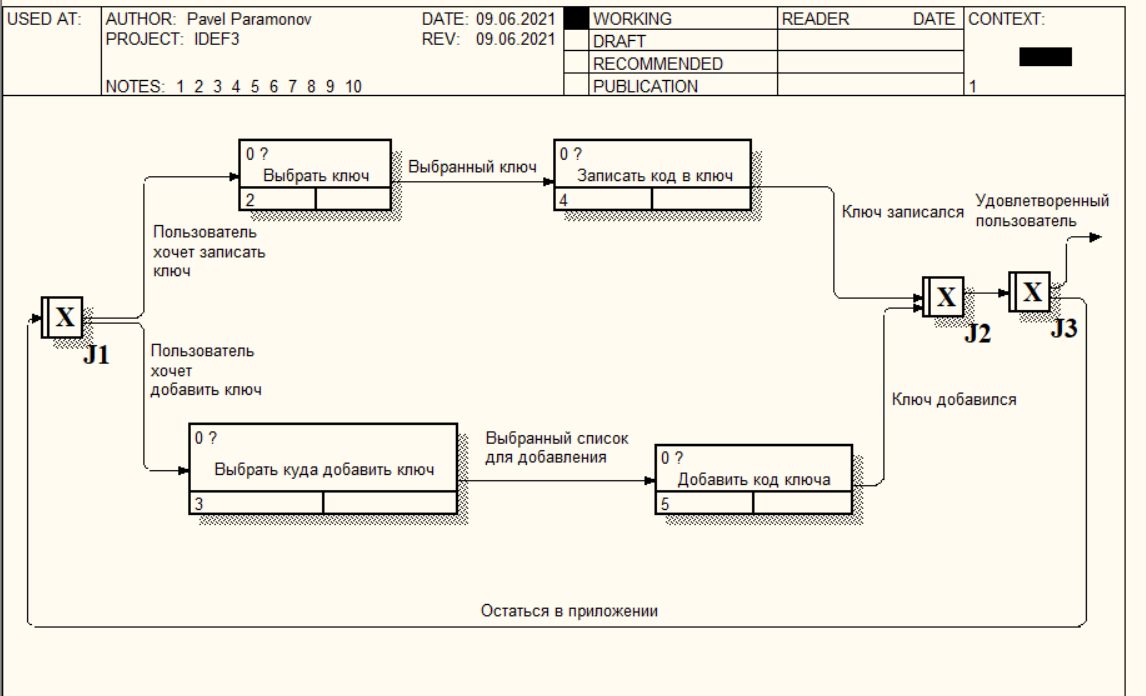
Описание работы системы на верхнем уровне показана на рисунке 10.



*Рис.10 Диаграмма IDEF3 верхний уровень*

Данная диаграмма является наиболее общим представлением. Как следует из диаграммы, пользователь взаимодействует с приложением. В результате взаимодействия пользователь выполняет свои нужды и становится “удовлетворенным”.

Первый уровень декомпозиции: «Работать с Android приложением»

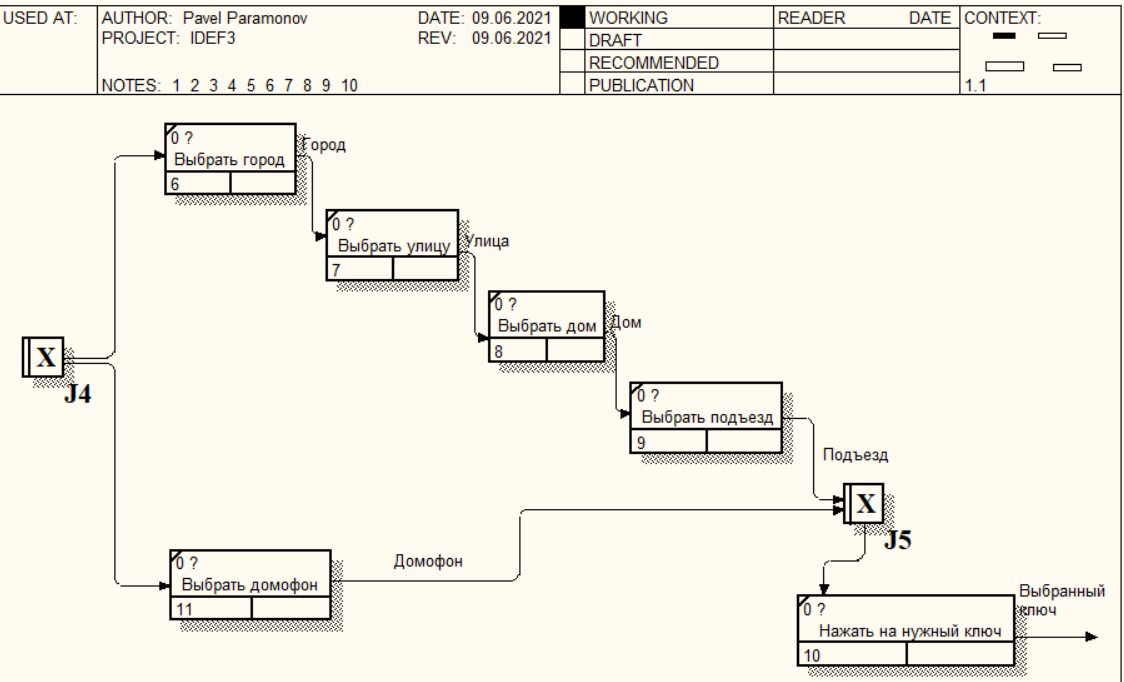


*Рис.11 Диаграмма IDEF3 первый уровень декомпозиции*

Данная диаграмма показывает декомпозицию диаграммы верхнего уровня.

На ней видно, что пользователь или выбирает нужный ему ключ, если он хочет записать его, затем записывает код в ключ, на выходе получаем записанный ключ. Или, если пользователь хочет добавить ключ, то он сначала выбирает куда нужно добавить ключ (подъезд или домофон), и добавляет код ключа в список. Затем, после того, как одно из событий выполнилось, пользователь либо продолжает использовать приложение, либо заканчивает с ним работу.

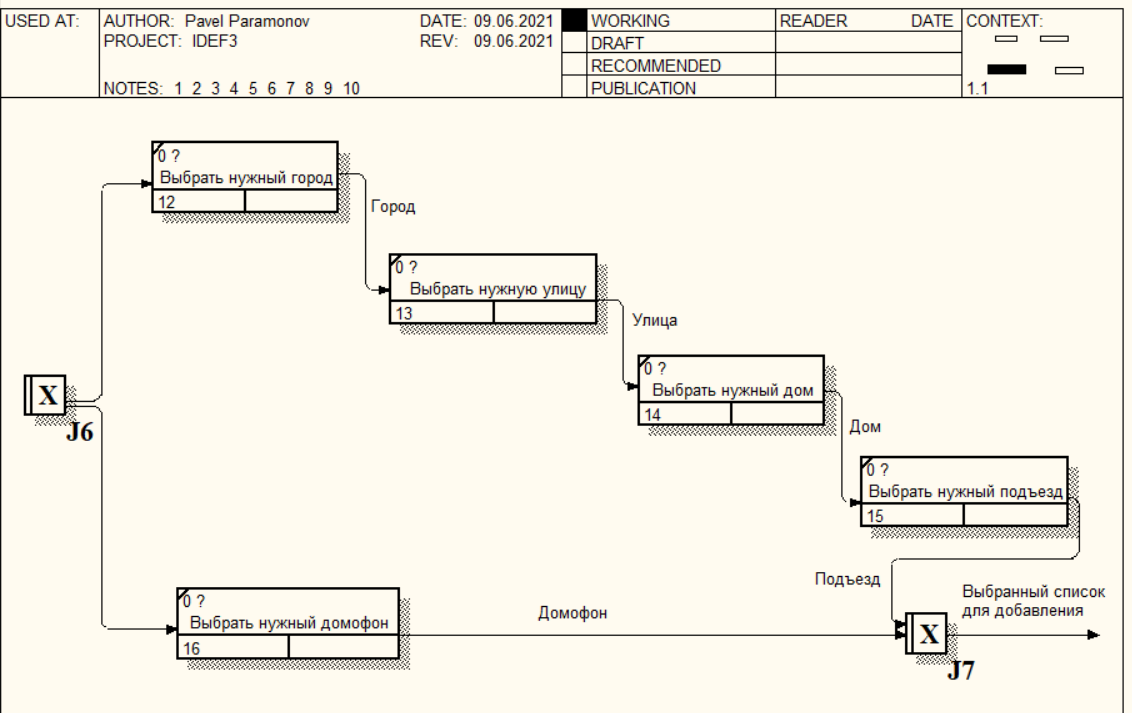
Второй уровень декомпозиции: «Выбрать ключ»



