



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н. Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н. Э. Баумана)

---

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

*K KУРСОВОЙ РАБОТЕ*

*на тему:*

*«Разработка базы данных для IoT-платформы умный дом»*

Студент ИУ7-66Б  
(Группа)

(Подпись, дата)

Мамврийский И. С.  
(И. О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

Гаврилова Ю. М.

(И. О. Фамилия)

2024 г.

# РЕФЕРАТ

Целью курсовой работы является разработка базы данных для IoT-платформы умный дом, а так же создание Web-приложения для взаимодействия с данными.

Расчетно-пояснительная записка содержит 43 страницы, 4 таблицы, 28 иллюстраций, 19 источников.

Ключевые слова: IoT-система, умный дом, статус устройства, многопользовательский доступ, владелец, чистая архитектура, GoLang, база данных, система управления базой данных, PostgreSQL.

# СОДЕРЖАНИЕ

<b>Введение</b>	<b>6</b>
<b>1 Аналитическая часть</b>	<b>7</b>
1.1 Существующие решения . . . . .	7
1.2 Формулировка требований к разрабатываемой базе данных и приложению . . . . .	8
1.3 Формализация данных . . . . .	8
1.4 Формализация пользователей . . . . .	9
1.5 Анализ существующих баз данных на основе формализации задачи . . . . .	10
1.5.1 Дореляционная база данных . . . . .	10
1.5.2 Реляционная база данных . . . . .	11
1.5.3 Постреляционная база данных . . . . .	11
<b>2 Конструкторская часть</b>	<b>13</b>
2.1 Сущности базы данных . . . . .	13
2.2 Функции базы данных . . . . .	16
2.3 Роли базы данных . . . . .	17
<b>3 Технологическая часть</b>	<b>18</b>
3.1 Выбор системы управления базой данных . . . . .	18
3.2 Выбор средств реализации . . . . .	18
3.3 Выбор среды разработки . . . . .	19
3.4 Архитектура приложения . . . . .	19
3.5 Реализация ролевой модели . . . . .	20
3.6 Реализация функции . . . . .	20
3.7 Тестирование . . . . .	20
3.8 Графический интерфейс . . . . .	21
<b>4 Исследовательская часть</b>	<b>28</b>
4.1 Технические характеристики . . . . .	28
4.2 Исследование . . . . .	28
<b>Заключение</b>	<b>31</b>

<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>33</b>
<b>Приложение А</b>	<b>34</b>
<b>Приложение Б</b>	<b>35</b>
<b>Приложение В</b>	<b>37</b>
<b>Приложение Г</b>	<b>38</b>

## Введение

В настоящее время технологии интернета вещей (IoT) становятся неотъемлемой частью нашей повседневной жизни, и одним из наиболее заметных примеров их применения является IoT-платформа для умного дома. Она объединяет различные устройства в доме, от светильников до умных терmostатов, в единую сеть, которая может управляться и контролироваться через интернет.

Основной целью IoT-платформы умного дома является создание интеллектуальной инфраструктуры, способной адаптироваться к потребностям и предпочтениям пользователей. Она обеспечивает возможность контролировать освещение, отопление, кондиционирование воздуха, безопасность и другие аспекты жизни в доме с помощью смартфона или другого устройства с доступом в интернет.

С учетом быстрого развития технологий IoT и роста спроса на умные системы, платформы умного дома становятся все более интегрированными, расширяя свои возможности и предлагая новые функции для улучшения качества жизни пользователей.

Целью курсовой работы является разработка базы данных для IoT-платформы умный дом. Для достижения поставленной цели необходимо выполнить следующие задачи:

1. провести анализ существующих решений;
2. сформулировать требования к разрабатываемой базе данных;
3. проанализировать существующие базы данных на основе данной задачи;
4. спроектировать и разработать базу данных;
5. спроектировать и разработать приложение для взаимодействия с базой данных;
6. провести исследование зависимости времени выполнения запросов с использованием индексов и без.

# 1 Аналитическая часть

В данном разделе будут рассмотрены существующие решения и проведен их анализ, сформулированы требования к разрабатываемой базе данных, проведен анализ существующих баз данных, а также описаны пользователи системы.

## 1.1 Существующие решения

Так как IoT-вещей активно развиваются, то на рынке уже представлены различные платформы умных домов. Рассмотрим наиболее популярные из них:

- Apple Homekit [1]
- Intel IoT Platform [2]
- MTS IoT HUB [3]
- Xiaomi MI [4]

Выделим следующие критерии для сравнения выбранных платформ:

- 1) многопользовательский доступ;
- 2) возможность создания нескольких домов;
- 3) история работы устройств.

Сравнение выбранных платформ по указанным критериям представлены в таблице 1.1:

Таблица 1.1 – Сравнение существующих решений

Решение	1	2	3
Apple Homekit	+	+	-
Intel IoT Platform	-	-	+
MTS IoT HUB	-	+	+
Предполагаемое решение	+	+	+

Таким образом, ни одна из платформ не удовлетворяет всем критериям.

## 1.2 Формулировка требований к разрабатываемой базе данных и приложению

В ходе выполнения курсовой работы необходимо разработать базу данных для хранения информации о пользователях, умных домах, устройствах.

Помимо этого, нужно спроектировать Web-приложение, которое будет предоставлять интерфейс для взаимодействия с базой данных с возможностью создавать умные дома, добавлять новые устройства в свой дом, просматривать историю работы устройств.

Необходимо также предусмотреть многопользовательский доступ к дому, возможность добавления других пользователей в свой дом для совместного управления. Требуется реализовать функциональность для разных категорий пользователей, каждый из которых получает свой определенный набор прав.

## 1.3 Формализация данных

Разрабатываемая база данных для IoT платформы умного дома должна содержать информацию о пользователях, устройствах, умных домах, истории работы устройств. Данные категории показаны на ER-диаграмме в нотации Чена 1.1.

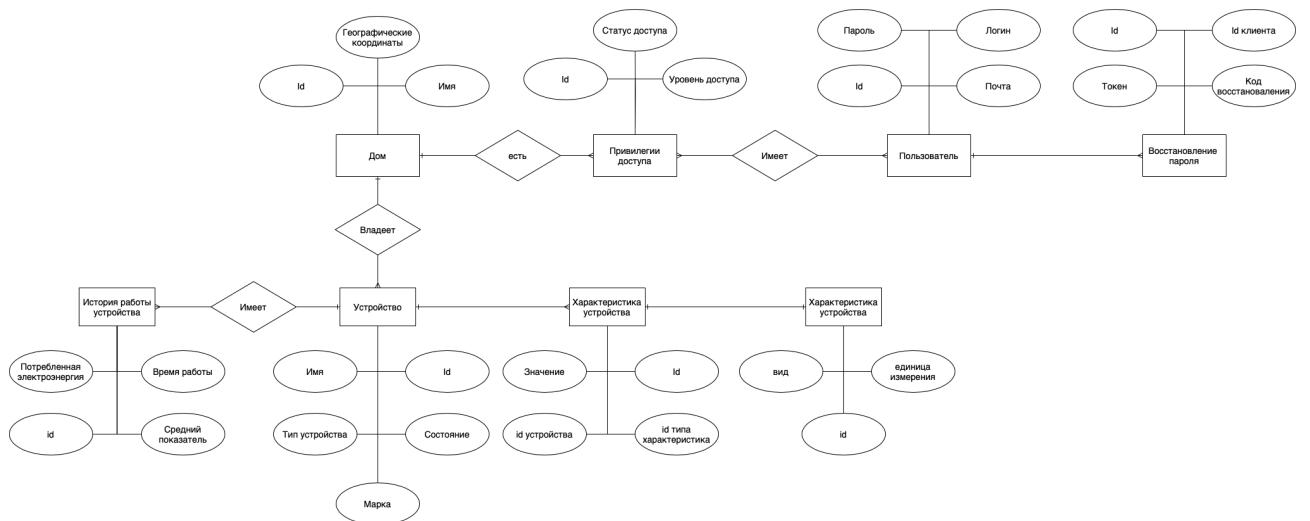


Рисунок 1.1 – ER-диаграмма в нотации Чена

## 1.4 Формализация пользователей

В зависимости от типа пользователя должен различаться набор доступных действий. Таким образом были выделены следующие категории пользователей:

1. гость – пользователь, имеющий возможность зарегистрироваться или авторизоваться на IoT-платформе умного дома;
2. авторизованный пользователь – пользователь, который прошел авторизацию; Он получает возможность создать собственный дом или присоединиться к уже существующему;
3. владелец дома – пользователь, который создал дом. Ему доступны все возможные функции управления домом.
4. участник дома – пользователь, который имеет доступ к дому другого пользователя; Набор доступных действий определяет хозяин дома.

Все доступные действия для каждой категории пользователей представлены на следующей Use-case диаграмме 1.2

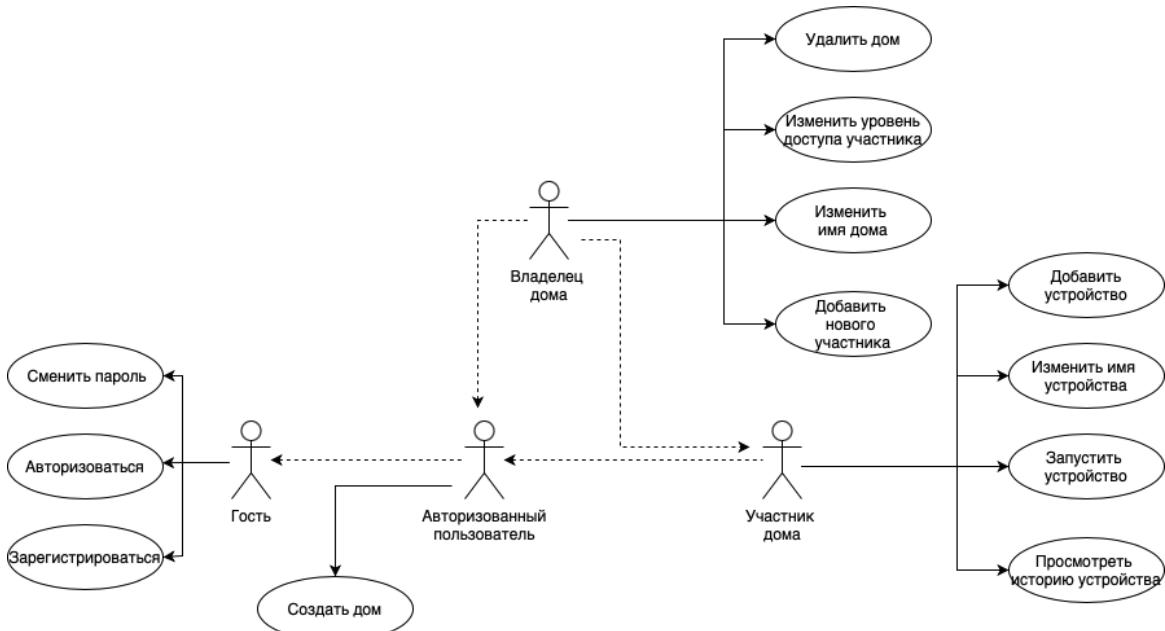


Рисунок 1.2 – Use-case диаграмм пользователей

## **1.5 Анализ существующих баз данных на основе формализации задачи**

База данных – **самодокументированное** собрание **интегрированных** записей [5]. Рассмотрим части данного определения:

1. База данных является самодокументированной, то есть содержит описание собственной структуры, которое называется словарем данных, каталогом данных или метаданными.
2. База данных – собрание интегрированных записей, она содержит:
  - файлы данных;
  - метаданные;
  - индексы;
  - может содержать данные приложений.
3. База данных является информационной моделью пользовательской модели предметной области.

