МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка API интерфейса пользователя с возможностью управления голосовыми командами для IOT устройства

Студент А.И. Таиров
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Митин
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема дипломной работы - Разработка пользовательского интерфейса API с возможностью управления голосовыми командами для IOT-устройства.

Данное исследование посвящено разработке системы с поддержкой голосового ввода команд для взаимодействия с IOT-устройствами.

Настоящая дипломная работа состоит из 19 изображений и рисунков, 1 таблицы, списка из 20 источников и 1 приложения.

Работа разделена на несколько логически связанных частей: введение, четыре раздела, заключение и список использованной литературы.

Во введении объясняется актуальность темы и затронутых проблем. Он также устанавливает цель и задачи, которые необходимо достичь.

В первом разделе исследуются методы взаимодействия с устройствами IOT, сравниваются готовые реализации и анализируется предметная область.

Второй раздел посвящен разработке структурной схемы системы, проведению сравнительного анализа технологий распознавания речи и протоколов передачи данных.

В третьем разделе описывается процесс реализации системы в виде мобильного приложения, а также отдельно рассматривается реализация каждой подсистемы.

В четвертом разделе разрабатывается пользовательский интерфейс, благодаря которому пользователь может взаимодействовать с устройствами IOT. В этом разделе также проводится тестирование рассматриваемого мобильного приложения.

В заключение следует подчеркнуть, что разработанная система представляет собой универсальный способ связи с различными устройствами IOT, а приложение, в котором эта система построена, имеет интуитивно понятный и простой в использовании интерфейс с поддержкой голосовых команд.

Abstract

The topic of this graduation work is *Development of API user interface with an option to control voice commands for an IOT device.*

This research is devoted to developing a system with the support for voice command input for interacting with IOT devices.

The present graduation work consists of 19 figures and drawings, 1 table, a list of 20 references and 1 appendix.

The work is divided into several logically connected parts which are an introduction, four chapters, a conclusion and a list of references.

The introduction explains the relevance of the topic and problems touched upon. It also sets the goal and objectives to be attained.

The first chapter explores the methods of interacting with IOT devices, compares the ready-made implementations and analyzes the subject area.

The second chapter deals with developing a system structural diagram, conducting a comparative analysis of speech recognition technology and data transmission protocols.

In the third chapter the process of implementing the system in the form of a mobile application is described, as well as the implementation of each subsystem is considered separately.

The fourth chapter develops a user interface, due to which a user can interact with IOT devices. This chapter also dwells on testing the mobile application in question.

In conclusion, it should be emphasized that the developed system is a universal way to communicate with various IOT devices, and the application in which this system is built, has an intuitive and easy-to-use interface with the support for voice commands.

Содержание

Введение

Одной из очень популярных бытовых систем является система «Умный дом», являющейся сложной концепцией, которая позволяет объединять самые разнообразные устройства домашнего быта в единый управляемый комплекс. Важной частью данной системы является пользовательский интерфейс. В зависимости от того, как он будет реализован, будет определять удобство пользования системой в целом. Кроме того, существует очень важная категория пользователей, на которых стоит обращать обособленное внимание и выделять их в новую целевую группу. Это пользователи с ограниченными возможностями здоровья. Для таких пользователей особое значение имеет реализация интерфейса в связи с их особенностями [12].

В том случае, если пользовательский интерфейс реализован как отдельная подсистема «Умного дома», то он легко может быть адаптирован к индивидуальным потребностям пользователя с минимальной доработкой остальных подсистем «Умного дома». В соответствии с потребностями пользователя может также использоваться управление голосом, датчики нейронных импульсов, управление жестами и т.д. Многие современные интерфейсы реализуются именно как голосовые, или имеют поддержку голосовых помощников.

Современная техника позволяет достаточно легко реализовывать такие интерфейсы. Такой вид интерфейса необходим пользователям с нарушениями зрения, двигательных функций. Кроме того, во многих ситуациях он может быть более удобен чем стандартный кнопочный интерфейс.

Голосовой интерфейс позволяет отказаться от ввода информации вручную, с помощью клавиатуры или кнопок управления, а также от чтения ответных сообщений. Однако серьезным недостатком такого ввода является низкая точность из-за влияния внешних шумов [21].

Направление голосового ввода в настоящее время очень активно

развивается. С 2014 года в данном направлении появились голосовые помощники. Google и Amazon представили в массы Google Home и Alexa. Эти помощники, представленные в виде колонок, реагируют на довольно обширный диапазон голосовых команд и могут исполнять роль посредника при управлении системами автоматизации .

Целью работы является разработка мобильного приложения для платформы Android с функцией голосового управления системой «Умный дом» и разработка пользовательского интерфейса.

В ходе разработки необходимо решить следующие задачи:

- провести анализ существующих аналогов интерфейса голосового управления;
- сбор и анализ информации о существующих технологиях голосового распознавания и взаимодействия с «Умным домом»;
- проектирование архитектуры и интерфейса мобильного приложения, удовлетворяющего общим стандартам платформы Android;
- реализация мобильного приложения.

1 Сравнительный анализ технологий

1.1 Обзор предметной области

При изучении системы «Умный дом» можно выделить несколько основных типов пользовательского интерфейса, позволяющих пользователю взаимодействовать с системой. Типы интерфейсов могут быть следующими:

- сенсорный или кнопочный интерфейс в виде пульта управления;
- сенсорная панель со встроенным программным обеспечением;
- голосовой интерфейс.

Каждый тип интерфейса имеет определенные недостатки. К примеру, пульт дистанционного управления сильно ограничен диапазоном своего действия, а сенсорная панель является довольно неудобной из-за необходимости установки её в каждой комнате для удобства пользования или необходимости каждый раз переходить в ту комнату, где сенсорная панель установлена [6].

По способу реализации можно выделить 2 основных типа интерфейсов:

- экранный интерфейс, в котором пользователь должен нажимать на определенные области экрана или вводить текст для выполнения определенного действия;
- голосовой интерфейс, распознающий голосовой текст и выполняющий на его основе определенные действия.

Голосовым интерфейсом называют программный продукт, который при помощи голосовой или речевой платформы позволяет пользователю взаимодействовать с системой «Умный дом», отдавая команды для запуска определенных отдельных действий голосом. В соответствии с этим, основными специфичными функциями такого типа интерфейсов являются распознавание голосовой команды, генерация ответного сообщения и воспроизведение ответа голосом [13].

Удобство голосовых интерфейсов особенно сильно проявляется в тех ситуациях, когда ввод текста является сложным или неудобным действием. Например, во время вождения автомобиля пользователю не только неудобно вводить запрос в текстовом виде, но и это опасно для жизни как самого водителя, так и остальных участников дорожного движения. С помощью голосового интерфейса пользователь может продиктовать нужный адрес, запустить какое-то действие, получить информацию, не отвлекаясь от основного занятия [21].

Одним из возможных решений реализации голосового интерфейса внутри помещения могло бы быть размещение микрофонов в различных точках внутри помещения таким образом, чтобы у пользователя была возможность произнести команду из любой точки в любой момент времени и услышать ответ от системы. Однако попытки реализации подобных систем столкнулись с такой проблемой, что для реализации потребовалось большое количество чувствительных микрофонов, которые к тому же необходимо объединить в одну систему, и поэтому такая система становилась дорогостоящей и крайне сложной технически. Кроме того, почти каждый современный человек обладает мобильным устройством, которое он носит с собой, и это делает реализацию голосового интерфейса намного более удобной для пользователя. Исходя из вышесказанного, для управления IoT-устройствами было принято решение реализовать мобильное приложение с поддержкой голосового интерфейса.

1.2 Обзор аналогов разрабатываемой системы

Для сравнения аналогов разрабатываемой системы в первую очередь необходимо определить критерии сравнения систем с голосовым управлением. Для сравнительного анализа были выделены следующие критерии:

- поддержка русского языка;
- возможность создания собственных команд;
- возможность обратного взаимодействия с пользователем с помощью звуковых или текстовых ответных сообщений;
- стоимость.

Для сравнения были выбраны следующие системы: Amazon Echo, Google Home, Apple Siri. Рассмотрим каждую технологию, выявим их преимущества и недостатки и проведем анализ.

1.2.1 Семейство Amazon Echo

Amazon Echo - это серия «умных» устройств, разработанных компанией Amazon. Первое поколение устройств появилось в 2014 году с выходом Bluetooth-колонки Amazon Echo. Amazon Echo - устройства, созданные как голосовые помощники, работающие в режиме «AlwaysOn». Взаимодействие с устройствами происходит только с помощью голоса, без использования кнопок или каких-либо иных интерфейсов. Для распознавания речи в колонках используется собственная технология Amazon - Alexa. Данная линейка устройств может управлять устройствами умного дома, воспроизводить музыку с телефона, Amazon Prime или Spotify, а также заказывать продукты с Amazon [22].

Внешний вид колонки из серии Amazon Echo представлен на рисунке 1.

В Echo второго поколения заявлено лучшее звуковоспроизведение благодаря системе Dolby, которая была представлена летом 2016 года - это позволяет использовать мультикомнатную аудиосистему.