*Рис.12 Диаграмма IDEF3 второй уровень декомпозиции “Выбрать ключ”*

Пользователь либо выбирает нужный ему ключ из списка домофонов, либо из следующей последовательности Город -> Улица -> Дом -> Подъезд -> нужный ключ.

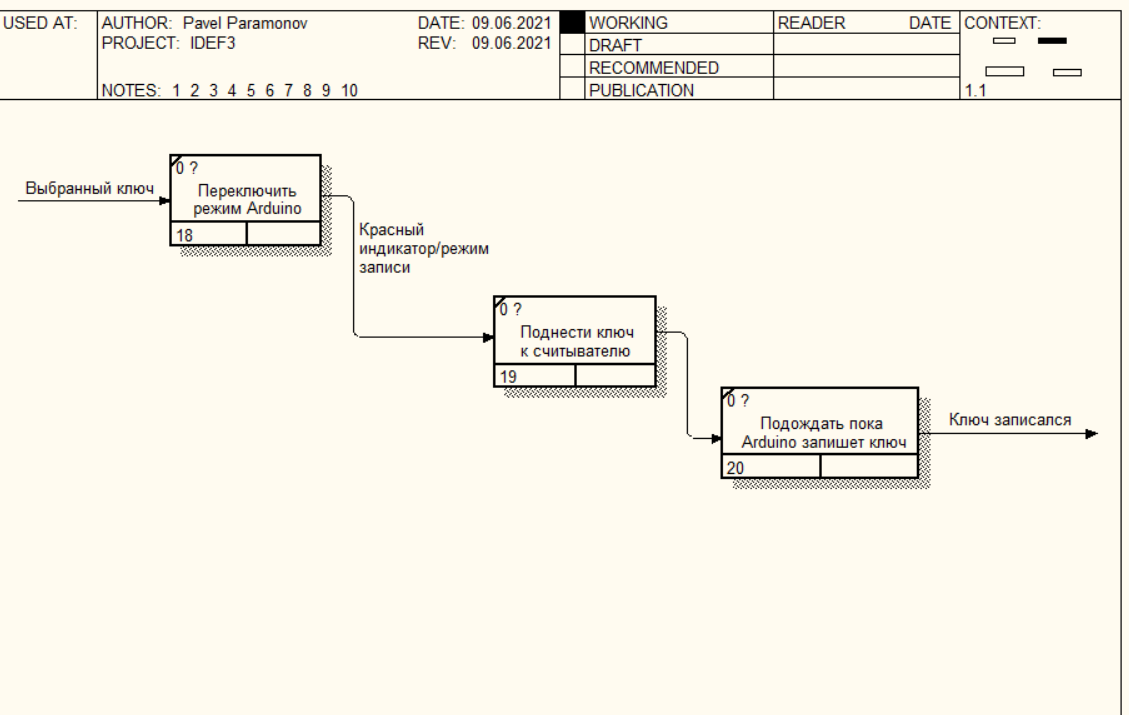
Второй уровень декомпозиции: «Выбрать куда добавить ключ»



*Рис.13 Диаграмма IDEF3 второй уровень декомпозиции “Выбрать куда добавить ключ”*

Пользователь либо выбирает нужный домофон куда добавить ключ, либо из следующей последовательности Город -> Улица -> Дом -> Подъезд, последний и будет тот список, куда добавится ключ.

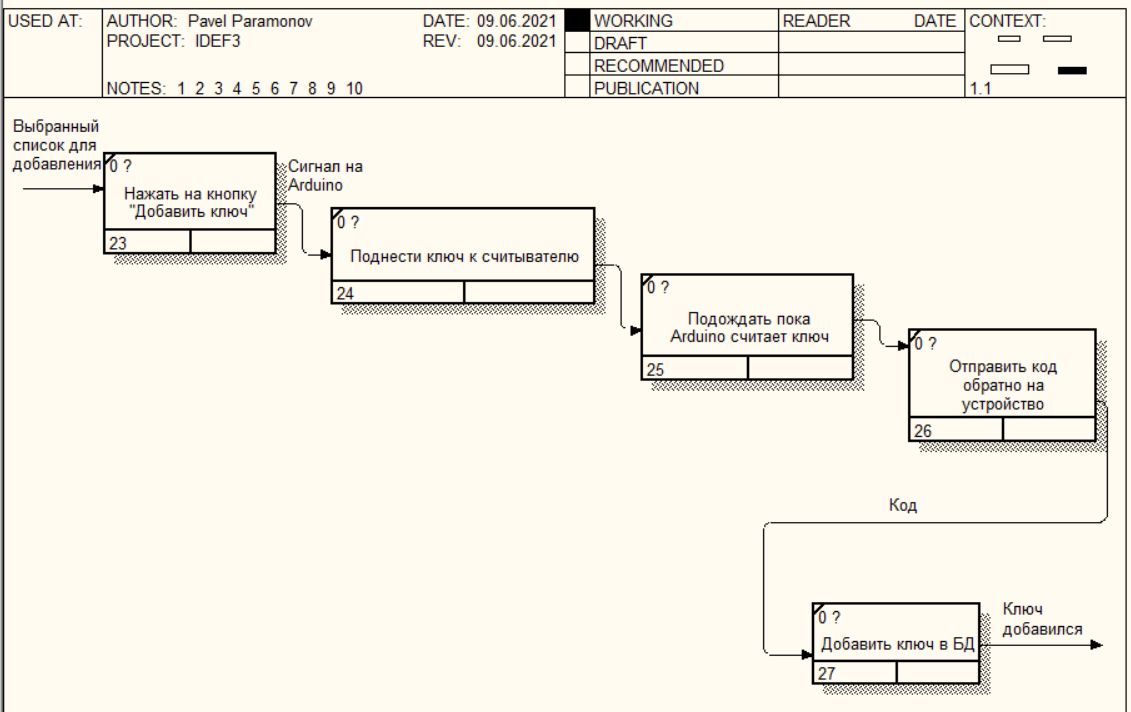
Второй уровень декомпозиции: «Записать код в ключ»



*Рис.14 Диаграмма IDEF3 второй уровень декомпозиции “Записать код в ключ”*

После того как пользователь выбрал нужный для записи ключ и передал его Arduino, он переключает режим Arduino для записи, подносит ключ к считывателю и немного ждет, пока Arduino запишет ключ. На выходе получаем записанный ключ.