Модель базы данных определяет логическую структуру базы данных и то, каким образом данные будут храниться, организовываться и обрабатываться.

Существует три основных типа моделей базы данных:

- дореляционные;
- реляционные;
- постреляционные.

### **1.5.1 Дореляционная база данных**

К дореляционным моделям баз данных относятся иерархическая и сетевая модели.

Иерархическая модель состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию. Она может быть представлена в виде дерева. Одним из больших минусов данной модели является невозможность отношения «многие-ко-многим».

Основными понятиями такой модели базы данных являются узел и связь. Узел – совокупность атрибутов данных, описывающих некоторый объект. Данная база данных может быть представлена в виде графа. При изменении структуры данной модели придется изменять и приложение, так как логика процедуры выборки данных зависит от физической организации этих данных [6].

### 1.5.2 Реляционная база данных

В реляционных моделях данные организованы в набор двумерных взаимосвязанных таблиц, каждая из которых представляет собой набор столбцов и строк, где столбец представляет атрибуты сущности, а строки представляют записи. Такое представление обеспечивает простой и эффективный способ хранения структурированной информации, доступа к ней, а также легкую сортировку.

Также в данной модели происходит разделение между физическим и логическим уровнями, что позволяет управлять физической системой хранения, не меняя данных, содержащихся в логической структуре [7].

### 1.5.3 Постреляционная база данных

Постреляционная база данных – это база данных, которая является расширенной версией реляционной модели базы данных и позволяет устранить ограничение неделимости данных, хранящихся в записях таблиц. Именно поэтому хранение данных в постреляционной модели по сравнению с реляционной считается более эффективным [8].

Недостатком такой модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных. [9].

## Вывод

В данном разделе проведен анализ аналогов IoT-платформ для умного дома. Ни одно из исследованных решений не соответствовало всем установленным критериям сравнения. При рассмотрении моделей баз данных было принято решение в пользу реляционной модели. Это обосновано необходимостью обеспечения целостности хранящихся данных в разрабатываемой базе данных для IoT-платформы, а также простотой хранения структурированной

информации и возможностью ее сортировки.

Кроме того, были формализованы поставленная задача, данные и категории пользователей.

## 2 Конструкторская часть

В данном разделе будут описаны сущности и построена диаграмма базы данных. Также будет формализована ролевая модель на уровне базы данных и описаны проектируемые функции.

### 2.1 Сущности базы данных

На рисунке 2.1 представлена диаграмма разрабатываемой базы данных.

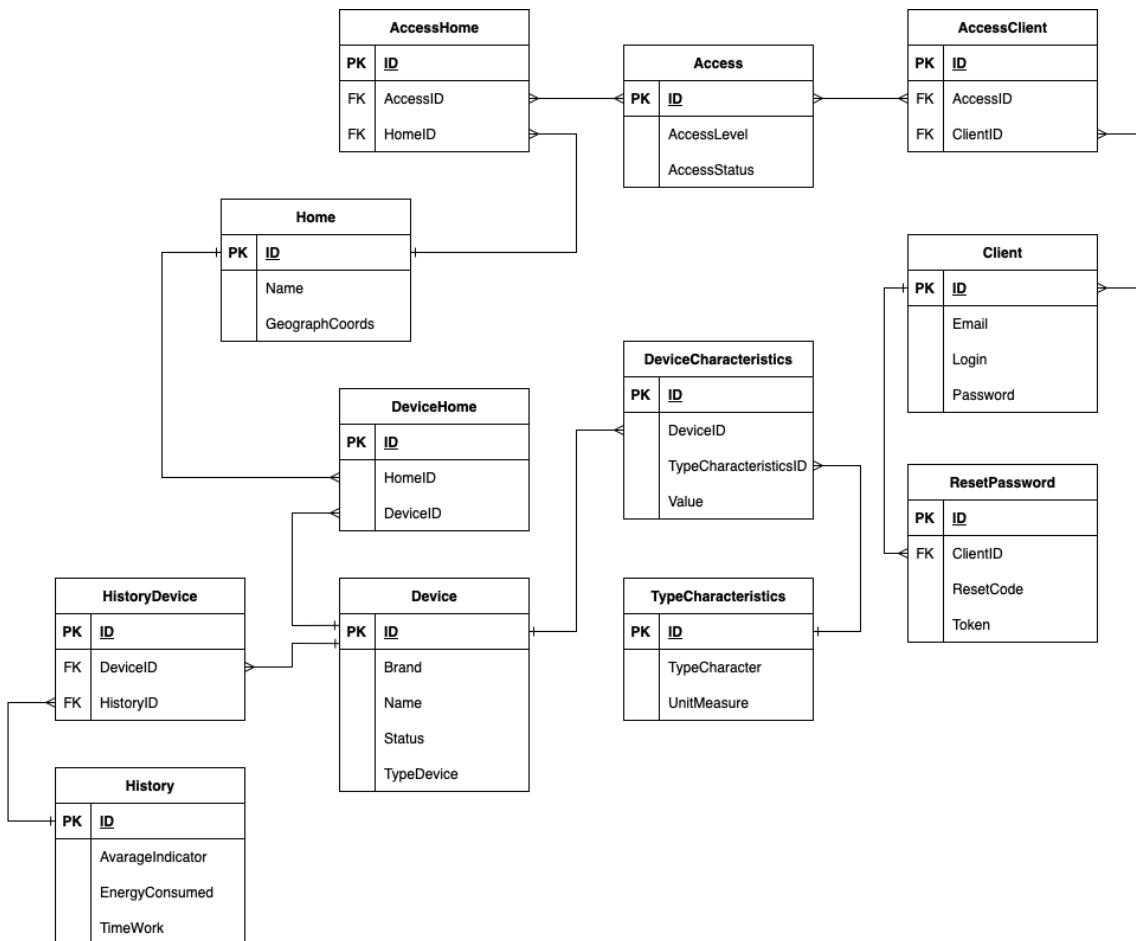


Рисунок 2.1 – Диаграмма разрабатываемой базы данных

На диаграмме представлены следующие таблицы:

- 1) Client – таблица, содержащая информацию о клиентах, состоит из следующих компонентов:
  - ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - Email – почта пользователя (символьный тип с переменным значением, непустая строка);
  - Login – логин пользователя (символьный тип с переменным значением, непустая строка);
  - Password – пароль пользователя (символьный тип с переменным значением, непустая строка), который в базе данных хранится как хешированная строка.
- 2) Access – таблица привилегий доступа, содержащая:
  - ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - AccessLevel – уровень доступа пользователя к дому;
  - AccessStatus – текущий доступ к системе умного дома.
- 3) AccessClient – таблица, которая реализует отношение «один-ко-многим» между таблицами «Client» и «Access», имеет два уникальных идентификатора (клиент и доступ).
- 4) Home – таблица, предоставляющая информацию о домах, содержит следующие поля:
  - ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - Name – имя дома;
  - GeographCoords – географические координаты, показывающие расположение дома.
- 5) AccessHome – таблица, которая реализует отношение «один-ко-многим» между таблицами «Home» и «Access», имеет два уникальных идентификатора (дом и доступ).

- 6) Device – таблица, предоставляющая информацию об устройствах, содержит следующие поля:
- ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - Name – имя устройства;
  - Brand – бренд устройства;
  - Status – текущее состояние устройства;
  - TypeDevice – тип устройства.
- 7) DeviceHome – таблица, которая реализует отношение «многие-ко-многим» между таблицами «Device» и «Home», имеет два уникальных идентификатора (устройство и дом).
- 8) History – таблица, содержащая историю работы устройств, имеет следующие поля:
- ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - AvarageIndicator – средний показатель характеристики устройства;
  - EnergyConsumed – потребленная энергия устройством за время работы;
  - TimeWork – время работы устройства.
- 9) HistoryDevice – таблица, которая реализует отношение «многие-ко-многим» между таблицами «Device» и «History», имеет два уникальных идентификатора (устройство и история).
- 10) ResetPassword – таблица, которая необходима для восстановления пароля пользователем, она содержит:
- ID – уникальный идентификатор(serial4), первичный ключ таблицы;
  - ClientID – уникальный идентификатор пользователя, который восстанавливает пароль;
  - ResetCode – код, необходимый для подтверждения проверки пользователя;
  - Token - токен доступа.