Рисунок 1 - Колонка Amazon Echo

Преимущества:

- возможность голосового оповещения с использованием колонок;
- возможность разработки для IoT-устройств различными способами (Backendess, Storyline);
- поддержка большого количества систем «Умного дома».

Недостатки:

- нет поддержки русского языка;
- высокая стоимость.

1.2.2 Google Home

Google Home - это беспроводной динамик, снабженный голосовым управлением, разработанный в Google. Продукт был объявлен 18 мая 2016 года на Google I/O. Представляет собой смарт-динамик. Google Home стал одним из устройств, поддерживающих работу персонального ассистента

«Google Assistant», наряду с текстовым чатом Allo и видеочатом Duo [4].

Продукт очень похож по своей концепции на Amazon Echo, и является его прямым конкурентом в промышленности «умных» колонок и персональных помощников. Его внешний вид представлен на рисунке 2.

Рисунок 2 - Смарт-динамик Google Home

Часть функциональности смарт-динамика похожа на голосовой помощник Apple Siri.

Преимущества:

– легкость интеграции в существующие решения из-за поддержки

разработчиков сторонних модулей;

- удобный голосовой помощник;
- легко устанавливается;
- хорошее качество звука.

Недостатки:

- высокая стоимость;
- для качественной работы требуется хорошее соединение с интернетом;
- проблемы с русским языком.

1.2.3 Apple Siri

Apple Siri - облачный персональный помощник и вопросно-ответная система, программный клиент которой входит в состав iOS, watchOS, macOS и tvOS компании Apple. Данное приложение использует обработку естественной речи, чтобы отвечать на вопросы и давать рекомендации. Siri приспосабливается к каждому пользователю, изучая его предпочтения в течении долгого времени [9].

Первоначально Siri стало доступно в App Store как приложение для iOS от Siri Inc. Вскоре, 28 апреля 2010 года, Siri Inc. была приобретена Apple Inc. Но ещё до того, как Apple купила Siri, было объявлено, что их программное обеспечение будет доступно для телефонов BlackBerry и телефонов под управлением Android, однако эти планы были отменены [6].

Для взаимодействия Siri и «Умного дома» компания Apple создала проект HomeKit. Устройства, которые могут объединяться с помощью технологии HomeKit, представлены на рисунке 3.

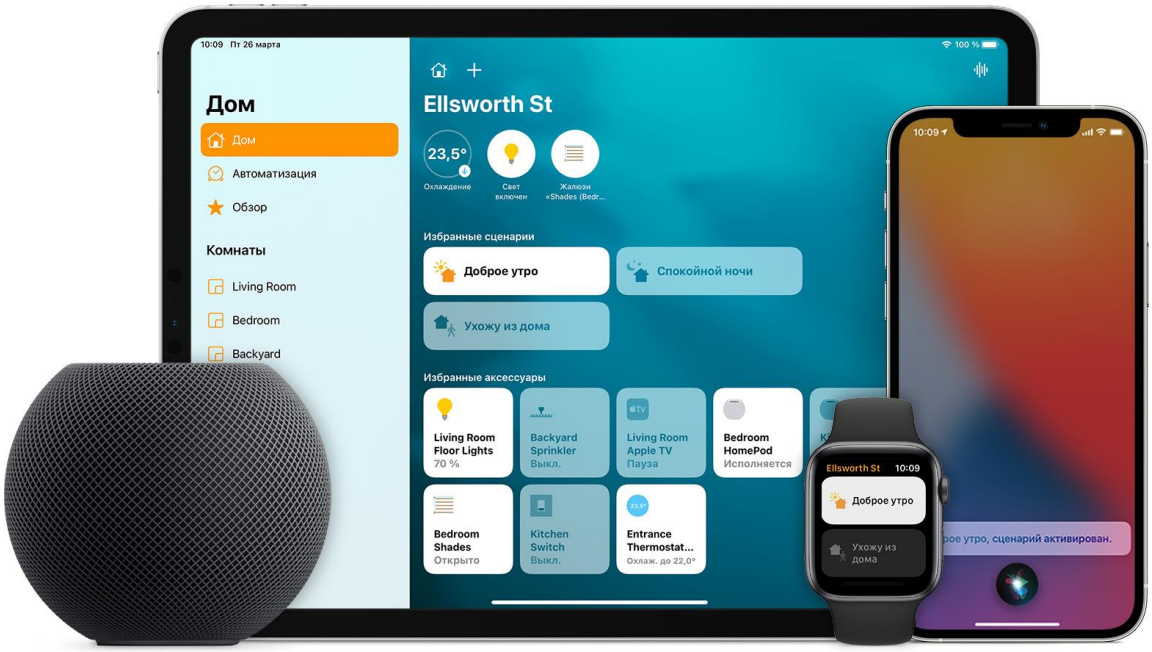

Рисунок 3 - Устройства Apple, работающие через HomeKit

Данная технология обладает следующими возможностями:

– Siri может регулировать работу аксессуаров HomeKit, например световых приборов или термостата;
– при использовании HomePod есть возможность управлять всеми аксессуарами HomeKit в комнате, где находится это устройство, одной командой;
– Siri может включать и выключать аксессуары HomeKit: от освещения до бытовых приборов;
– Siri знает, какие аксессуары HomeKit настроены в программе «Дом», а также их статус. Siri идентифицирует аксессуары по их именам, местоположению и по другим сведениям, указанным для них в программе «Дом». Если настроен HomePod, Apple TV или iPad в качестве домашнего центра, то можно использовать Siri для управления домом в любом месте;
– если аксессуары организованны по комнатам или зонам, можно управлять областями дома одной командой;

– с помощью сценариев можно управлять несколькими аксессуарами одновременно, а настроить сценарий, используя Siri, можно используя только голосовые команды.

Недостатки:

– необходимость использовать устройства только от компании Apple;
– для работы возможно использовать только сертифицированные компанией Apple компоненты «Умного дома».

Выводы по разделу 1

В результате изучения предметной области было принято решение о разработке интерфейса в виде мобильного приложения с пользовательским интерфейсом и поддержкой голосовых команд.

Были изучены аналоги разрабатываемой системы - Amazon Echo, Google Home, Apple Siri. В результате анализа данных систем был сделан вывод о том, что они являются слишком дорогостоящими для внедрения в систему «Умный дом».

2 Разработка приложения

2.1 Функции и подсистемы приложения

В разрабатываемом мобильном приложении будут реализованы следующие функции:

- функция автоматического запуска при старте операционной системы;
- функция распознавания голосовых команд;
- функция соединения и управления «Умным домом»;
- настройка подключения и взаимодействия с устройствами.

Функция распознавания голосовых команд является основной функцией мобильного приложения. Настройки приложения позволяют провести настройку подключения, настройку топиков, пользователя.

В мобильном приложении были выделены следующие подсистемы:

- голосовой интерфейс;
- интерфейс к аппаратной части, включающий в себя функцию соединения с «Умным домом»;
- хранение данных;
- подсистема выполнения команд;
- пользовательский интерфейс.

В результате анализа функций была создана следующая структурная схема, представленная на рисунке 4.

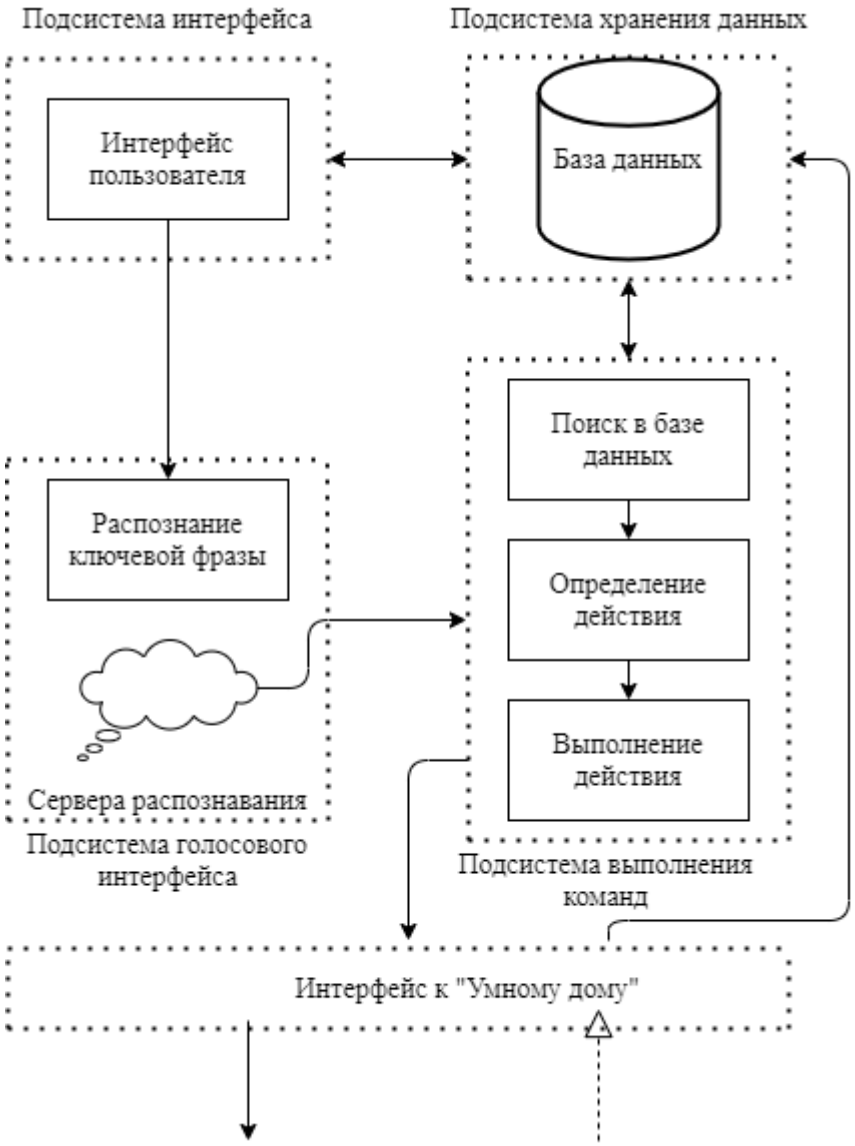

Рисунок 4 - Структурная схема приложения

На данной структурной схеме видно все реализуемые подсистемы, их взаимодействия между собой и содержание каждой из подсистем.