Второй уровень декомпозиции: «Добавить код ключа»

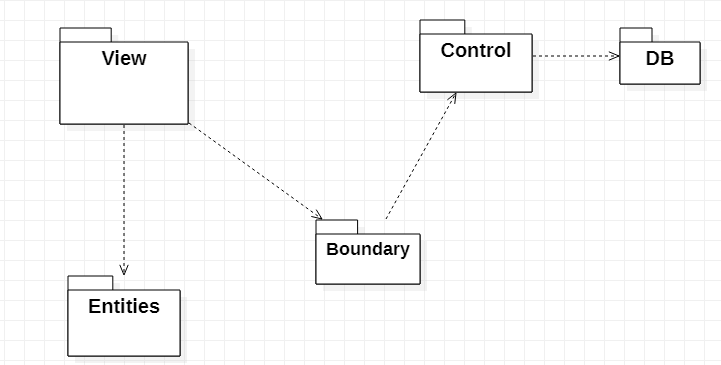


*Рис.15 Диаграмма IDEF3 второй уровень декомпозиции “Добавить код ключа”*

После того, как пользователь выбрал куда добавить ключ, он нажимает кнопку “Добавить ключ”, подносит ключ к считывателю, ждет пока Arduino считает ключ, затем Arduino отправляет код обратно на устройство, и приложение добавляет полученный код в БД.

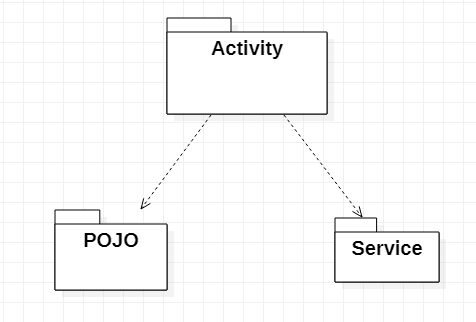
* + 1. Диаграмма пакетов

Ниже на рисунке 16 представлена диаграмма пакетов для серверной части.



*Рис.16 Диаграмма пакетов серверной части*

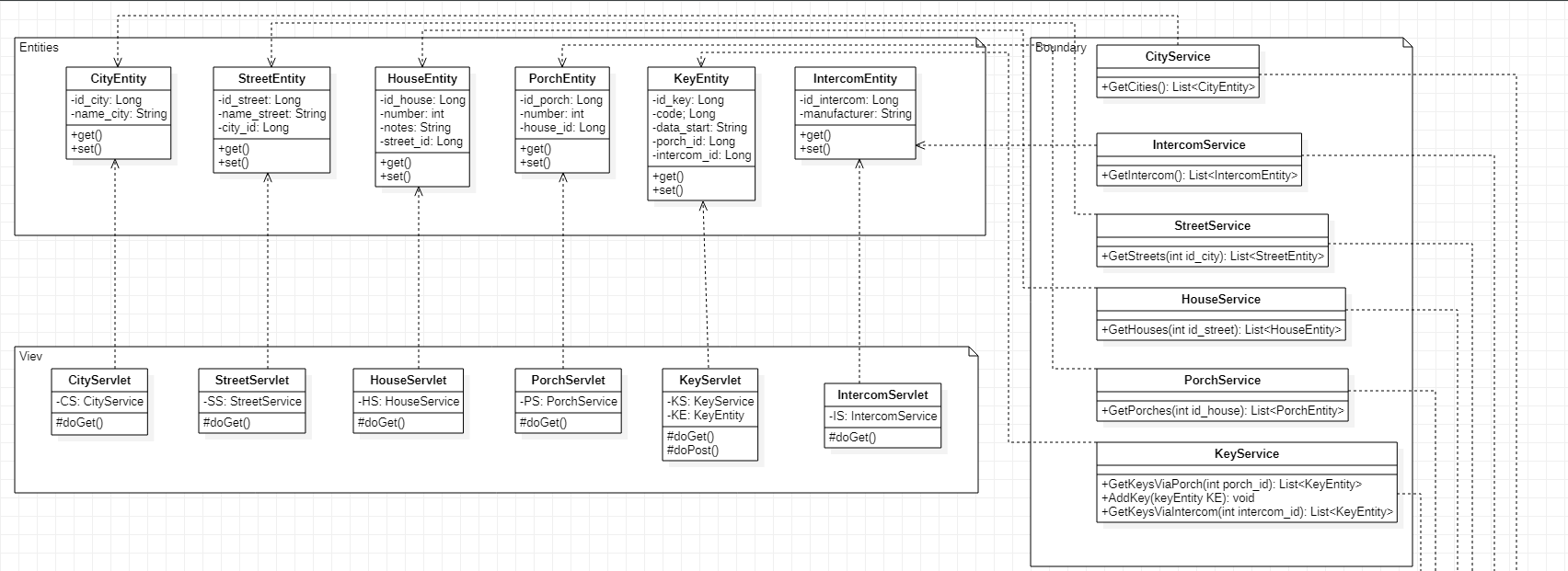
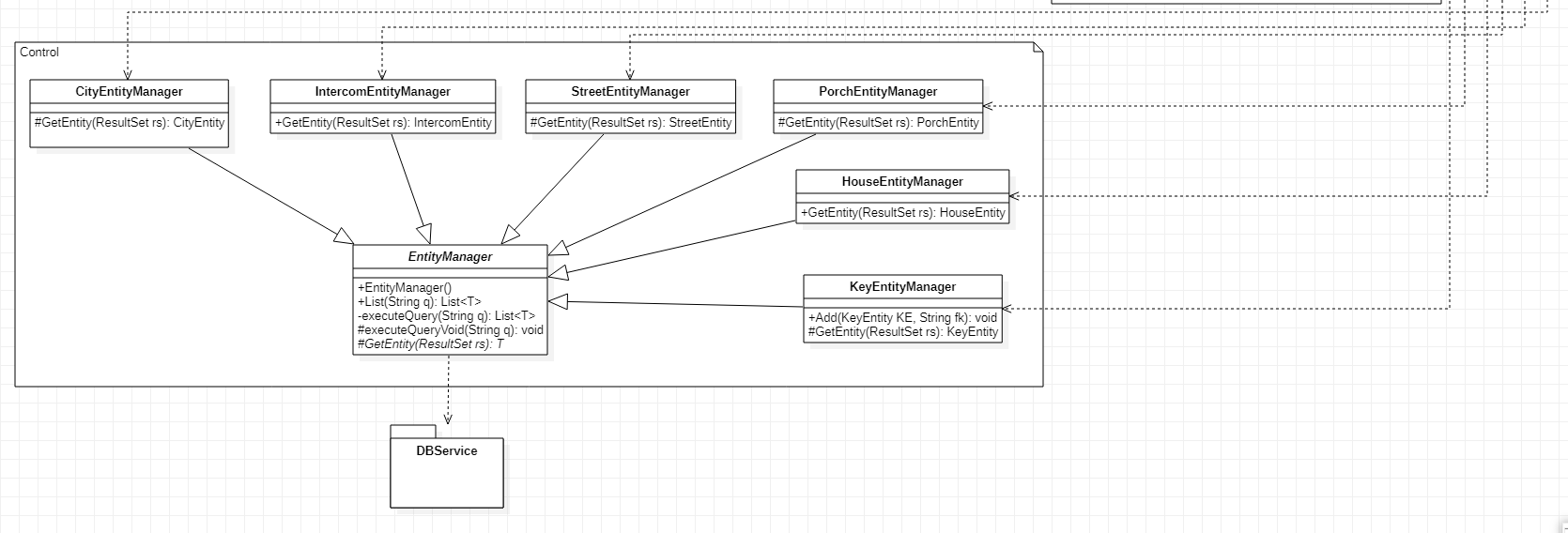
А также на рисунке 17 показана диаграмма пакетов для приложения:



*Рис.17 Диаграмма пакетов приложения*

Подробнее об этих пакетах, что в них лежит и для чего это нужно, будет рассказано в диаграмме классов.