## 2.2 Функции базы данных

Когда устройство начинает работать, необходимо обновлять статус для того, чтобы пользователь мог увидеть текущее состояние устройства. Для этого была описана функция, которая после окончания работы обновляет статус устройства с «Active» на «Inactive» и при запуске устройства с «Inactive» на «Active». Схема алгоритма данной функции указана на следующем рисунке 2.2:

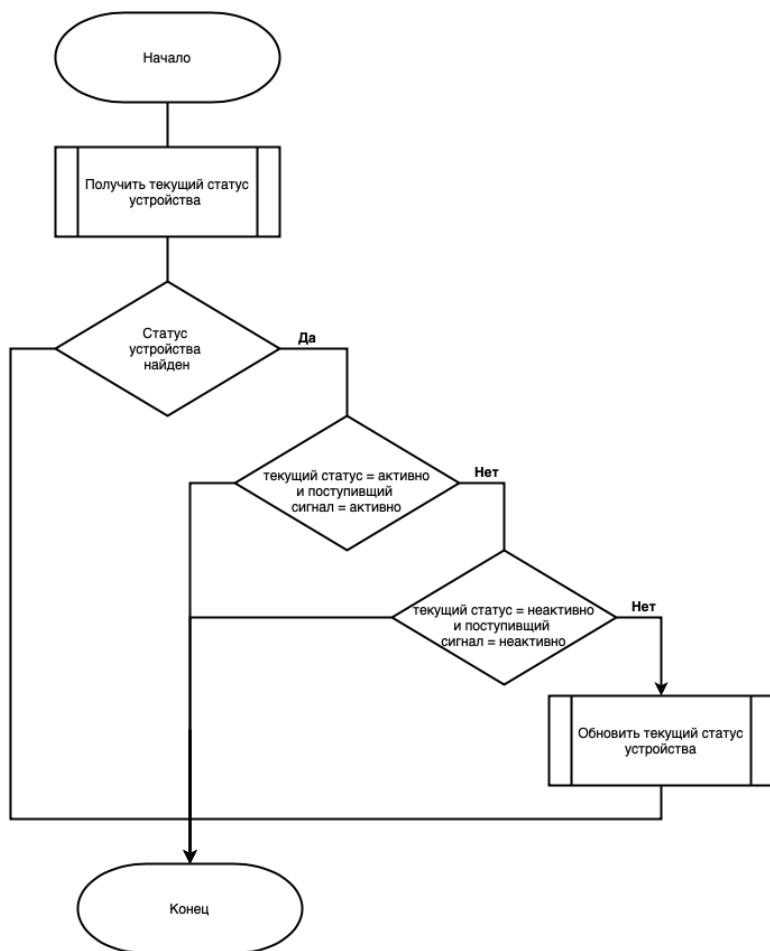


Рисунок 2.2 – ER-диаграмма в нотации Чена

## **2.3 Роли базы данных**

Для управления доступом к таблицам применяются роли. В аналитической части были установлены следующие роли: гость (неавторизованный пользователь), авторизованный пользователь, владелец дома и участник дома. Для каждой из этих ролей необходимо разработать соответствующие права доступа:

- 1) гость(неавторизованный пользователь) должен иметь доступ на изменение и просмотр данных в таблице «Client»;
- 2) авторизованный пользователь имеет все права гостя, так же ему предоставлены изменение и добавление данных в таблицу «Home»;
- 3) участник дома должен иметь все права авторизованного пользователя, так же ему доступен просмотр таблиц «History», «Device», «TypeCharacteristics», «Access», в зависимости от предоставленных прав владельцем дома участнику может быть доступно добавление, изменение данных в указанных таблицах;
- 3) владелец должен иметь возможность просматривать и изменять записи во всех существующих таблицах.

## **Вывод**

В данном разделе были описаны сущности и построена диаграмма базы данных, описана используемая функция и приведена схема ее алгоритма. Так же были formalизованы роли на уровне базы данных, чтобы обеспечить управление доступом к таблицам.

### **3 Технологическая часть**

В данном разделе происходит выбор средств реализации базы данных и приложения, листинги кода, а так же будет показан интерфейс программы.

#### **3.1 Выбор системы управления базой данных**

В аналитическом разделе была выбрана реляционная система управления, наиболее распространенными представителями данной системы являются: Microsoft SQL, PostgreSQL, Oracle, а также MySQL [10].

Выберем следующие критерии для сравнения выбранных систем:

- 1) возможность бесплатного использования;
- 2) опыт работы с данной системой;
- 3) наличие подробной документации.

Сравнение указанных систем управления базой данных(СУБД) представлены в таблице 3.1:

Таблица 3.1 – Сравнение СУБД по указанным критериям

СУБД	Microsoft SQL	PostgreSQL	Oracle	MySQL
1	-	+	-	+
2	-	+	-	-
3	+	+	+	+

По результатам сравнения для реляционной базы данных была выбрана PostgreSQL, так как это единственная система, которая удовлетворяет всем критериям.

#### **3.2 Выбор средств реализации**

Реализация части приложения, которая обеспечивает доступ к базе данных, была выполнена с использованием языка программирования GoLang [11]. Данный выбор обусловлен следующими причинами:

- сборщик мусора, который позволяет автоматически освобождать память;
- статическая типизация, которая помогает обнаруживать ошибки на этапе компиляции;

- встроенная поддержка параллельных вычислений, что позволяет делать разработку многопоточной.

Графический интерфейс был разработан с использованием HTML [12] и CSS [13], так же был выбран язык JavaScript [14] для обработки событий и динамического обновления элементов.

Причина выбора данных средств заключается в следующем:

- HTML и CSS поддерживаются всеми основными веб-браузерами, что гарантирует доступность содержимого на различных устройствах;
- опыт работы с HTML и CSS;
- JS имеет возможность получать доступ к структуре HTML-документа, что позволяет манипулировать элементами, стилями, атрибутами и событиями.

### **3.3 Выбор среды разработки**

В качестве среды разработки был выбран Visual Studio Code [15]. Данный выбор обусловлен следующими причинами:

- бесплатный доступ;
- поддержка выбранных языков;
- опыт работы в данной среде.

### **3.4 Архитектура приложения**

Для разрабатываемого приложения была выбрана чистая архитектура [16], то есть приложение будет состоять из трех слоев: графический интерфейс, бизнес-логика, доступ к данным.

Данное разделение имеет следующие преимущества:

- приложение не зависит от используемых библиотек и фреймворков;
- удобство тестирования;
- возможность использования различных баз данных, так как бизнес-логика не зависит напрямую от используемой базы данных.

### **3.5 Реализация ролевой модели**

Ранее были определены 3 роли: гость (неавторизованный пользователь), владелец дома, а также участник дома. В приложении Б4.2 – Б4.4 представлено создание данных ролей.

### **3.6 Реализация функции**

Функция, осуществляющая обновление текущего статуса устройства в зависимости от его состояния представлена в приложении А4.1. Следует отметить, что если устройство работает и какой-то другой участник дома захочет его запустить, то функция вернет ошибку.

### **3.7 Тестирование**

Чтобы обеспечить автоматизацию тестирования использовался пакет testcontainers [17], который автоматически поднимает контейнеры. Для создания таблиц и ограничений к ним в тестируемой базе данных, а так же для заполнения созданных таблиц использовался пакет goose [18], необходимый для использования миграций.

Во время проведения тестов в базе данных содержалась следующая информация:

1. устройство dev1 с статусом «Inactive»;
2. устройство dev2 с статусом «Inactive».

В таблице 3.2 приведены тесты, которые использовались для проверки правильности работы функции для изменения текущего статуса устройства.

Таблица 3.2 – Тесты, используемые для проверки правильности работы функции для изменения текущего статуса устройства.

Текущее состояние	Новое состояние	ID устройства	Переданный ID	Ожидаемый ответ	Ответ
inactive	inactive	1	1	-3	-3
inactive	inactive	2	2	-3	-3
inactive	active	2	2	0	0
inactive	active	1	1	0	0
active	active	2	2	-2	-2
active	inactive	1	1	0	0
active	inactive	2	2	0	0

Также были проведены тесты для других запросов к базе данных, результат которых представлен в приложении В4.1 – В4.2.

### 3.8 Графический интерфейс

Независимо от роли пользователь попадает на страницу с авторизацией 3.1, где ему предоставляется возможность войти в личный аккаунт с помощью пароля и почты или зарегистрироваться в системе, также пользователь может восстановить пароль.

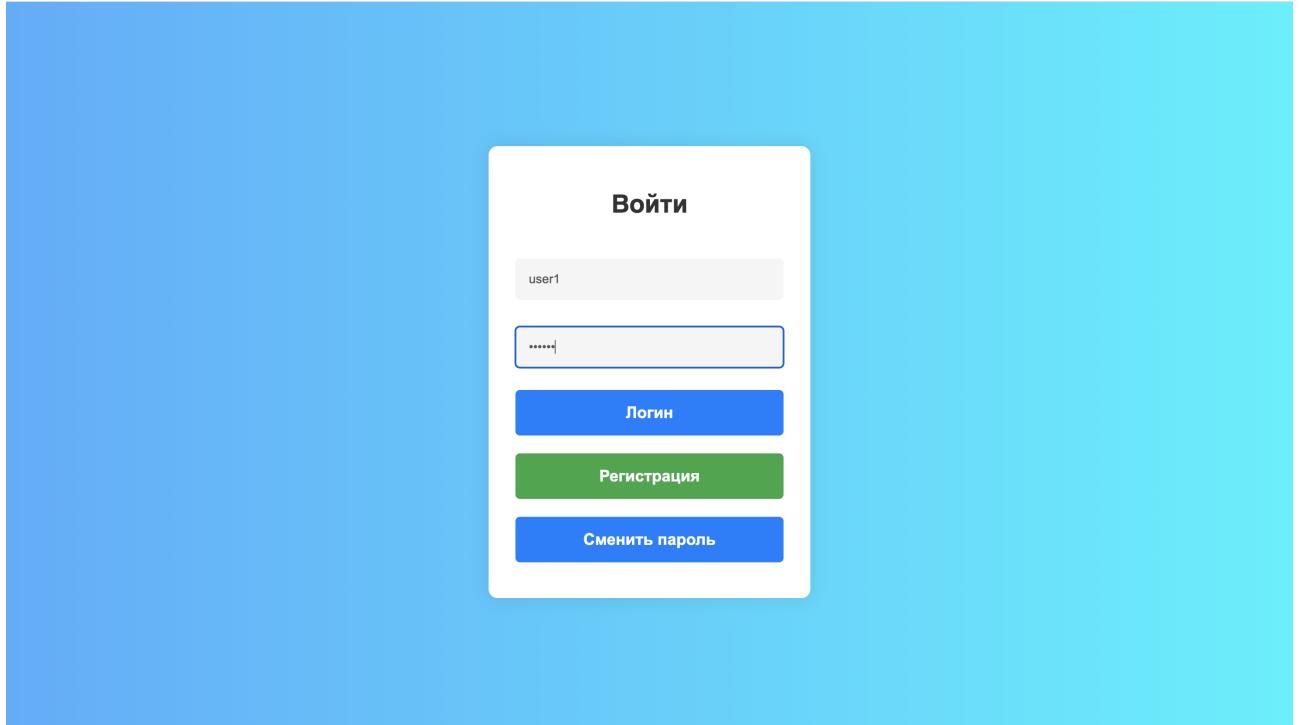


Рисунок 3.1 – Страница для авторизации пользователя

На странице регистрации пользователь может создать аккаунт, введя логин, пароль и почту 3.2.