Самой крупной подсистемой является подсистема выполнения команд, так как ей необходимо делать запрос в базу данных, обрабатывать результат этого запроса, определять действие, которое необходимо выполнить, корректно сформировать новый запрос для системы «Умного дома» и отправить его в эту систему. Кроме того, если устройство возвращает

какой-то результат, то подсистеме выполнения команд необходимо его обработать.

2.2 Подсистема голосового интерфейса

Задача разработки голосового интерфейса сводится к реализации системы для распознания речи и внедрения её как подсистемы в приложение. Но на рынке уже существует множество систем распознавания речи, открытых для программистов и которые они могут свободно использовать в разработке. Данные системы используют нейронные сети и быстрые алгоритмы для анализа, поэтому мы можем выбрать систему распознавания голоса из уже существующих.

Для реализации подсистемы голосового интерфейса и реализации функции распознавания голосовых команд необходимо проанализировать уже существующие технологии распознавания речи и выбрать из них самую подходящую под поставленные задачи.

Система обязательно должна поддерживать русский язык, легко встраиваться в систему, быть бесплатной и работать достаточно качественно, чтобы её можно было внедрить в качестве подсистемы.

2.2.1 Анализ технологий распознавания речи

Существует два основных подхода к распознаванию речи.

Первый заключается в онлайн распознавании голоса. Такие фирмы как Google, Яндекс, Apple, Amazon и другие, постоянно совершенствуют свои системы голосового управления умным домом. Эти системы чаще всего используют для своей работы искусственный интеллект: введенная фраза передается на центральный сервер, где обрабатывается и результат отсылается пользователю. Таким образом пользователь общается с системой с помощью голосовых команд. Такой подход требует наличие постоянного подключения к интернету [17].

Второй подход основан на сравнении введенных слов с имеющейся базой. Список слов формируется программистом при настройке системы «Умного дома» и учитывает конкретные пожелания владельца.

В такой системе нужно четко формулировать свои запросы, так как система не способна их интерпретировать. Например, включить свет в зале, выключить телевизор, закрыть шторы и т.д. Минусом такого подхода будет сложность в добавлении функционала без знаний языков программирования. Система голосового распознавания тесно связана с системой голосового вывода, так как распознанные команды желательно куда-то выводить. Это можно делать при помощи вывода информации на дисплей, синтеза речи или специально записанных семплов. В системе с онлайн распознаванием актуально использовать синтез речи или дисплей с текстовой информацией, так как заранее не всегда известно какой результат будет возвращен в ответ на голосовой запрос. В системе, использующей заранее заготовленную базу слов лучше использовать записанные семплы. В такой системе при программировании можно связать голосовой запрос и голосовое сообщение, выводящееся пользователю. Причем в этом случае сообщения будут всегда одинаковыми. Такой подход позволяет создать стабильную и быстро работающую систему, но на разработку этой системы уйдёт много времени [18].

2.2.2 Оценка технологий распознавания речи

При исследовании API для технологий распознавания голоса упор был поставлен прежде всего на наличие русского языка, корректность результатов распознавания и цену продукта.

Под корректностью подразумеваем совпадение результатов распознавания с тестовым набором сообщений. Набор сообщений включает в себя фразы: «Привет, дом», «Включи свет», «Температура».

Для проведения сравнения были проведены тесты, заключающиеся в написании тестовой программы и проведение измерений точности

распознавания речи и времени операции.

Были рассмотрены следующие технологии распознавания речи:

– Речевые технологии SpeechKit - комплекс речевых технологий от российской компании Яндекс, который включает распознавание и синтез речи, голосовую активацию и выделение смысловых объектов в произносимом тексте. Включает в себя Mobile SDK.

SpeechKit Mobile SDK позволяет встроить распознавание и синтез речи, а также голосовую активацию Яндекса в мобильное приложение для iOS, Android и Windows Phone [14].

Если количество голосовых обращений к приложению не превышает 10000 в сутки, использовать SpeechKit Mobile SDK можно бесплатно. Если запросов окажется больше, есть возможность либо оплачивать превышение лимита, либо перейти на тариф бизнес, в котором нет таких ограничений и доступны дополнительные функции. Например, создание специальных тематических модулей и уникальных голосов. Бесплатно для образовательных целей [11].

Преимущества:

- поддержка русского языка;
- безопасность: все сообщения передаются в шифрованном виде;
- работает на основных мобильных платформах.

Недостатки:

- возможны ограничения в работе программных продуктов из-за ограничения количества обращений;
- возможные перебои в работе из-за того, что технология ещё в разработке.

– Google Cloud Speech API - технология распознавания голоса от компании Google.

Google Cloud Speech API позволяет разработчикам преобразовывать аудио информацию в текст, применяя мощные модели нейронных сетей в

простой для использования API с использованием обработки на сервере.

Имеется поддержка множества устройств и программных продуктов. Имеется возможность работы на большинстве современных операционных систем: Android, iOS, Windows 10.

Google Cloud Speech API поддерживает более 110 языков, включая русский. Использование нейронных сетей позволяет быстро преобразовывать голосовую модель в текст [3].

Преимущества:

- бесплатен для использования на платформе Android;
- быстрое распознавание речи и обработка результатов;
- имеется поддержка русского языка;
- возможность распознавания без использования интернета на платформе Android.

Недостатки:

- при использовании на платформе, отличной от Android, вводится плата за обращения к серверу;
- закрытость платформы.

– Alexa Voice Service - технология распознавания голоса от компании Amazon.

Alexa Voice Service (AVS) позволяет интегрировать голосового помощника Alexa непосредственно в программные продукты. Amazon предоставляет доступ к набору ресурсов для быстрого и легкого создания продуктов с поддержкой Alexa, включая API, инструменты для разработки аппаратного и программного обеспечения и документацию. С AVS есть возможность добавить новый интеллектуальный интерфейс к своим продуктам и предложить клиентам доступ к растущему числу функций Alexa и интеграции в «Умные дома» [22].

Преимущества:

- быстрое распознавание и отклик из-за использования серверов по

всему миру.

Недостатки:

- не имеет поддержки русского языка;
- нельзя использовать бесплатно.

– SnowBoy - технология распознавания голоса с открытым исходным кодом под Android, iOS, Windows.

Данная технология позволяет запускать распознавание речи прямо на устройстве без использования сторонних серверов. SnowBoy - использует языковые модели. Для начала распознавания необходимо записать голос человека, который произносит необходимую фразу для распознавания.

Преимущества:

- запуск распознавания без использования серверов, что позволяет постоянно использовать распознавание на устройстве;
- бесплатная технология.

Недостатки:

- требуется составлять языковые модели.

2.2.3 Тестирование систем распознавания речи

Поскольку надежность распознавания речи является одной из самых важных характеристик для приложения, для выбора реализации API было проведено тестирование. Для этого была реализована тестовая программа, которая включает в себя функции:

– взаимодействия с API;
– замера времени распознавания от запуска до конечного результата в виде текста.

Для трех технологий, «SpeechKit», «Google Cloud Speech API» и «SnowBoy», были проведены в общей сумме тридцать тестов. Результаты тестирования представлены в таблице 1. Технология Alexa Voice Service не была включена в тестирование, так как не имеет поддержки русского языка.

В результате тестирования был сделан вывод, что SnowBoy наиболее

подходит для распознавания ключевого слова, а Google Cloud Speech API для распознавания ключевых фраз, поэтому технология Google Cloud Speech API лучше подходит для достижения поставленной цели.

Таблица 1 - Результаты тестирования технологий распознавания речи

Название технологии	**Среднее время распознавания после 10 запусков**	**Точность распознавания после 10 запусков**
SpeechKit	3,9	Неверные результаты, полностью несовпадающие с изначальными тестовыми данными
Google Cloud Speech API	1,6	8 из 10 правильно определенных текстов
SnowBoy	1,2	7 из 10 правильно определенных текстов

Данный вывод сделан из-за того, что для технологии SnowBoy не нужны удаленные сервера для распознавания ключевых слов, но для каждого слова требуется создание собственной языковой модели, поэтому данная технология хорошо подходит для распознавания кодового слова, например «Дом». Google Cloud Speech API хорошо подходит для того, чтобы распознать фразы, которые настроят пользователи, без необходимости каждый раз составлять модели. Поэтому в приложении будет использоваться именно Google Cloud Speech API.