1. 3. 6. Диаграмма классов

*Рис.18 Диаграмма классов сервера*

На рисунке 18 представлена диаграмма классов сервера:

В пакете Entity содержатся сущности модели базы данных:

* CityEntity

Поля:

* + id\_city
  + name\_city

Методы:

* + get()
  + set()
* StreetEntity

Поля:

* + id\_street
  + name\_street
  + city\_id

Методы:

* + get()
  + set()
* HouseEntity

Поля:

* + id\_house
  + number
  + notes
  + street\_id

Методы:

* + get()
  + set()
* PorchEntity

Поля:

* + id\_porch
  + number
  + house\_id

Методы:

* + get()
  + set()
* IntercomEntity

Поля:

* + id\_intercom
  + manufacturer

Методы:

* + get()
  + set()
* KeyEntity

Поля:

* + id\_key
  + code
  + data\_start
  + porch\_id
  + intercom\_id

Методы:

* + get()
  + set()

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

* CityServlet
* StreetServlet
* HouseServlet
* PorchServlet
* KeyServlet
* IntercomServlet

У каждого класса есть соответствующий экземпляр класса Service, и методы doGet(), doPost().

В пакете Boundary содержатся классы сервиса управления, которые вызывают методы соответствующих классов Control:

* CityService

Метод

* + GetCities(): List<CityEntity>
* IntercomService

Метод

* + GetIntercom(): List<IntercomEntity>
* StreetService

Метод

* + GetStreets(int id\_city): List<StreetEntity>
* HouseService

Метод

* + GetHouses(int id\_street): List<HouseEntity>
* PorchService

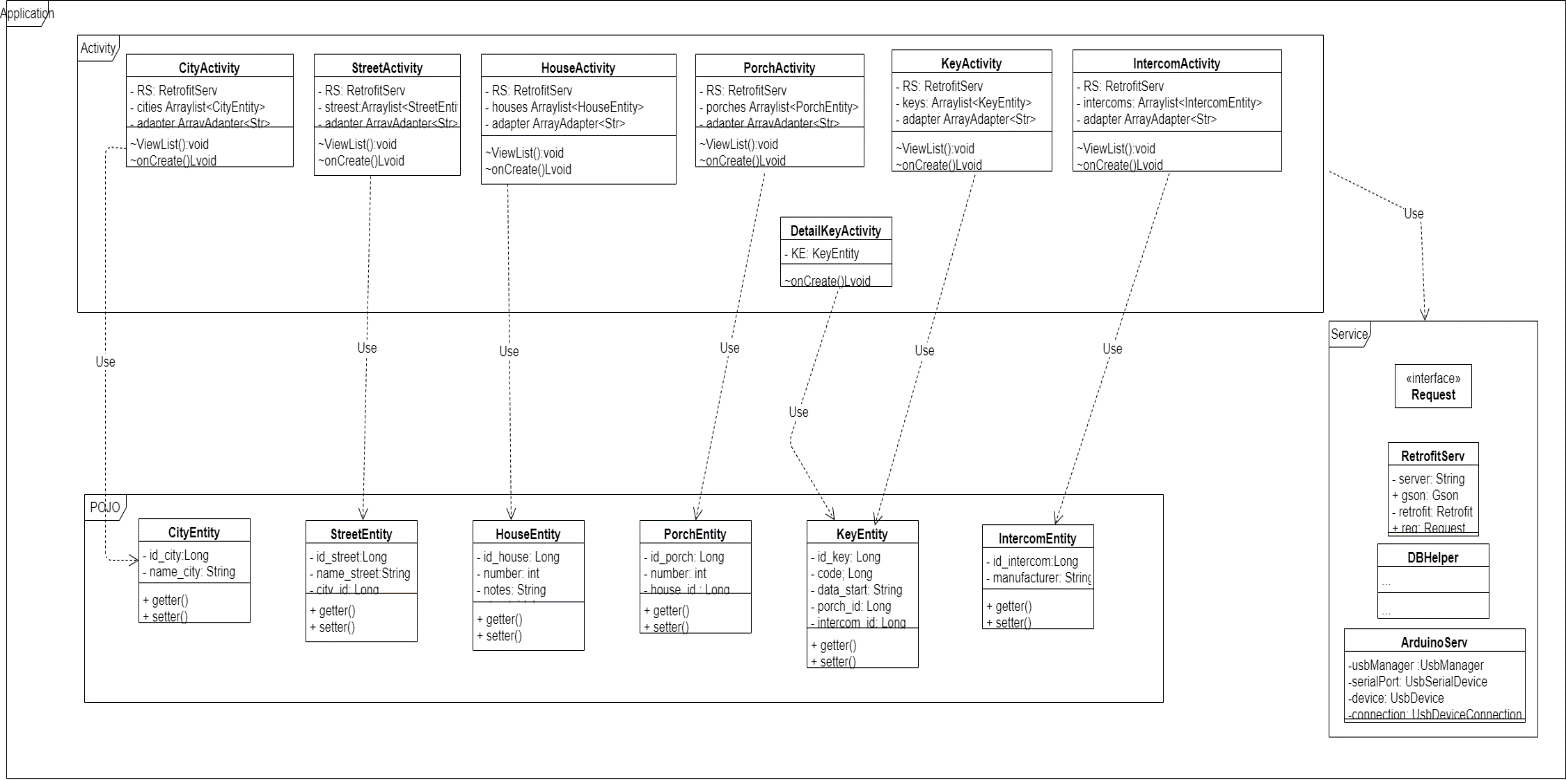
Метод

* + GetPorches(int id\_house): List<PorchEntity>
* KeyService

Методы

* + GetKeysViaPorch(int porch\_id): List<KeyEntity>
  + GetKeysViaIntercom(int intercom\_id): List<KeyEntity>
  + AddKey(keyEntity KE): void

В пакете Control классы делают SQL запросы (с помощью пакета DBService) базе данных. Все классы в данном пакете наследуются от абстрактного класса EntityManager с абстрактным методом GetEntity(ResultSet rs): T.



*Рис. 19 Диаграмма классов приложения*

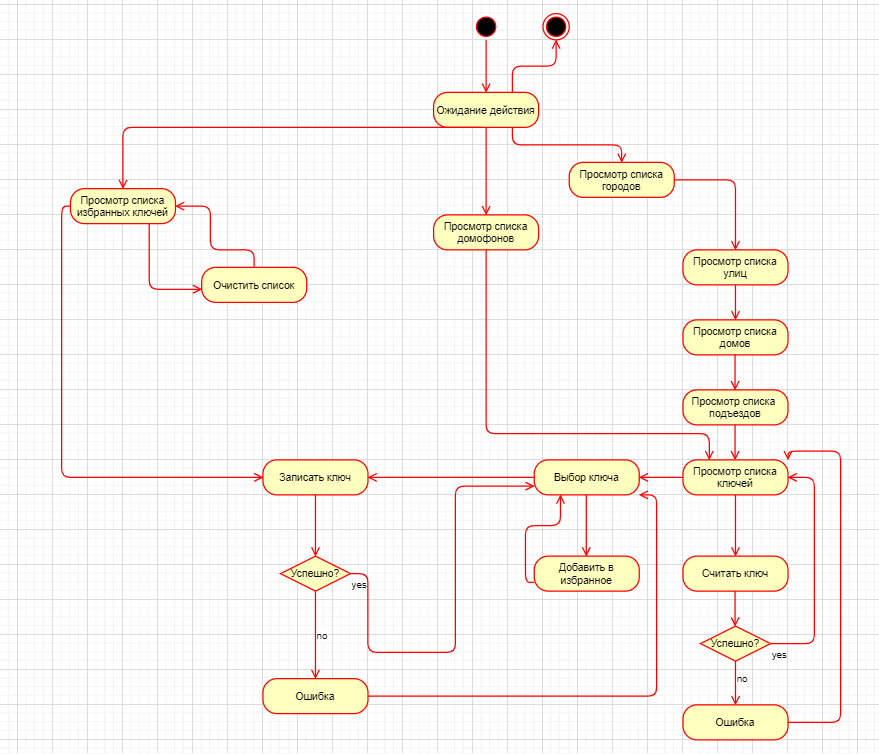
На рисунке 19 представлена диаграмма классов приложения:

В пакете POJO содержатся сущности модели базы данных, поля и методы которых аналогичны классам пакета Entities сервера

В пакете Activity содержатся активности для всех сущностей, которые используют библиотеку Retrofit для запросов на сервер и отображают разметку layout.xml. Так же в этот пакет входят такие активности для записи ключа, чтения ключа, избранных ключей, главной страницы.