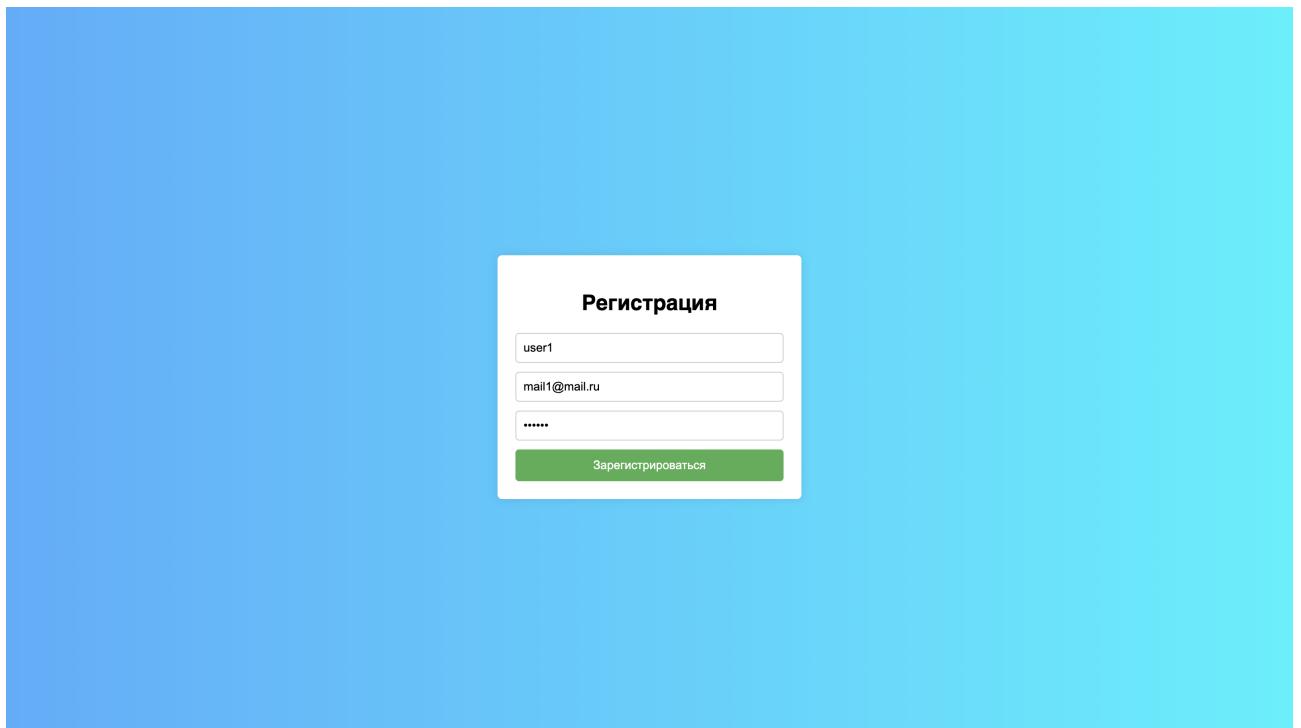


Рисунок 3.2 – Страница для регистрации пользователя

На рисунке 3.3 показана страница с восстановлением пароля. Пользователю необходимо ввести почту, которую он использовал при регистрации.

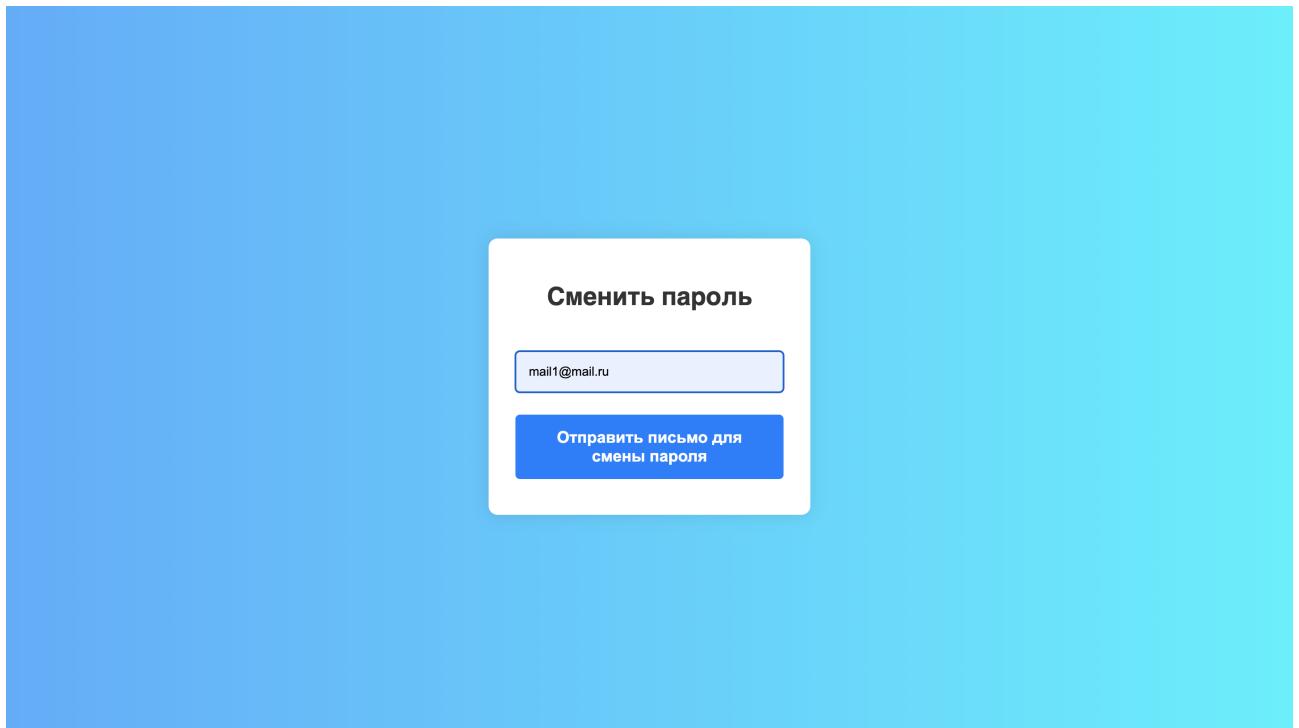


Рисунок 3.3 – Страница с вводом почты для смены пароля

На указанную почту будет выслан код, который необходимо ввести на странице 3.4.

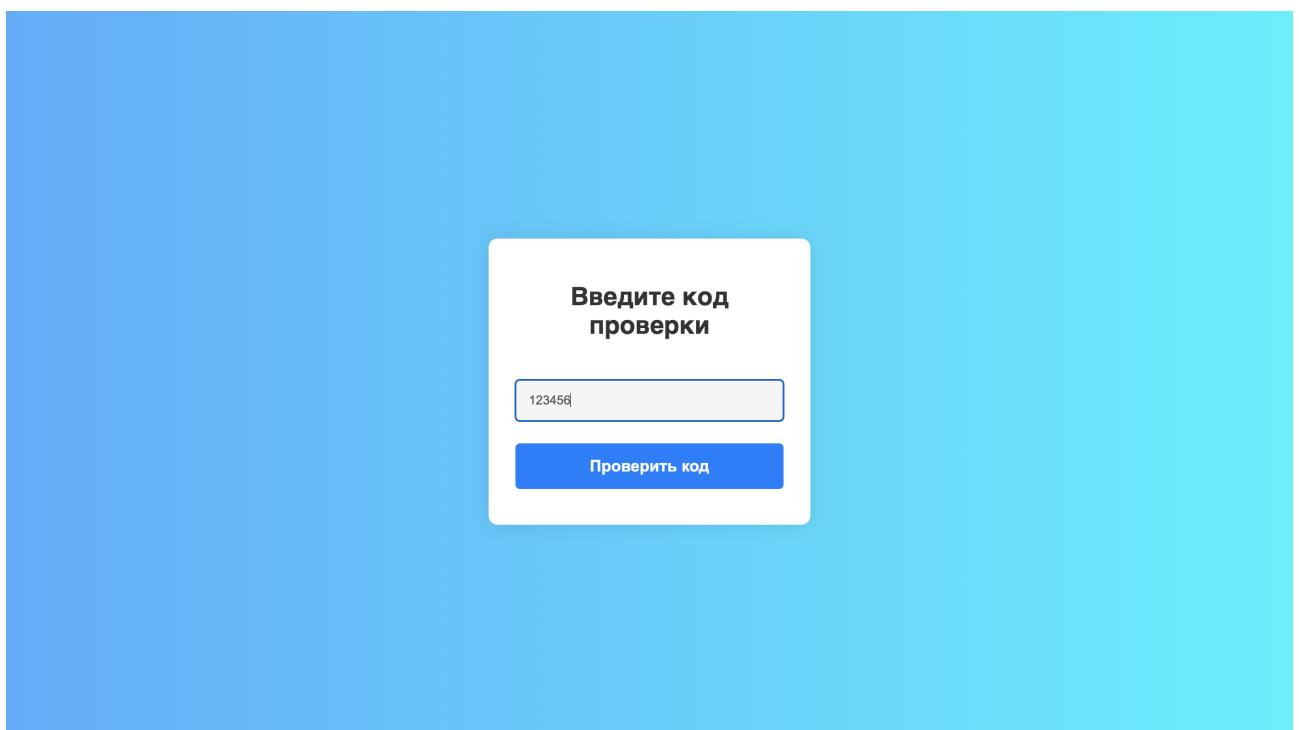


Рисунок 3.4 – Страница с вводом кода проверки

Далее пользователю будет предложено сменить пароль 3.5.