2.3 Выбор протокола обмена данными

Связь между приложением и системой «Умный дом» происходит с помощью протокола обмена данными. Для успешной разработки приложения необходимо провести анализ существующих протоколов обмена данными и выбрать из них подходящий.

На сегодняшний день не существует единой платформы и инфраструктуры для поддержки всех устройств домашней автоматизации. Беспроводные сети представлены множеством технологий, каждая из которых обладает теми или иными преимуществами. Все использующиеся ныне в системах домашней автоматизации протоколы создавались для иных целей или успели сильно устареть.

Но ни одна из сетей не может полностью удовлетворить все запросы разработчиков и потребителей. У каждой из них присутствует как ряд плюсов, так и минусов. Поэтому всё ещё отсутствует повсеместное универсальное решение [14].

Для выбора протокола необходимо сравнить их по следующим критериям:

- отказоустойчивость;
- достаточно легкая интеграция в систему;
- возможные форматы передачи данных;
- возможность администрирования;
- безопасность;
- нагрузка на сеть.

Сравним наиболее часто применяемые протоколы, их достоинства и недостатки.

2.3.1 MQTT

MQTT (Message Queue Telemetry Transport) - это упрощенный сетевой протокол, работающий поверх TCP/IP. Используется для обмена сообщениями между устройствами по принципу издатель-подписчик. Идеален для использования в контроллерах и датчиках, где требуется небольшой размер кода и есть ограничения по пропускной способности канала.

Протокол MQTT работает на прикладном уровне поверх TCP/IP и использует по умолчанию 1883 порт (8883 при подключении через SSL).

Также, возможна работа через Winsocket, что позволяет адаптировать его на многие платформы.

Обмен сообщениями в протоколе MQTT осуществляется между клиентом, который может быть издателем, подписчиком или брокером сообщений.

Издатель отправляет данные в брокер, указывая в сообщении определенную тему, топик. Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики. Клиент может быть одновременно и подписчиком, и издателем сообщений [7].

Общий вид связи протокола с устройствами представлен на рисунке 5.

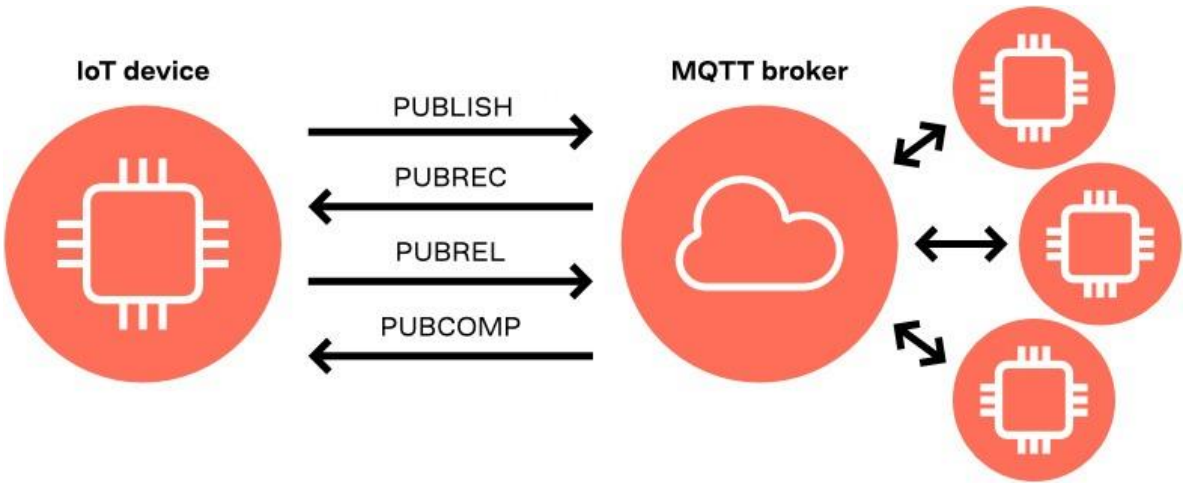

Рисунок 5 - Связь устройств с помощью MQTT

Преимущества:

- прост в использовании;
- открытый стандарт;
- шаблон проектирования издатель-подписчик удобен для большинства решений с датчиками;
- легок в администрировании;
- снижена нагрузка на канал связи;

– работа в условиях постоянной потери связи или других проблем на линии;
– нет ограничений на формат передаваемых данных.

Недостатки:

– из-за отсутствия ограничения формата данных возможно несоответствие формата данных у разных производителей оборудования;
– слабая безопасность [20].

2.3.2 XMPP

XMPP был разработан для системы мгновенного обмена сообщениями для связи между людьми с помощью текстовых сообщений. XMPP означает Extensible Messaging and Presence Protocol, или расширяемый протокол обмена сообщениями и информацией о присутствии. В протоколе XMPP используется текстовый формат XML в качестве встроенного типа, обеспечивая естественную связь между людьми. Протокол работает по TCP/IP [16].

Связь устройств с помощью данного протокола представлена на рисунке 6.

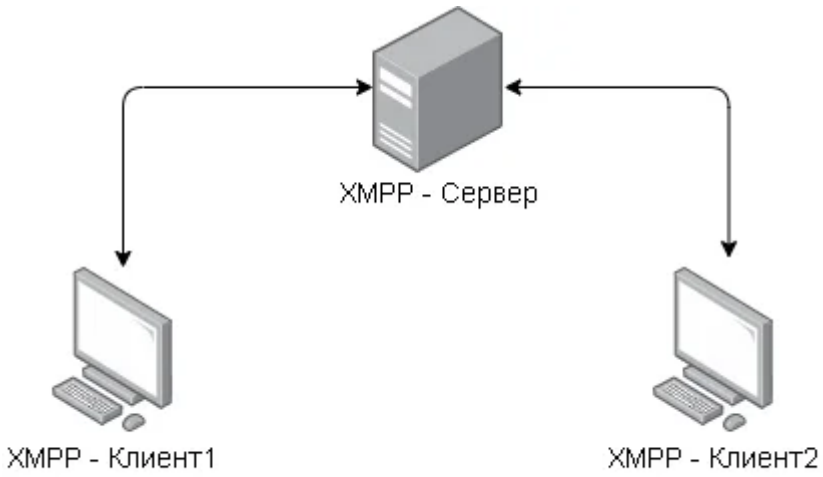

Рисунок 6 - Связь устройств с помощью XMPP

Преимущества:

- открытый стандартам;
- безопасность: XMPP серверы могут быть изолированы от публичных сетей XMPP и хорошо защищены;
- децентрализация: Архитектура сети XMPP схожа с электронной почтой; кто угодно может запустить свой собственный XMPP-сервер и нет какого-либо центрального сервера.

Недостатки:

- избыточность передаваемой информации: как правило, более 70% трафика XMPP составляют сообщения о присутствии, около 60% которых являются излишними, что создает лишнюю нагрузку на канал связи;
- неэффективность передачи бинарных данных: так как XMPP является, по сути, одним длинным XML-документом, невозможно передать не модифицированную двоичную информацию. В результате этого, для передачи файлов стараются использовать дополнительные протоколы, например, HTTP.

2.3.3 AMQP

AMQP (Advanced Message Queuing Protocol) - открытый протокол для передачи сообщений между компонентами системы. Основная идея состоит в том, что отдельные подсистемы (или независимые приложения) могут обмениваться произвольным образом сообщениями через AMQP-брокер, который осуществляет маршрутизацию, возможно гарантирует доставку, распределение потоков данных, подписку на нужные типы сообщений.

AMQP фигурирует тремя понятиями:

Сообщение - единица передаваемых данных, основная его часть никак не интерпретируется сервером, к сообщению могут быть присоединены

структурированные заголовки.

Точка обмена - в нее отправляются сообщения. Точка обмена распределяет сообщения в одну или нескольких очередей. При этом в точке обмена сообщения не хранятся. Точки обмена бывают трех типов:

- fanout - сообщение передается во все прицепленные к ней очереди;
- direct - сообщение передается в очередь с именем, совпадающим с ключом маршрутизации;
- topic - нечто среднее между fanout и direct, сообщение передается в очереди, для которых совпадает маска на ключ маршрутизации.

Очередь - здесь хранятся сообщения до тех пор, пока не будут забраны клиентом. Клиент всегда забирает сообщения из одной или нескольких очередей [15].

Пример связи устройств с помощью данного протокола представлен на рисунке 7.