В пакете Service содержатся классы сервиса, такие как интерфейс библиотеки Retrofit для запросов на сервер, класс для локальной БД – DBHelper, сервис самой библиотеки Retofit, и последний класс – это ArduinoServ - который выполняет операции, связанные с Arduino, такие как подключение Arduino, принятие сообщение от него.

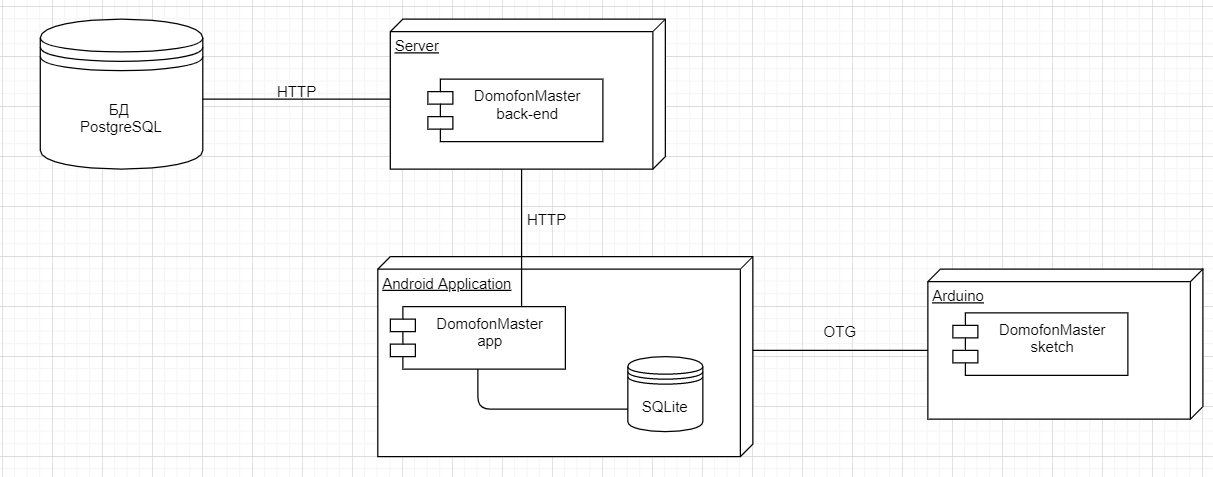
* + 1. Диаграмма состояний



*Рис. 20 Диаграмма состояний*

На рисунке 20 изображена диаграмма состояний, которая дает понять состояния программы во время действий пользователя и их последовательность.

* + 1. Диаграмма развертывания



*Рис. 21 Диаграмма развертывания*

На рисунке 21 изображена диаграмма развертывания приложения. По ней мы видим какие узлы существуют, какие компоненты в этих узлах и как все связано.

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

* 1. Архитектура приложения

В разработке приложения были выбраны такие основные технологии как:

* ЯП Java
* Jakarta EE (Java EE)
* Apache Tomcat 9
* 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.

Gradle подтянул следующие зависимости:

* com.github.felHR85:UsbSerial:6.1.0
* junit:junit:4.12
* com.squareup.retrofit2:retrofit:2.9.0
* com.google.code.gson:gson:2.8.6

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 неплохой 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.

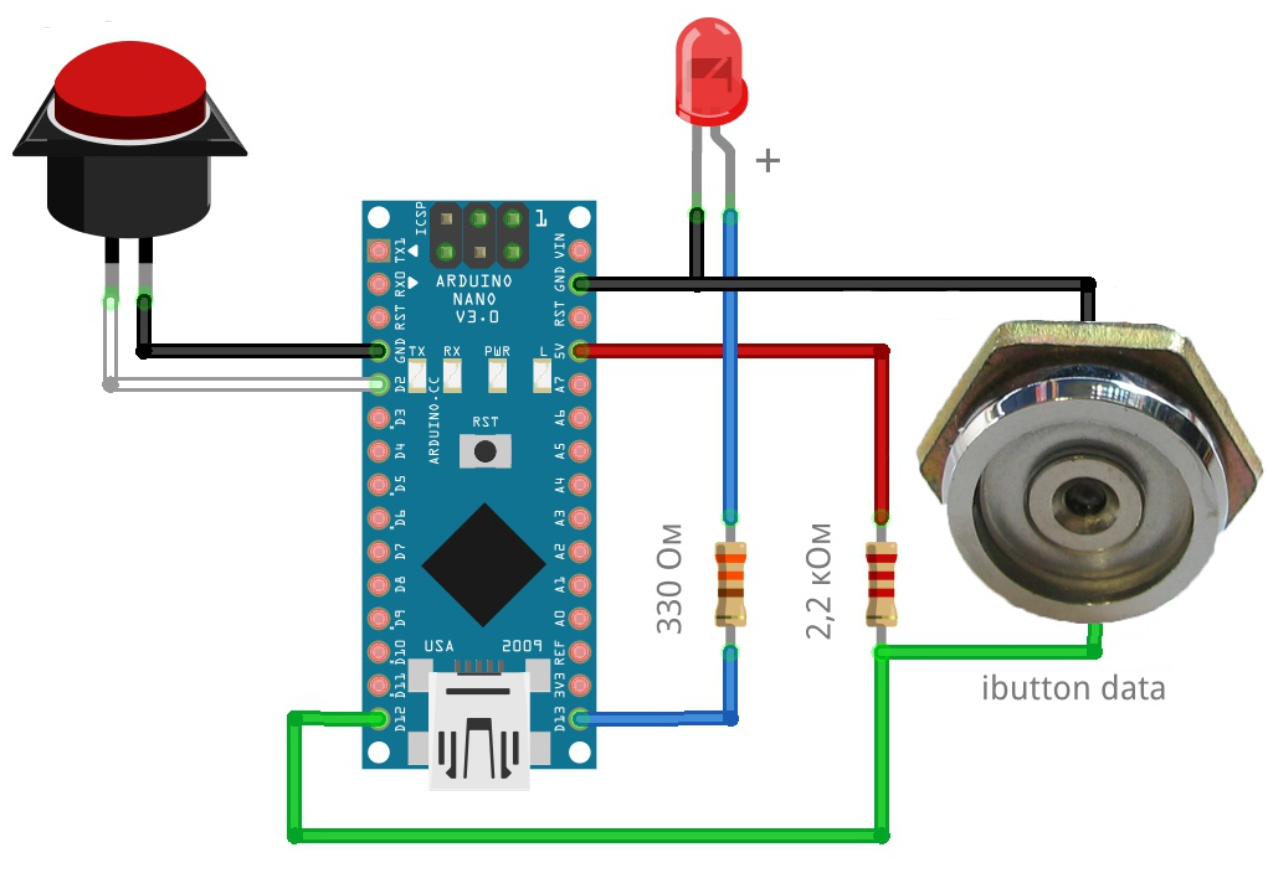
* 1. Архитектура аппаратно-программного средства Arduino

Данное устройство было собранно на Arduino Nano. Подключение с телефоном происходит через OTG-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 22.