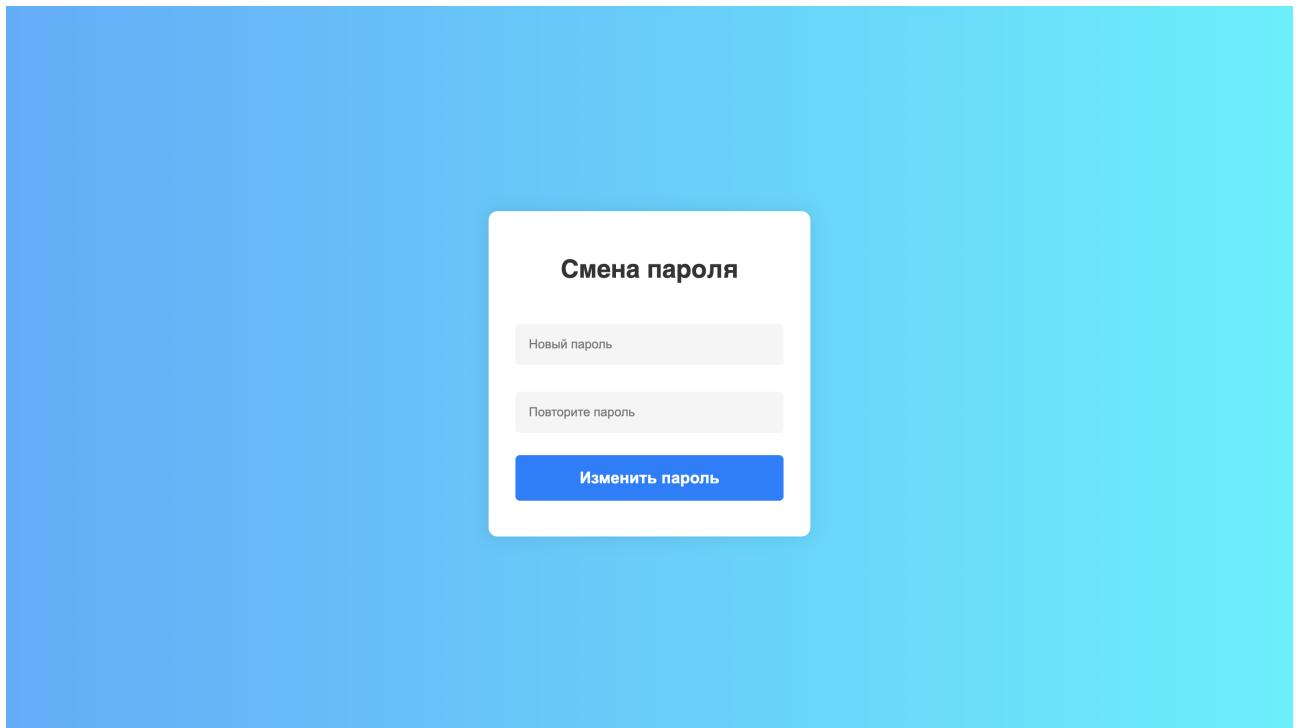


Рисунок 3.5 – Страница со сменой пароля

После авторизации пользователь будет перемещен в меню, которое содержит кнопки для перехода на страницы дома, устройств и прав доступа 3.6.

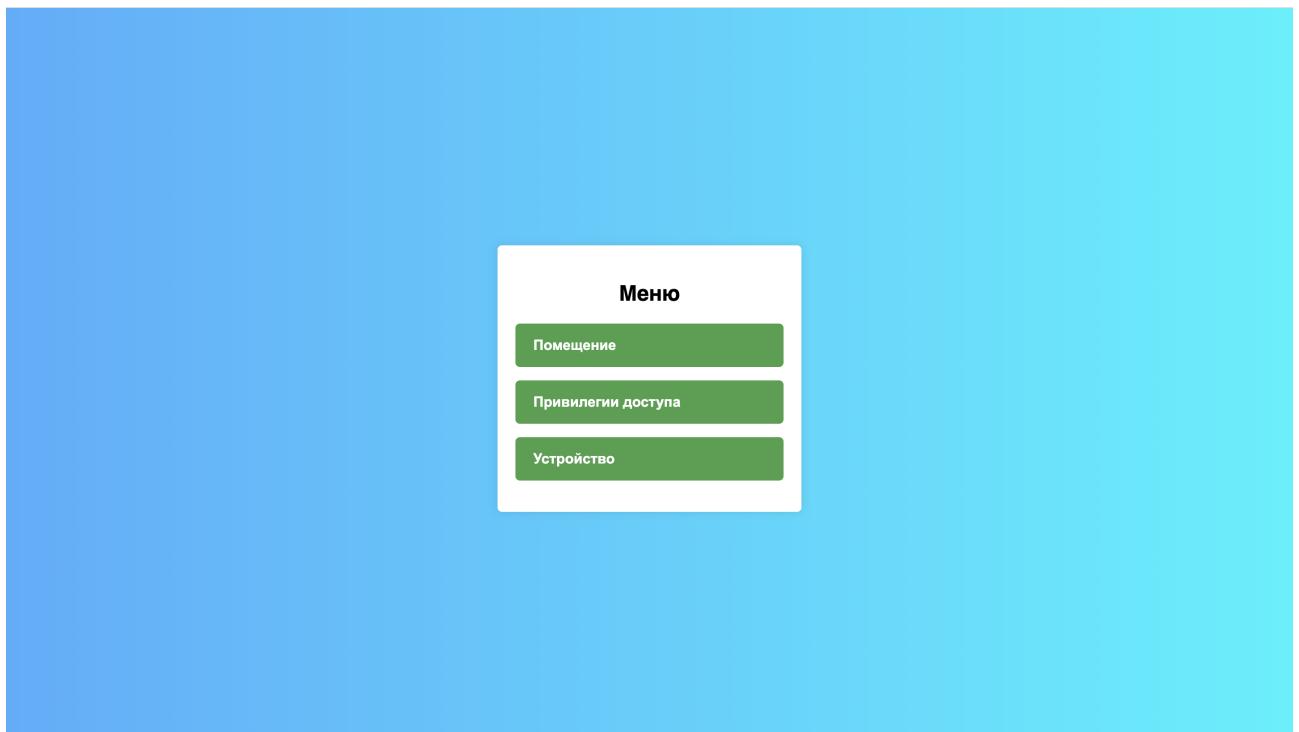


Рисунок 3.6 – Страница со сменой пароля

При нажатии на кнопу «Помещение» пользователь перейдет на страницу

управления домом 3.7. На данной странице предоставляется возможность создать или удалить дом, обновить его имя, а также просмотреть список домов, которыми владеет пользователь.

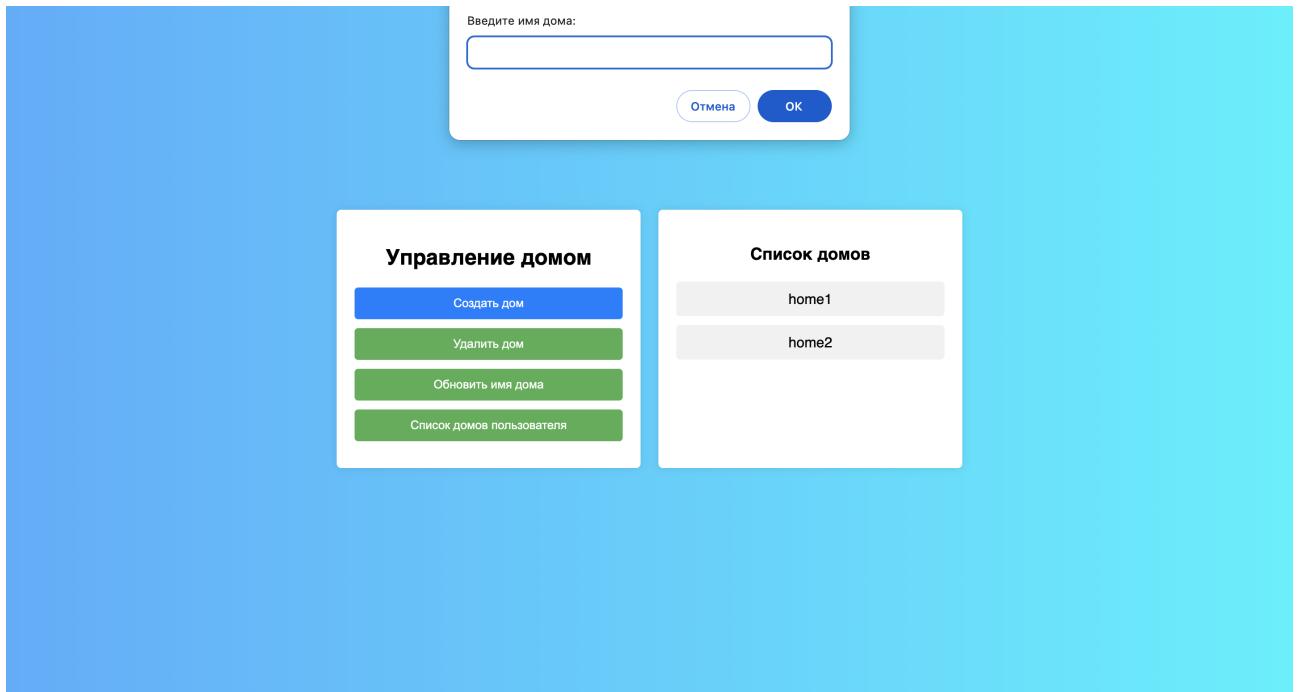


Рисунок 3.7 – Страница для управления домом

При нажатии на кнопку «Привилегии доступа» пользователь переходит на страницу, которая предоставляет возможность управлять многопользовательским режимом 3.8. На данной странице владелец дома может добавить новых участников или удалить их. Также ему предоставляется возможность просмотреть список участников дома и обновить уровень доступа участника.