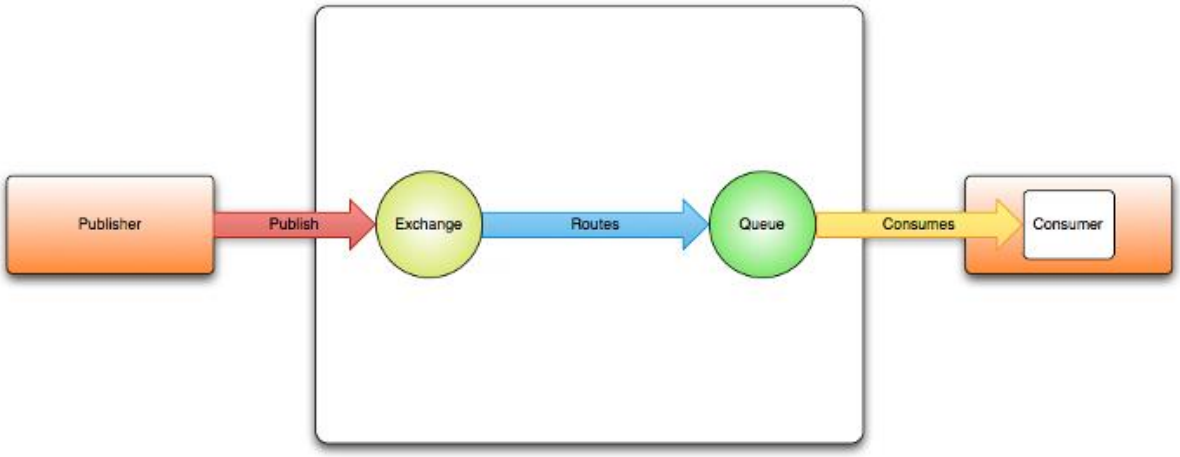

Рисунок 7 - Связь устройств с помощью протокола AMQP

Преимущества:

- открытый стандарт

- разработан для исключения потерь данных между устройствами
- ориентирован на банковские системы, что дает предположения о его надежности и безопасности в использовании.

Недостатки:

- сложность в реализации и обслуживании.

Выводы по разделу 2

На основе анализа, произведенного в первой главе, были выявлены основные требования к разрабатываемой системе, на основе которых было четко сформировано задание и разработана структурная схема разрабатываемого приложения, был проведен сравнительный анализ технологий распознавания речи и протолоков передачи данных.

3 Реализация приложения

3.1 Реализация голосового интерфейса

Для реализации приложения необходимо реализовать следующие основные модули:

- подсистема распознавания речи;
- подсистема пользовательского интерфейса;
- подсистема выполнения команд;
- подсистема хранения данных.

Реализация приложения выполнена на языке программирования Java в среде IntellijIDEA.

Для реализации голосового интерфейса и распознавания команд пользователя была использована технология Google Cloud Speech API. Google Cloud Speech API может работать как с использованием интернета, так и без него, при условии установленного языкового пакета в ОС Android. При этом технология не требует создания никаких дополнительных языковых моделей.

После запуска приложения у пользователя есть возможность управлять приложением с помощью голосового интерфейса при нажатии одной кнопки. Голосовой интерфейс позволяет пользователю перейти на экран настроек, изменить имя и пароль пользователя, адрес подключения к умному дому, список топиков, на которые он хочет быть подписан, подключиться к системе «Умного дома» с помощью протокола MQTT.

Голосовой интерфейс представлен на рисунке 8.

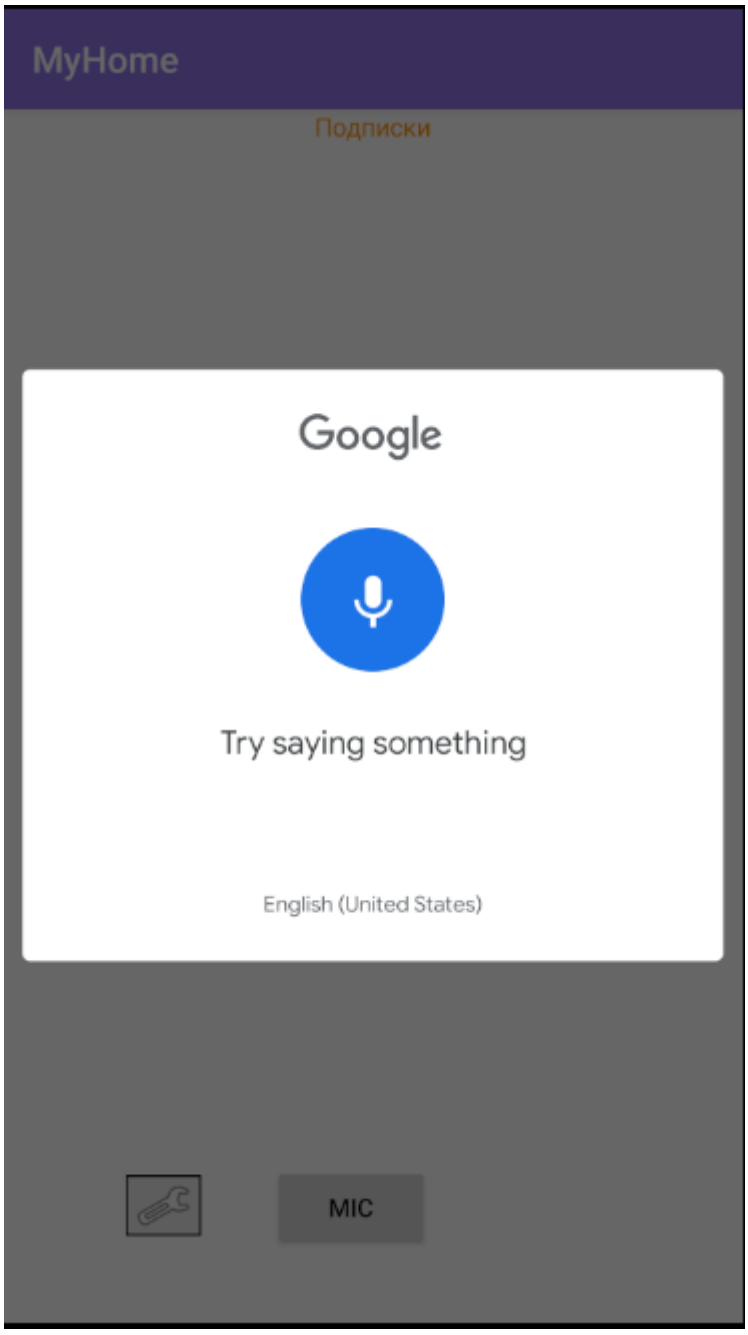

Рисунок 8 - Функция распознавания ключевой фразы

Для выполнения любого действия необходимо четко произнести ключевую фразу после нажатия на соответствующую кнопку для вызова микрофона. В том случае, если ключевая фраза не была распознана, приложение ничего не выполняет и продолжает свою работу.

3.2 Реализация интерфейса к аппаратной части

Для взаимодействия с «Умным домом» был выбран протокол MQTT. В качестве открытой реализации протокола для ОС Android взят проект Paho MQTT. Данный проект сочетает в себе весь необходимый функционал для взаимодействия с брокером MQTT.

Для подключения приложения к MQTT брокеру используется метод connect(), в который передаются параметры введенные пользователем:

- адрес подключения,
- имя пользователя,
- пароль.

После подключения приложение начинает считывать с брокера все сообщения вида название топика и значение и выводить их на экран.

Экран настроек для подключения приложения к системе «Умного дома» с помощью MQTT представлен на рисунке 9.

Кроме изменения параметров подключения, на данном экране пользователь может включить параметр, позволяющий программе подключаться к системе «Умный дом» при запуске приложения и перейти на экран со списком топиков, чтобы подписаться.

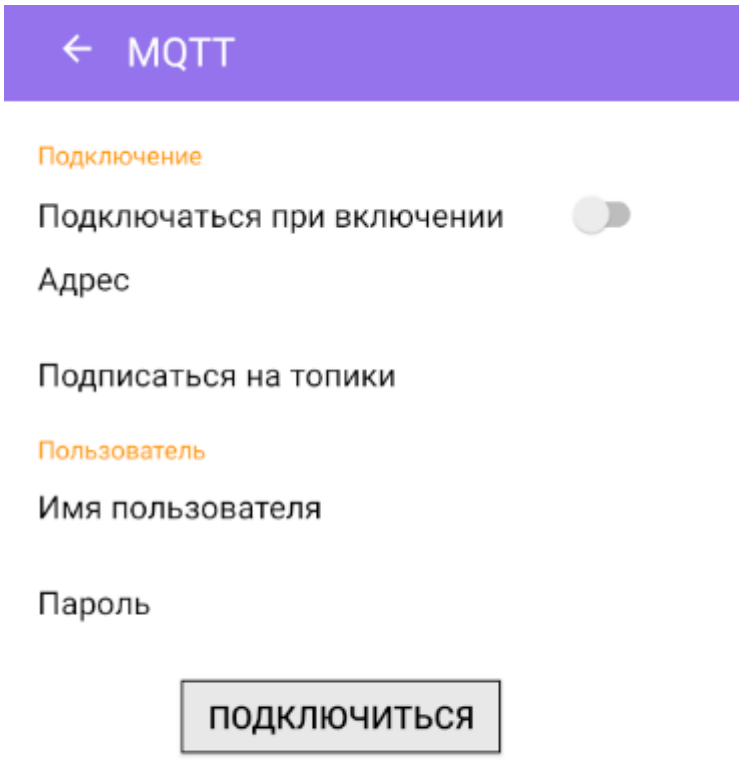

Рисунок 9 - Экран настроек MQTT

Для отключения вызывается метод disconnect(), где происходит отписка от топиков и разрыв соединения с брокером. Значения параметров сохраняются между запусками приложения.