*Рис.22 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.

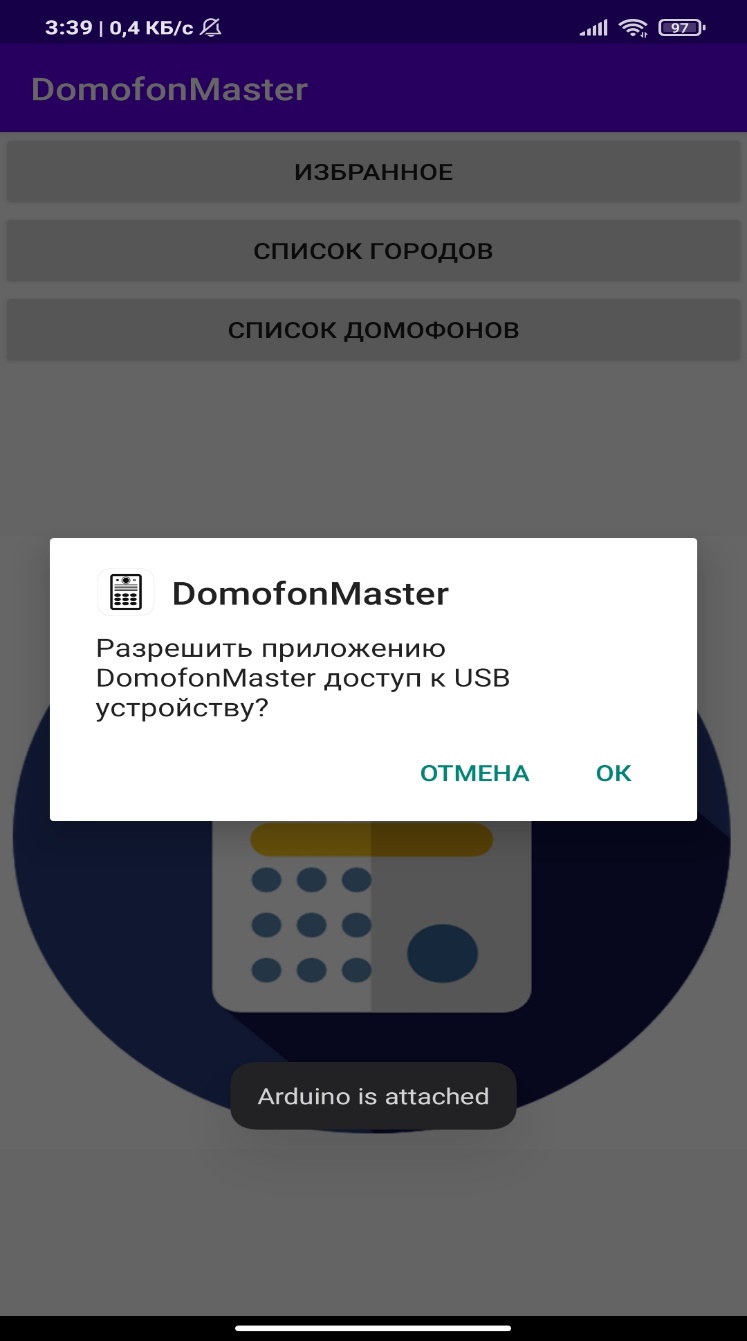
Чтобы записать ID в перезаписываемый ключ, надо нажать кнопку на устройстве. При этом зажигается светодиод, что говорит о готовности устройства к записи. Прикладываем записываемый ключ к контактной площадке считывателя: светодиод при этом гаснет, а примерно через секунду начинает часто моргать, что говорит 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.

В итоге, после сборки схемы и подключения ее к устройству, подключение выглядит следующим образом (рисунок 23):

//тут фото

# Практическая часть

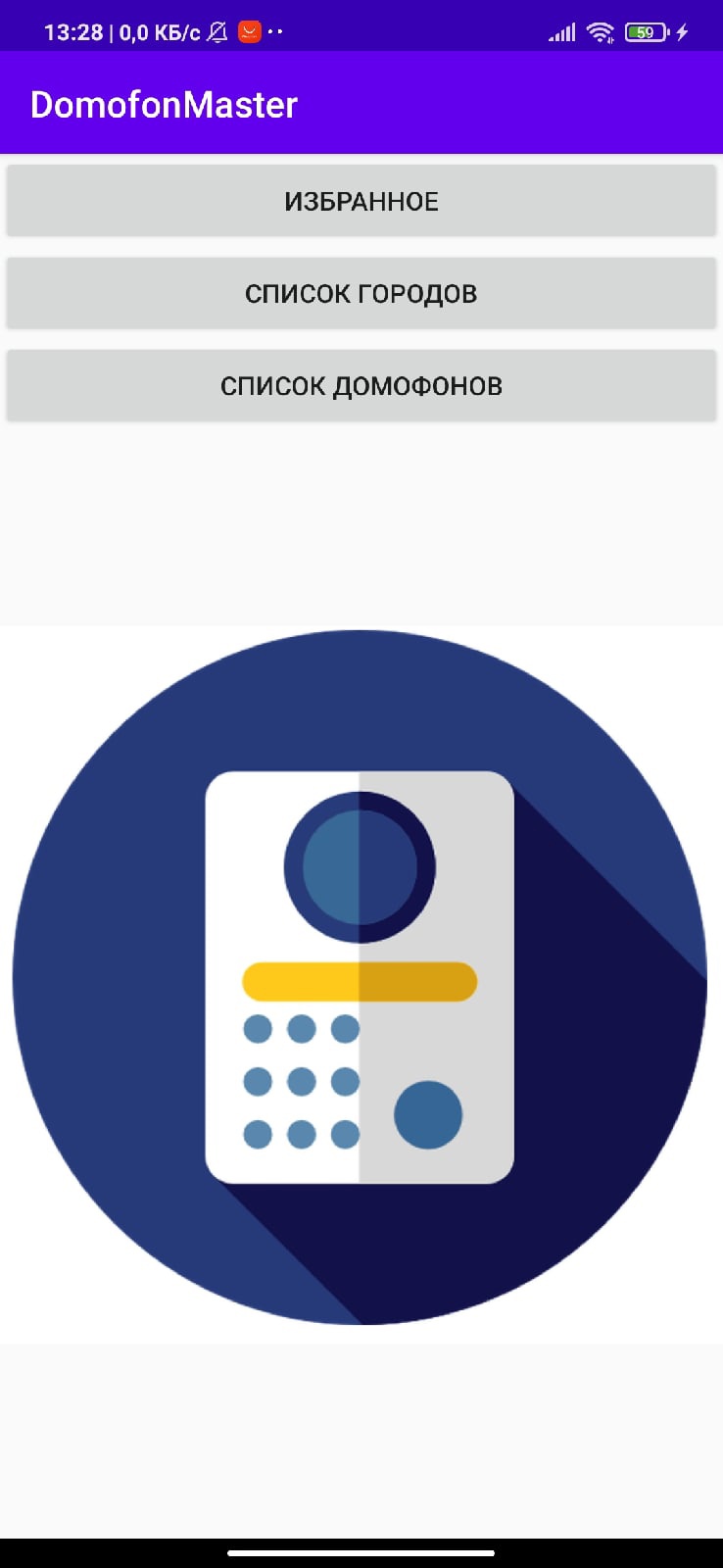
* 1. Подключение Arduino к телефону



*Рис.24 Подключение Arduino*

При старте приложения, когда мы подключаем Arduino устройство, всплывает запрос на разрешение доступа к USB устройству. А так же появляется сообщение, что Arduino было подсоединено.