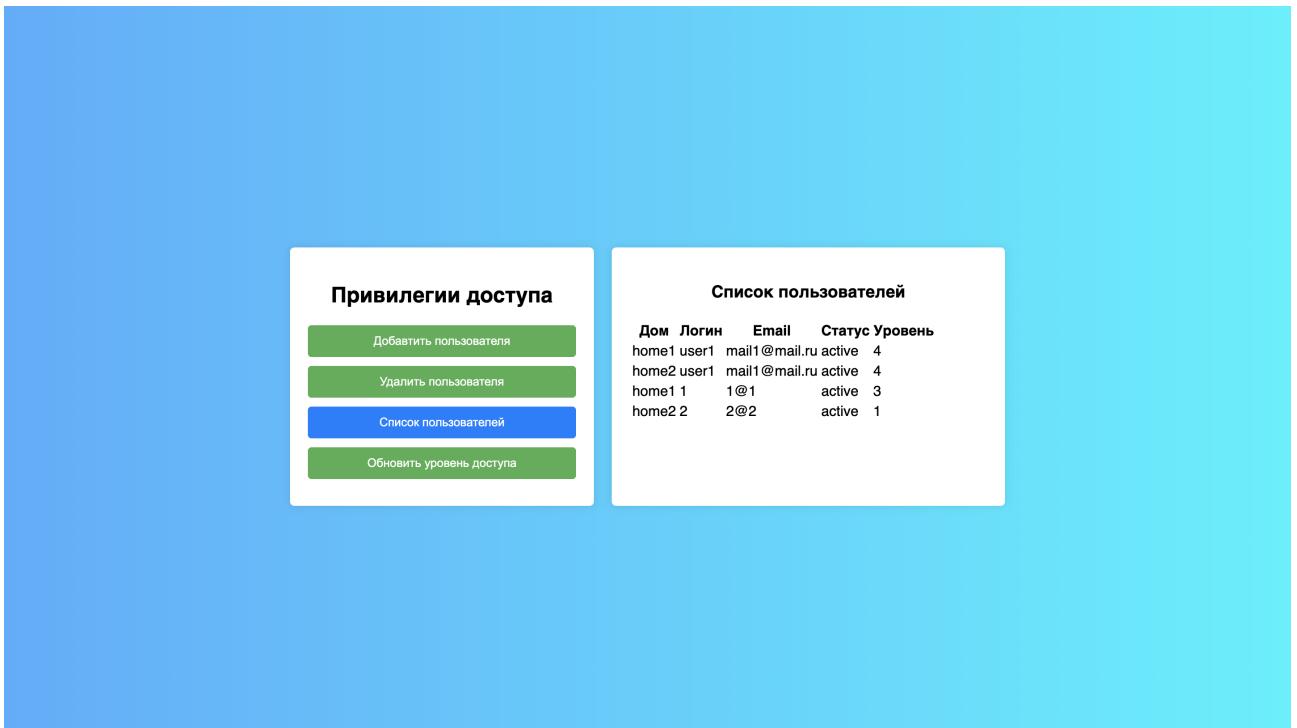


Рисунок 3.8 – Страница многопользовательского режима

При нажатии на кнопу «Устройство» пользователь переходит на страницу управления устройствами 3.9. На данной странице пользователь может добавить новое устройство в свой дом или удалить уже добавленное. Также ему предоставляется возможность просмотреть список устройств, запустить устройство и просмотреть историю работы устройства.

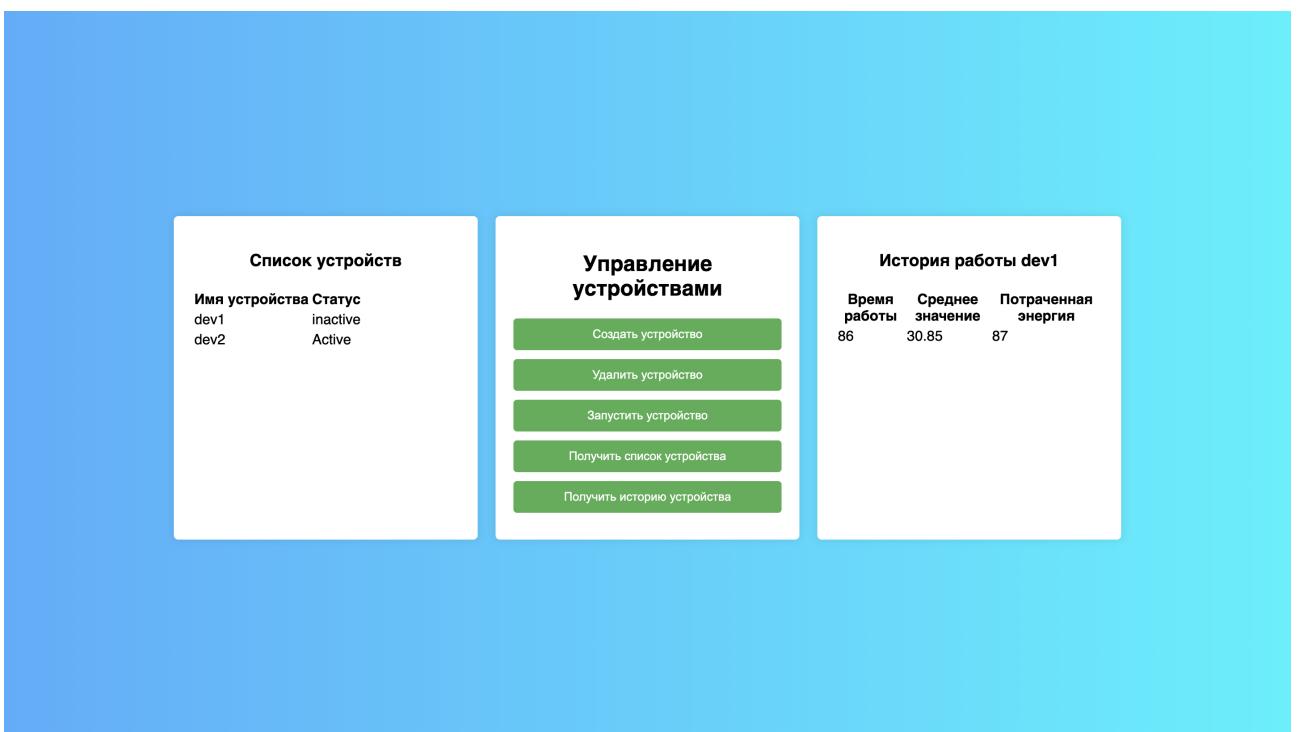


Рисунок 3.9 – Страница управления устройством

## **Вывод**

В данном разделе был произведен выбор системы базы данных, а также были выбраны средства реализации и среда разработки. Также было произведено тестирование описанной ранее функции и запросов, выполняемых к базе данных, описан интерфейс приложения.

## **4 Исследовательская часть**

В данном разделе представлены результаты исследования, которое заключалось в нахождении зависимости времени выполнения запроса с использованием индексов и без, а так же технические характеристики устройства, на котором данное исследование было сделано.

### **4.1 Технические характеристики**

Исследование было выполнено на устройстве, которое имеет следующие характеристики:

- операционная система macOS Sonoma 14.1.1 [19];
- 2 GHz 4-ядерный процессор Intel Core i5;
- оперативная память: 16 Гб.

### **4.2 Исследование**

Для проведения исследования зависимости времени выполнения запроса при наличии индексов и без был выбран поиск пользователя по почте.

Замеры производились при количестве записей от 10000 до 1000000 с шагом 10000. Для того, чтобы избежать статистические выбросы каждый замер происходил 1000 раз, качестве результата бралось среднее значение, которое получалось делением суммы всех повторных замеров на их количество.

Результаты исследования представлены в таблице 4.1.

Таблица 4.1 – Результаты замера времени выполнения запроса по почте в зависимости от использований индексов, с и без.

Количество зарегистрированных пользователей	Время выполнения запроса (без индексов), мкс	Время выполнения запроса (с индексами), мкс
10 000	4 943	2 188
20 000	5 930	2 266
30 000	6 044	2 735
40 000	6 515	2 819
50 000	7 145	3 128
60 000	10 028	4 472
70 000	10 507	4 524
80 000	10 305	4 616
90 000	13 005	4 628
10 0000	14 542	4 827

По полученным данным был составлен график 4.1.

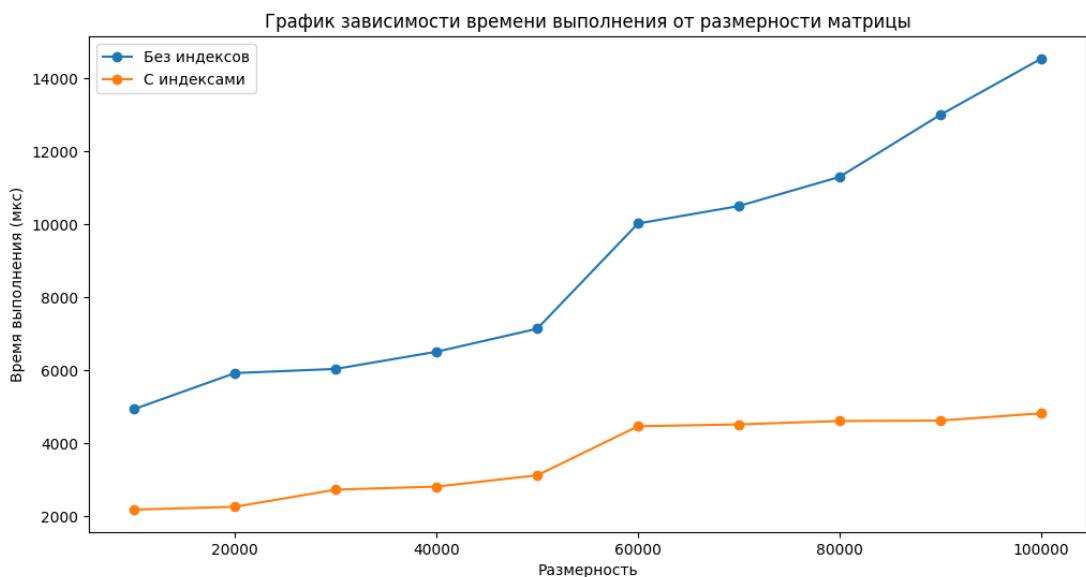


Рисунок 4.1 – График зависимости времени выполнения запроса по почте в зависимости от использования индексов, с и без, и количества записей

По результатам исследования можно сделать вывод о том, что использование индекса помогает ускорить время запроса примерно в 2-3 раза.

## **Вывод**

В данном разделе было описано и проведено исследование, которое заключалось в получении зависимости выполнения запроса с использованием индексов и без. Так же были описаны технические характеристики устройства, на котором проводилось исследование.

Результатом исследования стал вывод о том, что использование индексов помогает ускорить время выполнения запроса в 2-3 раза.