3.3 Реализация базы данных

Для хранения пользовательских данных и топиков MQTT используется встроенная в Android база данных SQLite 3. Данная база данных имеет функции создания таблиц, создания и изменения данных и использования поиска по параметрам. В базе данных для нашего приложения созданы 3 таблицы. Каждая таблица предназначена для хранения строго определенных

вещей:

1. Для хранения данных подключения по протоколу MQTT (таблица mqtt).
2. Для хранения данных о пользователе (таблица user).
3. Для хранения данных о действиях (таблица actions).

Для каждого топика MQTT в базе данных создается запись с определенным id, который сопоставляется с данным топиком во всех таблицах, что позволяет не создавать повторные записи с данным топиком и отсеивать их по нужным параметрам.

Для хранения данных MQTT была создана таблица mqtt, в ней содержится:

1. Название топика.
2. Текущее значение.
3. Активность топика.
4. Параметр отображения топика на начальном экране.
5. Альтернативное имя топика.
6. Подписка на топик.
7. Назначенная команда.

Список команд хранится в отдельной таблице actions, структура которой представлена на рисунке 10.

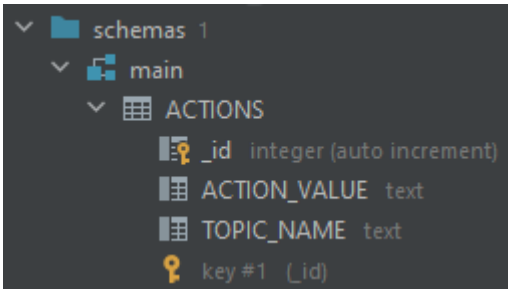

Рисунок 10 - Структура таблица actions

Таблица actions содержит поле id для каждого действия, значение

действия и поле topic_name в качестве внешнего ключа для связи действий с топиками.

Таблица user содержит в себе следующие поля:

1. уникальный идентификатор пользователя;
2. имя пользователя;
3. пароль;
4. адрес подключения;
5. параметр, подключающий приложение при запуске.

Для реализации базы данных и работы с ними были созданы 4 класса ActionDBHelper, MQTTDBClear, MQTTDBHelper, UserDBHelper. Каждый класс связан с соответствующей таблицей, кроме класса MQTTDBClear, который служит только для очистки таблицы, содержащей все топики. Все остальные классы наследуются от SQLiteOpenHelper и переопределяют его методы onCreate() и onUpgrade(). Пример кода одного из классов представлен в приложении А. Остальные классы отличаются только выражениями sql для соответствующих таблиц.

Метод onCreate() создаёт новую таблицу в базе данных в том случае, если она отсутствует и подключается к ней, если она существует.

Метод onUpgrade() уничтожает соответствующую таблицу и вызывает метод onCreate() для создания обновленной таблицы.

3.4 Реализация подсистемы выполнения команд

В подсистеме выполнения команд происходит распознавание ключевой фразы, её сравнение с существующими ключевыми фразами и выполнение всех необходимых действий со значением топика.

Блок-схема алгоритма работы подсистемы представлена на рисунке 11.

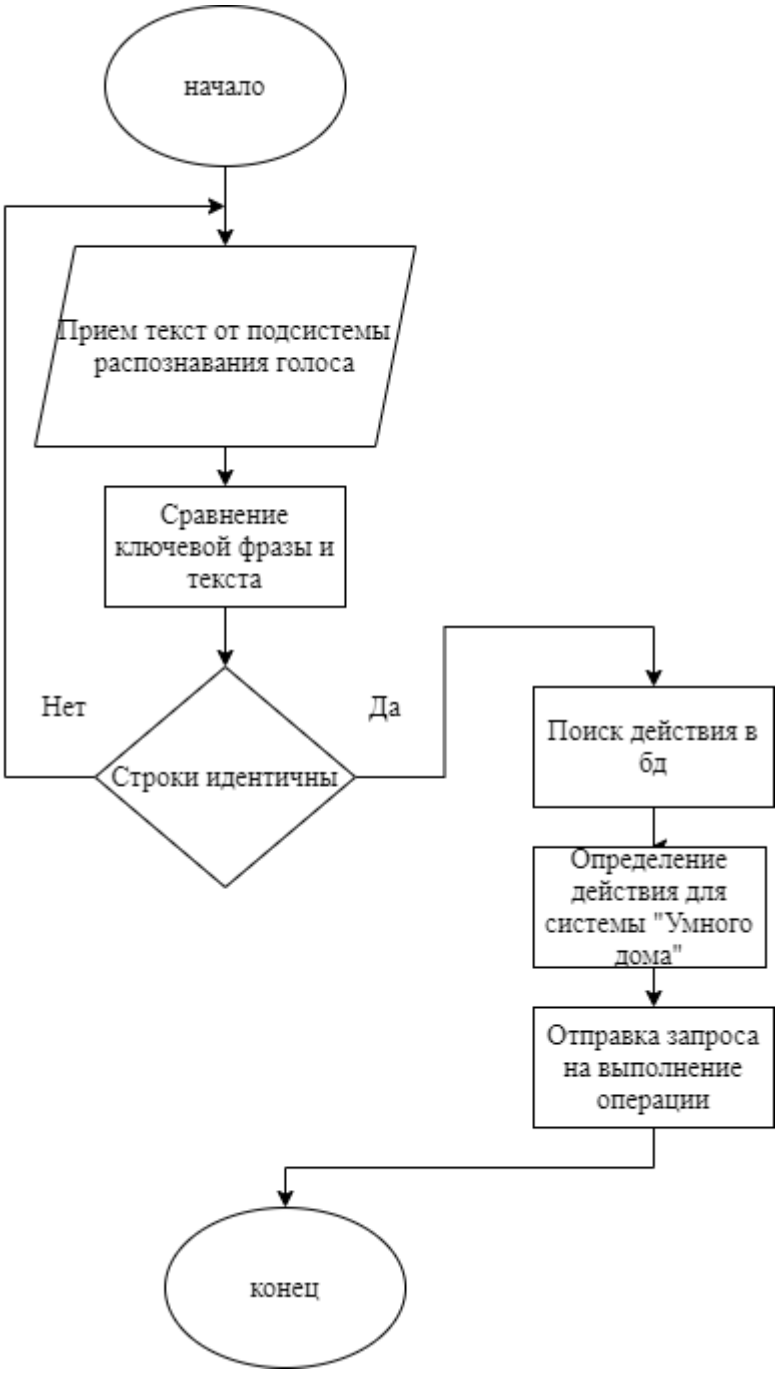

Рисунок 11 - Алгоритм работы подсистемы выполнения команд

В том случае, если произнесенная фраза не соответствует ни одной из ключевых фраз, пользователю необходимо заново произнести её. В случае успешного распознавания ключевой фразы, приложение формирует запрос в систему «Умный дом» и передает его по протоколу MQTT.

Выводы по разделу 3

На данном этапе было реализовано мобильное приложение. Реализация была осуществлена с помощью языка программирования Java и среды разработки IntellijIDEA. В первую очередь был реализован голосовой интерфейс с использованием технологии Google Cloud Speech API.

Для взаимодействия с системой «Умный дом» следующим был реализован интерфейс к аппаратной части, задача которого заключается в установке соединения с системой «Умный дом» или MQTT-брокером.

Необходимой системой для успешной работы приложения является подсистема хранения данных, содержащая в себе базу данных, состоящую из трех таблиц и вспомогательные классы для взаимодействия с базой.

Заключительной подсистемой, которая необходима приложению для корректной работы, является подсистема выполнения команд, основная задача которой заключается в отправке и обработке сообщений с помощью топиков, или использование действий из базы данных.

4 Разработка интерфейса приложения

4.1 Пользовательский интерфейс

Для взаимодействия с приложением был разработан пользовательский интерфейс с поддержкой голосового ввода.

Мобильное приложение можно условно разделить на три основных экрана пользовательского интерфейса и одну фоновую задачу:

1. Экран «Избранное».
2. Экраны настроек.
3. Уведомление в панели уведомлений ОС Android.
4. Фоновая задача подключения к системе «Умный дом».

Мобильное приложение поддерживает русский язык, так как оно направлено в первую очередь на русскоязычную аудиторию.

Внешний вид начального экрана, который видит пользователь при запуске экрана, представлен на рисунке 12.

Рисунок 12 - Начальный экран приложения

При включении в настройках топика параметра «отображать на стартовой странице», внешний вид начального экрана меняется на тот, который представлен на рисунке 13.