* 1. Начальный экран



*Рис.25 Начальный экран*

На рисунке 25 изображен начальный экран приложения. Здесь можно выполнить 3 действия:

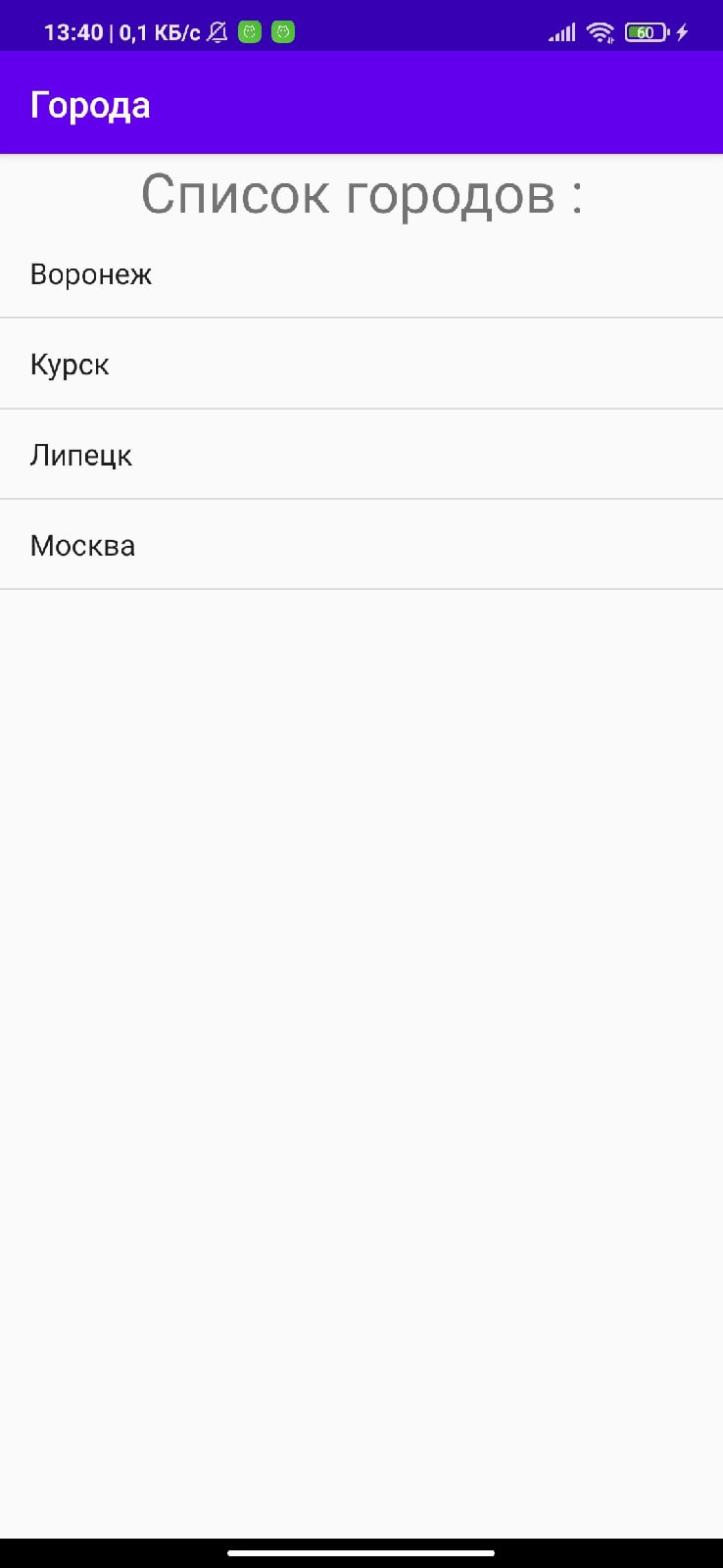
* Посмотреть список избранных ключей
* Посмотреть список городов
* Посмотреть список домофонов
  1. Получение списка ключей из БД

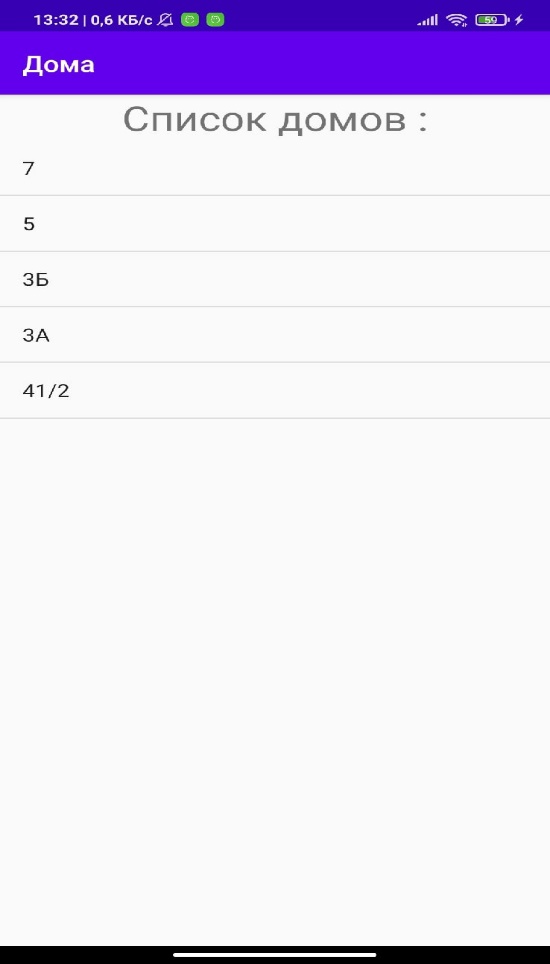
На следующих рисунках (26-30) мы выбираем нужный нам их списка город, затем улицу этого города, затем нужный нам дом, и его подъезд.

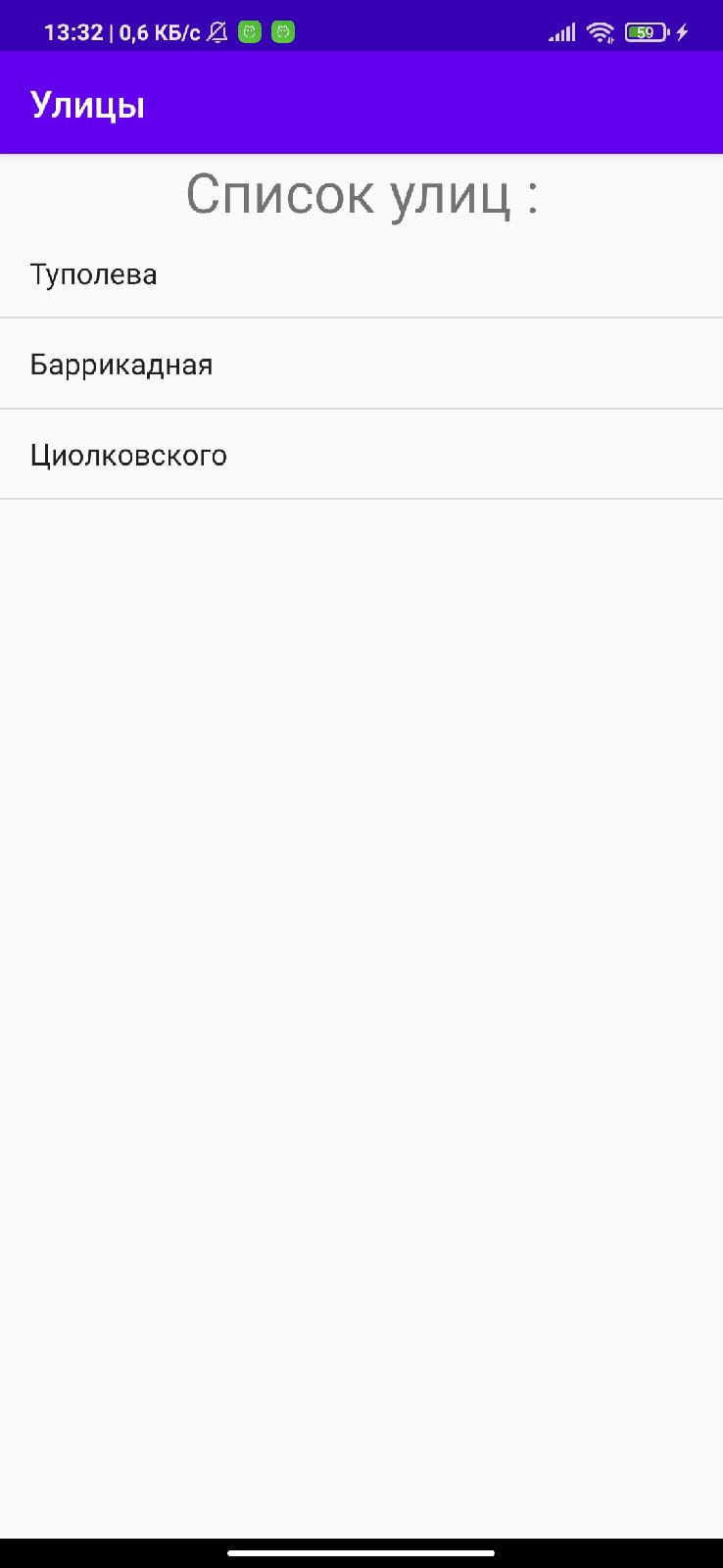
Итак, выберем следующие пункты. Воронеж -> Туполева -> 7 ->1