## **Заключение**

Цель курсовой работы, которая заключалась в разработке базы данных для IoT-платформы умный дом, достигнута. Так же были выполнены следующие задачи:

- проведен анализ существующих решений;
- сформулированы требования к разрабатываемой базе данных;
- проанализированы существующие базы данных на основе данной задачи;
- спроектирована и разработана база данных;
- спроектировано и разработано приложение для взаимодействия с базой данных;
- проведено исследование зависимости времени выполнения запросов с использованием индексов и без.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. HomeKit [Электронный ресурс]. — URL: <https://www.apple.com/shop/accessories/all/homekit> (Дата обращения: 20.05.2024).
2. Intel IoT Platform [Электронный ресурс]. — URL: <https://www.intel.com/content/www/us/en/support/ru-banner-inside.html> (Дата обращения: 20.05.2024).
3. MTS IoT HUB [Электронный ресурс]. — URL: <https://business.mts.ru/moskva/iot-platforma> (Дата обращения: 20.05.2024).
4. Xiaomi MI [Электронный ресурс]. — URL: <https://www.mi.com/ru/smart-home/> (Дата обращения: 20.05.2024).
5. Кренке Д. Теория и практика построения базы данных. — Издательство дом «Питер», 2003. — 39 с.
6. Петрозаводский государственный университет [Электронный ресурс]. — URL: [https://cs.petrsu.ru/studies/filatova\\_information/CMD\\_1996566\\_M/my\\_files/Inform/DataBase/](https://cs.petrsu.ru/studies/filatova_information/CMD_1996566_M/my_files/Inform/DataBase/) (Дата обращения: 20.05.2024).
7. Oracle [Электронный ресурс]. — URL: <https://www.oracle.com/cis/database/what-is-a-relational-database/> (Дата обращения: 20.05.2024).
8. Сибирское отделение Российской академии наук [Электронный ресурс]. — URL: [http://db4.sbras.ru/elbib/data/show\\_page.phtml?13+2300](http://db4.sbras.ru/elbib/data/show_page.phtml?13+2300) (Дата обращения: 20.05.2024).
9. Learn Microsoft [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data> (Дата обращения: 20.05.2024).
10. Тортника А. С., Ершов А. С. Обзор и анализ современных систем управления базами // Компьютерные и информационные науки. — 2020. — С. 79—81.
11. Документация GoLang [Электронный ресурс]. — URL: <https://go.dev/doc/> (дата обращения: 22.05.2024).

12. Документация HTML [Электронный ресурс]. — URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (дата обращения: 22.05.2024).
13. Документация CSS [Электронный ресурс]. — URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата обращения: 22.05.2024).
14. Документация JavaScript [Электронный ресурс]. — URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата обращения: 25.05.2024).
15. Документация Visual Studio Code [Электронный ресурс]. — URL: <https://code.visualstudio.com/docs> (дата обращения: 25.05.2024).
16. *Мартин Р.* Чистая архитектура. Искусство разработки программного обеспечения. — СПб: Питер, 2019. — 106 с.
17. Документация Testcontainers [Электронный ресурс]. — URL: <https://golang.testcontainers.org> (дата обращения: 26.05.2024).
18. Документация Goose [Электронный ресурс]. — URL: <https://pkg.go.dev/github.com/pressly/goose/v3> (дата обращения: 26.05.2024).
19. macOS Sonoma [Электронный ресурс]. — URL: <https://www.apple.com/macos/sonoma/> (дата обращения: 26.05.2024).

## Приложение А

Листинг 4.1 – Реализация функции, используемой для смены статуса устройства в соответствии с текущим состоянием

```
1 CREATE FUNCTION update_status(device_id integer, newStatus
2   varchar(10))
3 RETURNS integer AS $$ 
4 DECLARE
5   current_status varchar(10);
6 BEGIN
7   SELECT status INTO current_status FROM device WHERE deviceid =
8     device_id;
9
10  IF NOT FOUND THEN
11    RETURN -1;
12  END IF;
13
14  IF current_status = 'active' AND newStatus = 'active' THEN
15    RETURN -2;
16  END IF;
17
18  IF current_status = 'inactive' AND newStatus = 'inactive' THEN
19    RETURN -3;
20  END IF;
21
22  UPDATE device
23  SET status = newStatus
24  WHERE deviceid = device_id;
25
26  RETURN 0;
27
28 END;
29 $$ LANGUAGE plpgsql;
```

## Приложение Б

Листинг 4.2 – Создание роли гостя (неавторизованного пользователя)

```
1 CREATE ROLE guest with
2     NOSUPERUSER
3     NOCREATEDB
4     NOCREATEROLE
5     NOINHERIT
6     NOBYPASSRLS
7     NOREPLICATION
8     LOGIN
9     PASSWORD 'guest'
10    connection limit -1;
11
12 GRANT select
13     on public."client"
14     TO guest;
15
16 GRANT INSERT
17     on public."client"
18     TO guest;
```

Листинг 4.3 – Создание роли авторизованного пользователя

```
1 CREATE ROLE authorized with
2     NOSUPERUSER
3     NOCREATEDB
4     NOCREATEROLE
5     NOINHERIT
6     NOBYPASSRLS
7     NOREPLICATION
8     LOGIN
9     PASSWORD 'guest'
10    CONNECTION LIMIT -1;
11
12 GRANT SELECT
13     ON ALL TABLES IN SCHEMA public
14     TO authorized;
15
16 GRANT INSERT (homeid, coords, name)
17     ON public.home
18     TO authorized;
```

Листинг 4.4 – Создание роли владельца дома

```
1 CREATE ROLE owner
2     NOSUPERUSER
3     NOCREATEDB
4     NOREPLICATION
5     LOGIN
6     PASSWORD 'owner'
7     CONNECTION LIMIT -1;
8
9 GRANT ALL PRIVILEGES
10    ON ALL TABLES IN SCHEMA public
11    TO owner;
```

## Приложение В

```
2024/08/26 14:59:25 github.com/testcontainers/testcontainers-go - Connected to docker:  
  Server Version: 20.10.20  
  API Version: 1.41  
  Operating System: Docker Desktop  
  Total Memory: 7859 MB  
Testcontainers for Go Version: v0.32.0  
Resolved Docker Host: unix:///var/run/docker.sock  
Resolved Docker Socket Path: /var/run/docker.sock  
Test SessionID: 2ca83f8c7543b90ef8a97ad139319283f40fcf68eebeaf7ea9e8333f3f815dbd  
Test ProcessID: 75a4a10c-51a5-4bad-biae-e33f170b83a4  
2024/08/26 14:59:25 🚀 Creating container for image testcontainers/ryuk:0.7.0  
2024/08/26 14:59:25 ✅ Container created: 1bca6d5fedfc  
2024/08/26 14:59:25 🚀 Starting container: 1bca6d5fedfc  
2024/08/26 14:59:26 ✅ Container started: 1bca6d5fedfc  
2024/08/26 14:59:26 🕒 Waiting for container id 1bca6d5fedfc image: testcontainers/ryuk:0.7.0. Waiting for: &{Port:8080/tcp timeout:<nil> PollInterval:100ms}  
2024/08/26 14:59:26 🌟 Container is ready: 1bca6d5fedfc  
2024/08/26 14:59:26 🚀 Creating container for image postgres:15.4-alpine  
2024/08/26 14:59:27 ✅ Container created: 9bce8e2b5273  
2024/08/26 14:59:27 🚀 Starting container: 9bce8e2b5273  
2024/08/26 14:59:27 ✅ Container started: 9bce8e2b5273  
2024/08/26 14:59:27 🕒 Waiting for container id 9bce8e2b5273 image: postgres:15.4-alpine. Waiting for: &{Port:5432/tcp timeout:<nil> PollInterval:100ms}  
2024/08/26 14:59:30 🌟 Container is ready: 9bce8e2b5273  
2024/08/26 14:59:30 OK 20240822064416_defaultteststable.sql  
2024/08/26 14:59:30 OK 20240822082318_limitation.sql  
2024/08/26 14:59:30 OK 20240825153153_functiontest.sql  
2024/08/26 14:59:30 goose: no migrations to run. current version: 20240825153153  
    === RUN TestCreateClient  
    === RUN TestCreateClient/Test1  
    === RUN TestCreateClient/Test2  
    === RUN TestCreateClient/Test3  
    === PASS: TestCreateClient (0.03s)  
      --- PASS: TestCreateClient/Test1 (0.01s)  
      --- PASS: TestCreateClient/Test2 (0.01s)  
      --- PASS: TestCreateClient/Test3 (0.01s)
```

Рисунок 4.1 – Результаты тестирования работы запросов к базе данных

```
    === RUN TestGetClient  
    === RUN TestGetClient/Test1  
    === RUN TestGetClient/Test2  
    === RUN TestGetClient/Test3  
    === PASS: TestGetClient (0.03s)  
      --- PASS: TestGetClient/Test1 (0.01s)  
      --- PASS: TestGetClient/Test2 (0.01s)  
      --- PASS: TestGetClient/Test3 (0.01s)  
    === RUN TestUpdateStatusFunc  
    === RUN TestUpdateStatusFunc/Test1  
    === RUN TestUpdateStatusFunc/Test2  
    === RUN TestUpdateStatusFunc/Test3  
    === RUN TestUpdateStatusFunc/Test4  
    === RUN TestUpdateStatusFunc/Test5  
    === RUN TestUpdateStatusFunc/Test6  
    === RUN TestUpdateStatusFunc/Test7  
    === RUN TestUpdateStatusFunc/Test8  
    === PASS: TestUpdateStatusFunc (0.08s)  
      --- PASS: TestUpdateStatusFunc/Test1 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test2 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test3 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test4 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test5 (0.00s)  
      --- PASS: TestUpdateStatusFunc/Test6 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test7 (0.01s)  
      --- PASS: TestUpdateStatusFunc/Test8 (0.01s)  
    === RUN TestCreateHome  
    === RUN TestCreateHome/Test1  
    === RUN TestCreateHome/Test2  
    === RUN TestCreateHome/Test3  
    === PASS: TestCreateHome (0.03s)  
      --- PASS: TestCreateHome/Test1 (0.01s)  
      --- PASS: TestCreateHome/Test2 (0.01s)  
      --- PASS: TestCreateHome/Test3 (0.01s)  
PASS  
ok      github.com/Mamriyskiy/database_course/main/testsdatabase      5.943s
```

Рисунок 4.2 – Результаты тестирования работы запросов к базе данных

# Приложение Г



**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

## **Разработка базы данных для IoT-платформы умный дом**

**Студент:** Мамврийский Иван Сергеевич ИУ7-66Б

**Научный руководитель:** Гавrilova Юлия Михайловна

Рисунок 4.1 – Презентация – слайд 1

### **Цель и задачи**

**Целью** курсовой работы является разработка базы данных для IoT- платформы умный дом.

#### **Задачи:**

- провести анализ существующий решений;
- сформулировать требования к разрабатываемой базе данных;
- проанализировать существующие базы данных на основе данной задачи;
- спроектировать и разработать базу данных;
- спроектировать и разработать приложение для взаимодействия с базой данных;
- провести исследование зависимости времени выполнения запросов с использованием индексов и без.