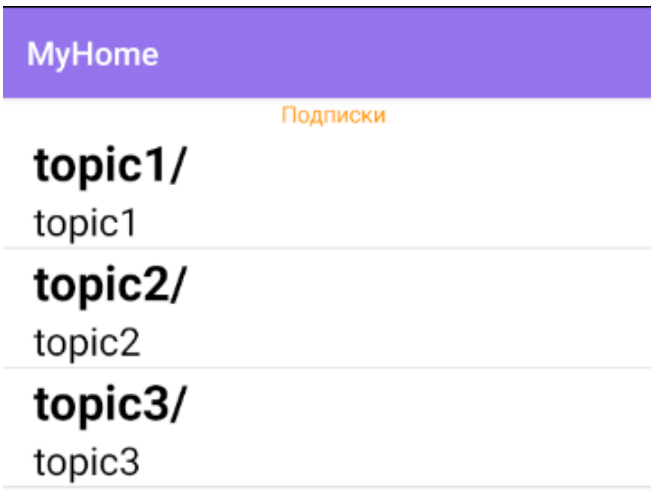

Рисунок 13 - Начальный экран с подписанными топиками

Уведомление в панели уведомлений выглядит как обычное сообщение и не мешает пользователю работать. После выключения приложения уведомление пропадает.

4.2 Настройки приложения

Важным пунктом приложения являются настройки. Экран настроек представляет из себя меню, через которое можно перейти в основные настройки, в которых можно произвести настройку топиков или удалить их, и настройки MQTT, в которых можно настроить подключение к брокеру.

Пункт «Основные настройки» содержит в себе три пункта:

1. Список всех топиков.
2. Создание нового топика.
3. Сброс базы данных.

Пункт «Список всех топиков» содержит отображает на экране список всех хранящихся в базе данных топиков. При нажатии на любую запись происходит переход к настройкам выбранного топика. Экран настройки конкретного топика представлен на рисунке 14.

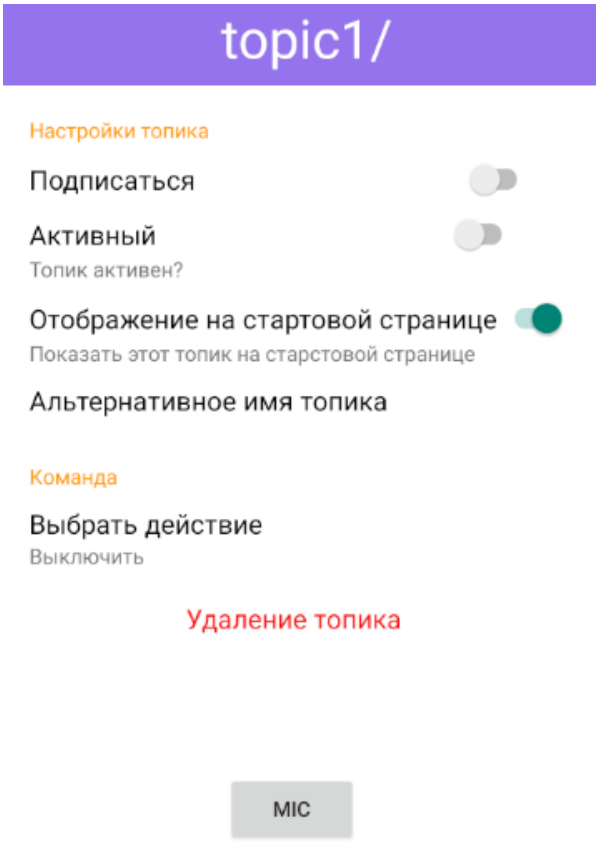

Рисунок 14 - Настройки конкретного топика

На данном экране можно изменить следующие настройки:

- подписка на данный топик;
- активное прослушивание топика;
- отображение на стартовой странице;
- задать альтернативное имя топика;
- выбрать выполняемое действие.

Кроме того, на данном экране присутствует возможность вызвать голосовой интерфейс, чтобы вызвать экран отправки сообщения. Данный экран представлен на рисунке 15.

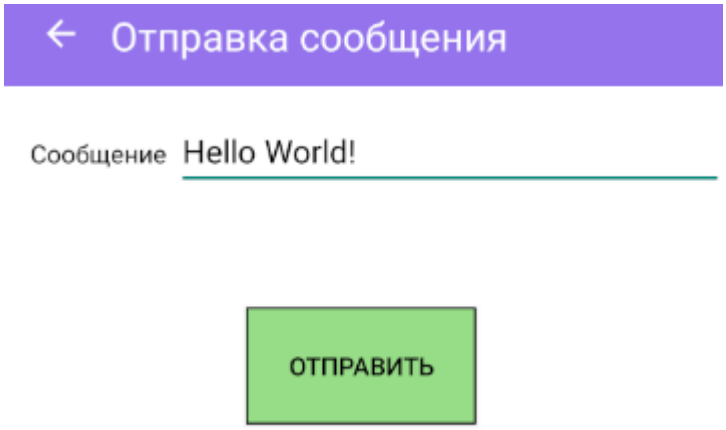

Рисунок 15 - Экран отправки сообщения

При отправке сообщения пользователь получает уведомление как на рисунке 16.

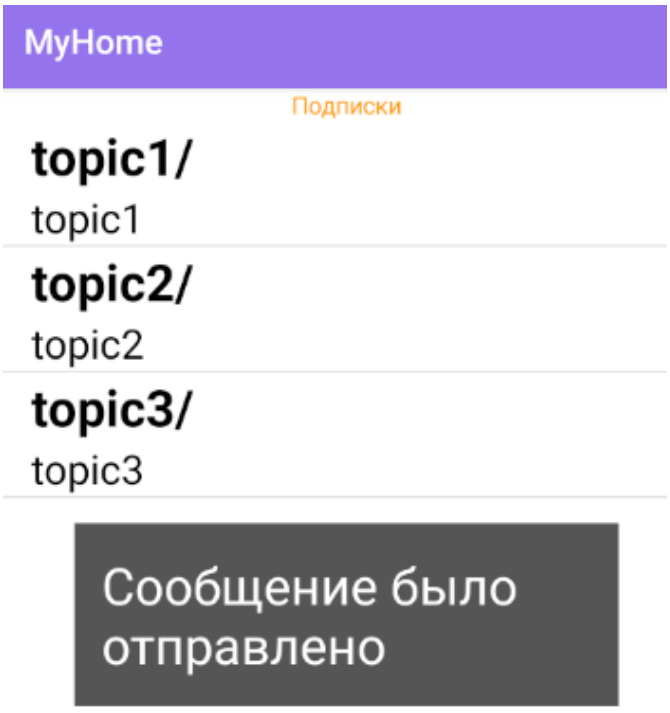

Рисунок 16 - Уведомление об отправки сообщения

Пункт «Создание нового топика» отображает на экране два текстовых

поля для ввода пользователем имени топика и его значения соответственно.

После добавления топика пользователь получает сообщение о добавленном топике, и топик отображается в списке.

Экран с созданием нового топика представлен на рисунке 17.

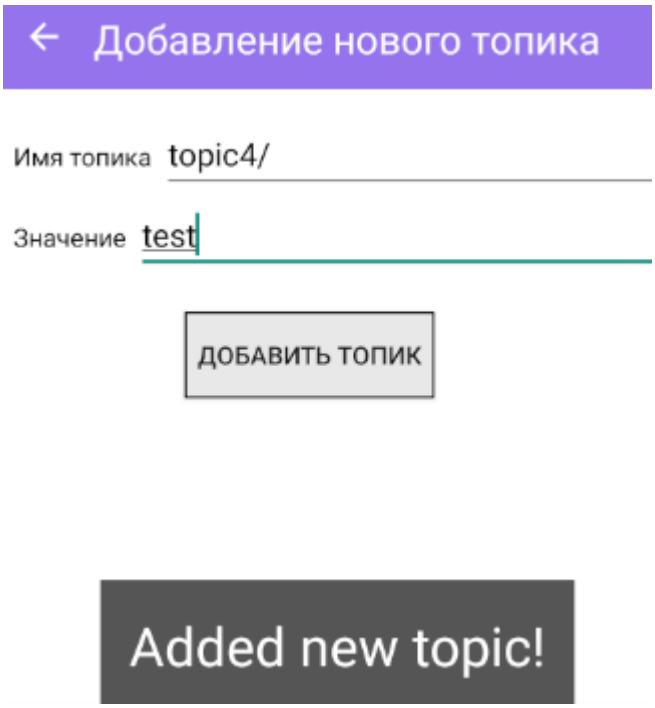

Рисунок 17 - Добавление нового топика

Пункт «Сброс базы данных» переходит на экран, содержащий одну кнопку, которую необходимо нажать для подтверждения действия. После подтверждения, пользователь получает сообщение об успешной очистке базы данных, а все топики из базы удаляются. Экран с подтверждением представлен на рисунке 18.

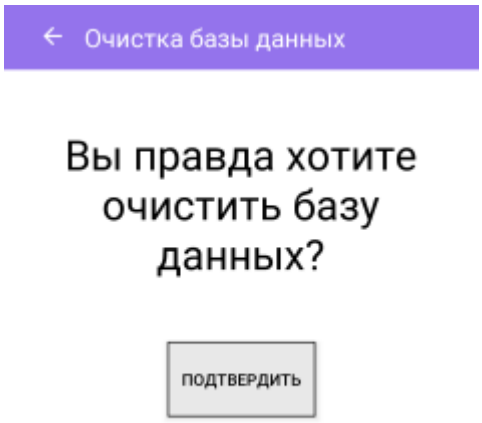

Рисунок 18 - Подтверждение очистки базы данных

Пункт «Настройки MQTT» содержит следующие пункты:

1. Параметр запуска подключения при старте приложения.
2. Адрес подключения.
3. Ссылка на страницу с топиками для дальнейшей подписки.
4. Имя пользователя.
5. Пароль.
6. Ручное подключение к системе «Умный дом»

Экран настроек MQTT с заполненными полями для подключения и нажатой кнопкой представлен на рисунке 19.