Для города, улицы, дома и подъезда соответственно.

*Рис. 26 Список городов*







*Рис. 27 Список улиц Рис. 28 Список домов*

**



*Рис. 29 Список подъездов Рис. 30 Список ключей*

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

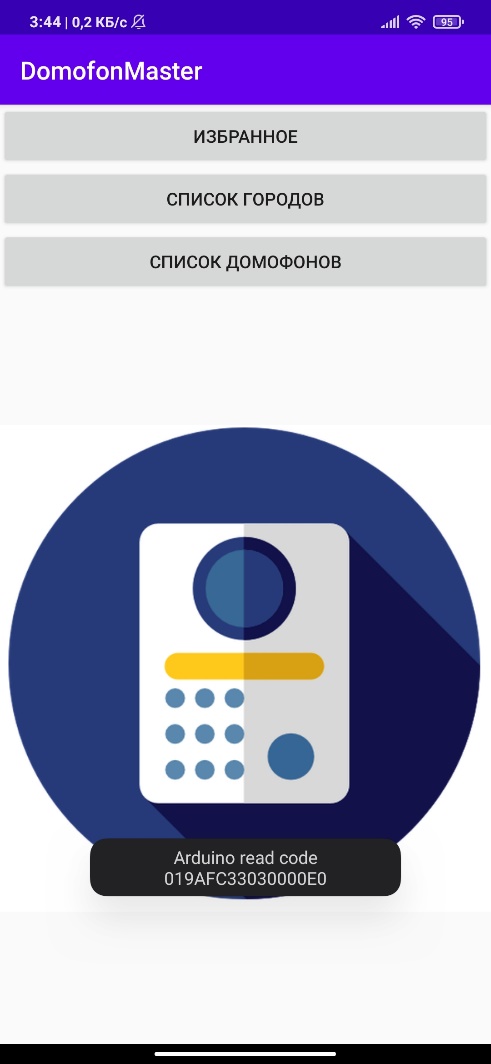
* 1. Добавление ключа в БД

Вернемся и выберем пустой подъезд без ключей. В базе данных на данный момент нет ключей (рисунок 31).



*Рис.31 Отсутствие ключей*

Нажмем кнопку “Добавить ключ”, после чего появится информация о дальнейших действиях, а именно нажать на кнопку и поднести ключ к считывателю (рисунок 32). 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 33). Далее обновим страницу и увидим, что для данного подъезда появился ключ, который мы только что добавили (рисунок 34).

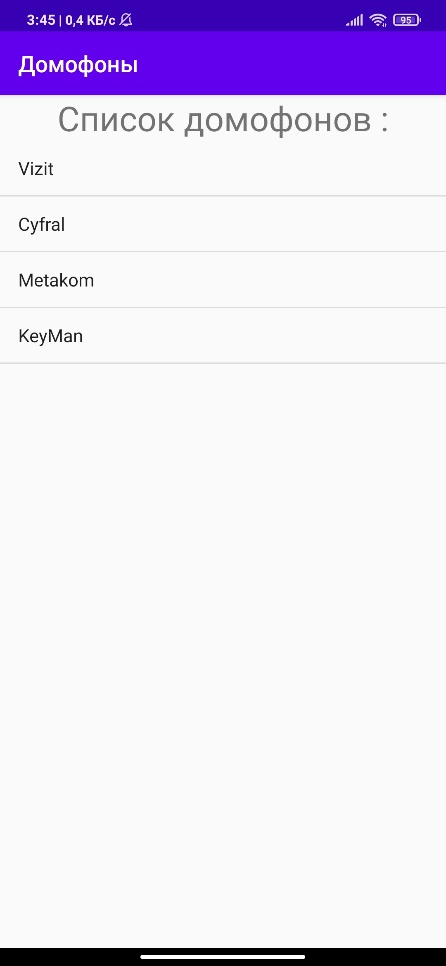
*Рис 32 Считывание ключа Рис. 33 Добавление ключа*

**

*Рис. 34 Добавленный ключ*

* 1. Запись ключа

Выберем текущий ключ для разнообразия из списка домофонов, а именно Vizit. И появится экран, где можно прочитать этот ключ, нажав на кнопку “Записать ключ”, тогда мы передаем Arduino код ключа и после так же появится информация о дальнейших действиях, как можно записать этот код в ключ.

*Рис. 35 Список домофонов Рис. 36 Список ключей в домофоне*

* *

*Рис. 37 Детальное меню ключа Рис. 38 Запись ключа*

**

*Рис. 39 Сообщение о записи ключа*

* 1. Избранные ключи

Добавим ключ из домофона Vizit в избранное. Затем вернемся на начальный экран и выберем “Избранные”. Мы увидим только что добавленный ключ (рисунок 41). Если мы выберем этот ключ, то мы сможем записать его точно так же, как и в предыдущем пункте.

*Рис. 40 Добавление ключа Рис. 41 Список избранного*

*в избранное*

Далее вернемся в список избранных ключей, и очистим его, нажав на соответствующую кнопку. Список очистится и высветится соответствующее сообщение об этом (рисунок 42).



*Рис. 42 Очистка списка избранного*

# Заключение

Данное приложение реализовано полностью, с учетом той архитектуры и требований, которые были указаны.

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
* 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
* Считать код поднесенного ключа к считывателю и отправить его приложению
* Записать в ключ, переданный через приложение код

# Список использованных источников

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

<https://stackoverflow.com> (дата обращения: 28.03.2021-08.06.2021). – 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
3. Взаимодействие Arduino и Android: сайт – URL: <https://radioprog.ru/post/148> (дата обращения: 04.06.2021). – Текст: электронный
4. Форум Arduino: сайт – URL: <http://arduino.ru/forum> (дата обращения 25.05.2021-08.06.2021). – Текст: электронный
5. Библиотека USB Serial: сайт – URL: <https://github.com/felHR85/UsbSerial> (дата обращения 05.06.2021). Текст: электронный