2

Рисунок 4.2 – Презентация – слайд 2

## Анализ существующих решений

### Критерии сравнения:

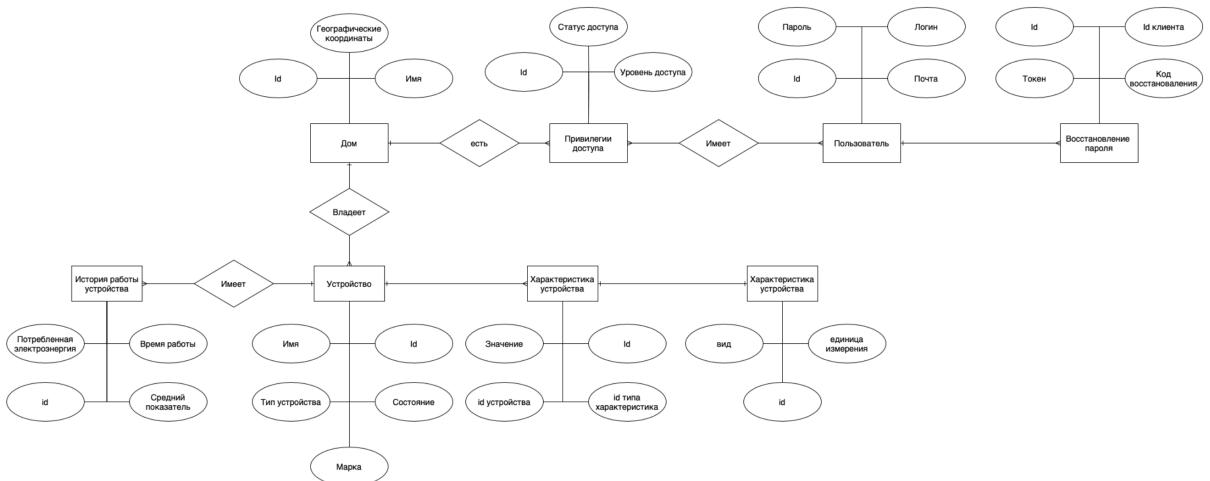
1. многопользовательский доступ;
2. возможность создания нескольких домов;
3. история работы устройств.

Критерий	Apple Homekit	Intel IoT Platform	MTS IoT HUB	Предлагаемое решение
1	+	+	+	+
2	+	+	-	+
3	+	+	+	+

3

Рисунок 4.3 – Презентация – слайд 3

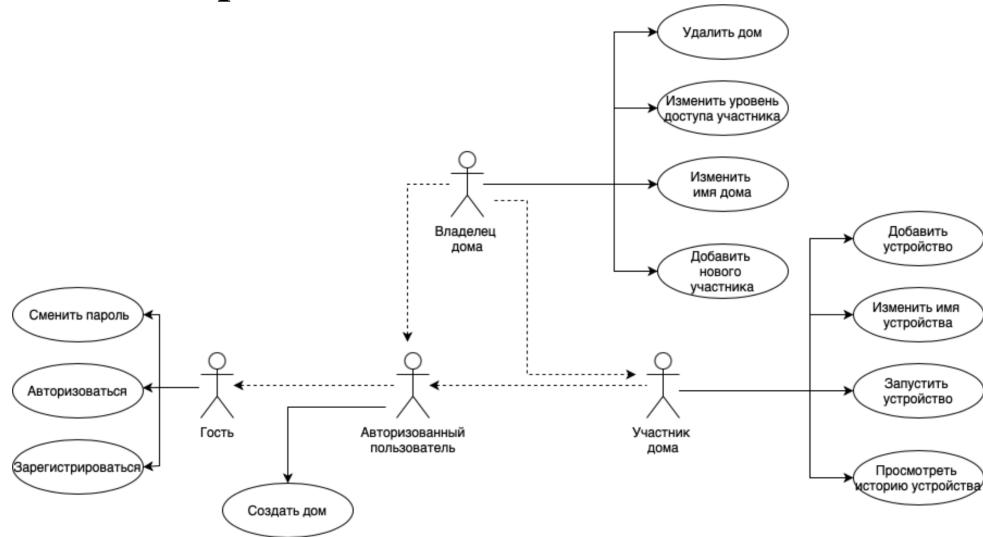
## ER-диаграмма в нотации Чена



4

Рисунок 4.4 – Презентация – слайд 4

## Use-case диаграмма пользователей



5

Рисунок 4.5 – Презентация – слайд 5

## Выбор реляционной СУБД

### Критерии сравнения:

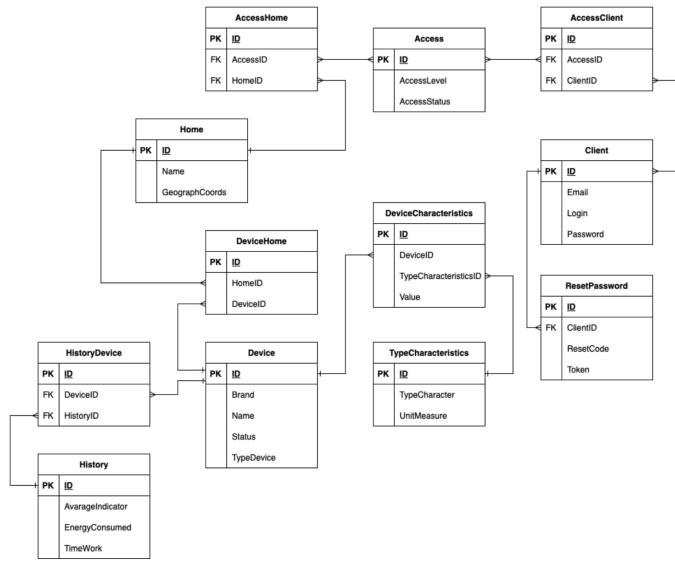
1. возможность бесплатного использования;
2. опыт работы с данной системой;
3. наличие подробной документации.

Критерий	Microsoft SQL Server	PostgreSQL	Oracle	MySQL
1	-	+	-	+
2	-	+	-	-
3	+	+	+	+

6

Рисунок 4.6 – Презентация – слайд 6

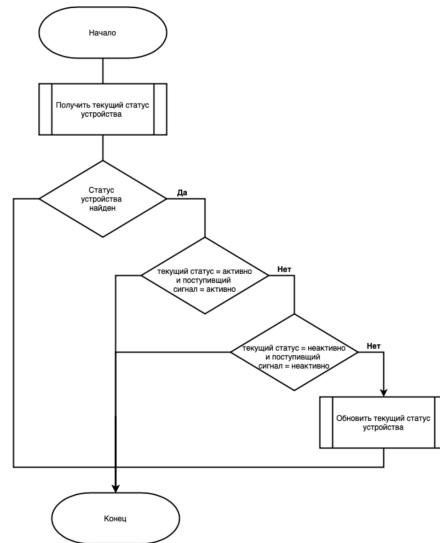
## Диаграмма базы данных



7

Рисунок 4.7 – Презентация – слайд 7

## Схема функции обновления статуса устройства



8

Рисунок 4.8 – Презентация – слайд 8

## **Архитектура приложения и средства реализации**

Для разрабатываемого приложения была выбрана чистая архитектура, то есть приложение будет состоять из трех слоев: графический интерфейс, бизнес-логика, доступ к данным.

### **Средства реализации ПО:**

- Язык программирования: GO;
- Среда разработки: Visual Studio Code;
- Графический интерфейс: HTML, CSS, JS.

9

Рисунок 4.9 – Презентация – слайд 9

## **Исследовательская часть**

Для проведения исследования зависимости времени выполнения запроса при наличии индексов и без был выбран поиск пользователя по почте.

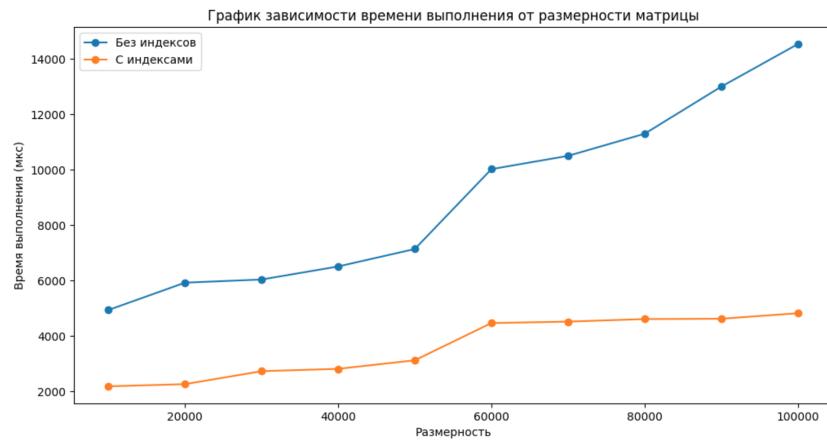
- объем данных: от 10000 до 100000;
- шаг: 10000;
- Количество замеров на каждом шаге: 1000.

В результате эксперимента было установлено, что использование индекса помогает ускорить время запроса примерно в 2-3 раза.

10

Рисунок 4.10 – Презентация – слайд 10

## Исследование зависимости времени выполнения запроса от наличия индексов



11

Рисунок 4.11 – Презентация – слайд 11

## Заключение

В ходе выполнения курсовой работы были выполнены следующие задачи:

- проведен анализ существующих решений;
- сформулированы требования к разрабатываемой базе данных;
- проанализированы существующие базы данных на основе данной задачи;
- спроектирована и разработана база данных;
- спроектировано и разработано приложение для взаимодействия с базой данных;
- проведено исследование зависимости времени выполнения запросов с использованием индексов и без.

Поставленная цель была достигнута.

12

Рисунок 4.12 – Презентация – слайд 12