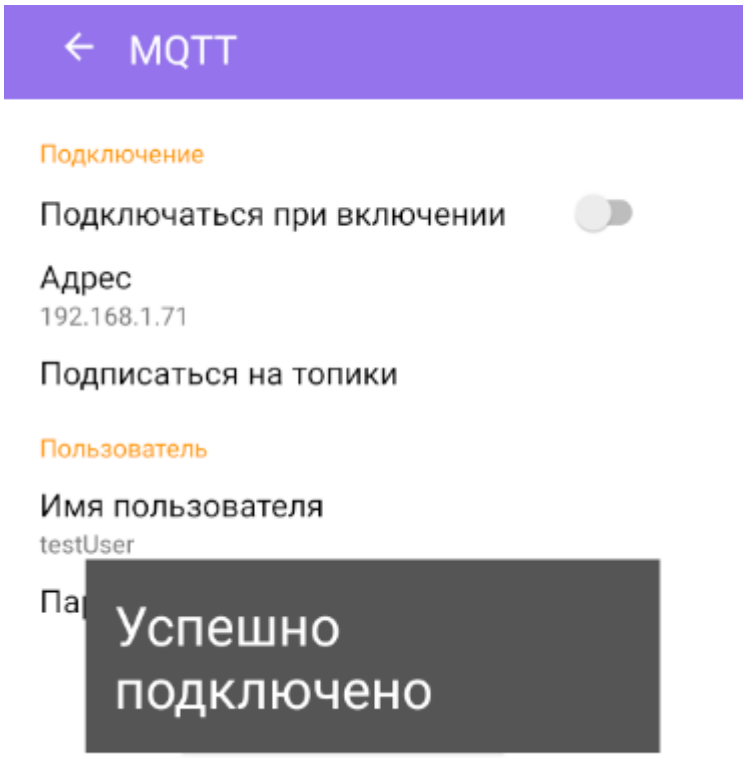

Рисунок 19 - Подключение через настройки MQTT

Следующим этапом является тестирование приложения для проверки его функциональных возможностей и оценки работоспособности.

4.3 Тестирование приложения

Для тестирования приложения была создана искусственная система наподобие системы «Умный дом», являющейся MQTT-брокером, способной принимать, отправлять, обрабатывать сообщения и создавать новые топики. Были проведены следующие тесты.

Настройка подключения, с вводом корректного и ошибочного адреса подключения. В результате ввода ошибочного адреса подключение не было совершено и брокер никак не реагировал на попытки отправить сообщения, а в приложении не отображались входящие сообщения на соответствующем

топике. Для корректного адреса соединение было успешно установлено, обмен сообщениями работал корректно.

Настройка топиков MQTT с вводом одинаковых ключевых фраз, в результате программа сообщила об ошибке, что невозможно использовать ключевую фразу дважды.

Произнесение ключевых фраз для каждого теста. При произнесении «Подключиться» приложение выдало ошибку, так как в настройках MQTT был неправильно указан адрес подключения, все остальные тесты с переходом между экранами, созданием нового топика, отправкой сообщения прошли успешно.

В ходе тестирования приложения получены следующие результаты, что приложение не может определить кодовую фразу при очень тихой речи или при большом количестве шумов. Также были замечены случайные срабатывания созвучных ключевых фраз, но критического влияния на систему «Умный дом» этого не оказывает. Для того, чтобы избежать случайного срабатывания системы было принято решение в дальнейшем добавить функцию подтверждения потенциально опасных команд.

Выводы по разделу 4

Последним этапом для реализации мобильного приложения является разработка интерфейса для пользователя. Данный интерфейс был реализован и представлен на рисунках 12-19.

Для проверки работоспособности приложения оно было протестировано, и результаты тестирования представлены в разделе 4.3.

Заключение

Цель выпускной квалифицированной работы заключалась в разработке API-интерфейса для управления IoT-устройствами. Для достижения цели необходимо было решить следующие задачи:

- провести анализ существующих аналогов интерфейса голосового управления;
- сбор и анализ информации о существующих технологиях голосового распознавания и взаимодействия с «Умным домом»;
- проектирование архитектуры и интерфейса мобильного приложения, удовлетворяющего общим стандартам платформы Android;
- реализация мобильного приложения.

В ходе решения задач были получены следующие результаты:

Был проведен анализ существующих аналогов интерфейса голосового управления. Для анализа были выбраны три технологии-Google Home, Amazon Echo и Apple Siri. В результате сравнения был сделан вывод, что данные технологии являются слишком дорогостоящими для интеграции с системой «Умный дом»;

Был проведен сбор и анализ информации о технологиях голосового распознавания. В результате анализа и тестирования с помощью простой программы был сделан вывод о том, что Google Cloud Speech API лучше всего подходит для распознавания ключевых фраз и команд, а технология SnowBoy лучше подходит для распознавания ключевого слова.

На следующем этапе было спроектировано мобильное приложение и был проведен сравнительный анализ между протоколами передачи данных. Для сравнения были выбраны протоколы MQTT, XMPP и AMQP, определены критерии сравнения, и проведен анализ каждого протокола. С помощью данного исследования было установлено, что для взаимодействия с системой «Умный дом» и датчиками лучше всего подходит протокол MQTT из-за его

универсальности в плане форматов передаваемых данных и отсутствия избыточности данных.

После проектирования структуры приложения и выбора протокола передачи данных, приложение было реализовано на языке программирования Java в среде разработки IntellijIDEA с использованием базы данных SQLite 3. Кроме того, был разработан пользовательский интерфейс, предоставляющий пользователю возможность ввода информации с помощью голоса или используя кнопки, текстовые поля и переключатели.

Приложение было протестировано и показало свою работоспособность в искусственных условиях с использованием MQTT-брокера, способного обрабатывать, принимать и отправлять сообщения и имитировать работу системы «Умный дом».

Поставленные задачи были решены и цель бакалаврской работы была достигнута.

Список используемых источников

1. Biskup A. How Do Smart Homes Work / A. Biskup. 2021. 48 с.
2. Chin R. A DIY Smart Home Guide: Tools for Automating Your Home Monitoring abd Security Using Arduino, ESP8266, and Android / R. Chin. 2020. 304 с.
3. Cloud Speech-To-Text [Электронный ресурс] // Google Cloud. - Режим доступа: http://cloud.google.com
4. Google Home [Электронный ресурс] // Google Cloud. - Режим доступа: http://cloud.google.com
5. HomeKit [Электронный ресурс] // Apple. - Режим доступа: http://www.apple.com
6. Kennedy J., Strengers Y. The Smart Wife / Y. Strengers, J. Kennedy. 2020. 304 с.
7. MQTT and MQTT-SN messaging protocols [Электронный ресурс] // Eclipse Paho. - Режим доступа: http://www.eclipse.org
8. RabbitMQ: Введение в AMQP [Электронный ресурс] // Habrahabr. - Режим доступа: http://habrahabr.ru
9. Siri [Электронный ресурс] // Национальная библиотека им. Н.Э.Баумана. - Режим доступа: http://ru.bmstu.wiki
10. Speak to Alexa on Products with Alexa Voice Service Built-In [Электронный ресурс] // Alexa Voice Service. - Режим доступа: http://developer.amazon.com
11. Vandome N. Smart Homes in easy steps / N. Vandome. 2018. 192 с.
12. Young C. Smart Home / C. Young. 2019. 516 с.
13. Голосовой интерфейс [Электронный ресурс] // Теплица социальных технологий. - Режим доступа: http://te-st.ru
14. Комплекс речевых технологий Яндекса [Электронный ресурс] // Яндекс. - Режим доступа: http://tech.yandex.ru

15. Протоколы «Интернета вещей»: основные сведения [Электронный ресурс] // Средства и системы автоматизации. - Режим доступа: http://www.rtsoft.ru
16. Протоколы связи для «умного дома» [Электронный ресурс] // Аналитические обзоры компьютеров. - Режим доступа: http://www.ferra.ru
17. Распознавание речи для чайников [Электронный ресурс] // Habrahabr. - Режим доступа: http://habrahabr.ru
18. Сверхбыстрое распознавание речи без серверов на реальном примере [Электронный ресурс] // Habrahabr. - Режим доступа: http://habrahabr.ru
19. Технологии распознавания речи [Электронный ресурс] // ПостНаука. - Режим доступа: http://postnauka.ru
20. Что такое MQTT и для чего он нужен в IoT [Электронный ресурс] // Промышленные компьютеры. - Режим доступа: http://ipc2u.ru
21. Что такое голосовые технологии [Электронный ресурс] // Теплица социальных технологий. - Режим доступа: http://te-st.ru
22. Экосистема Amazon Alexa [Электронный ресурс] // Geektimes. - Режим доступа: http://geektimes.ru
23. Языки для «Умного дома» [Электронный ресурс] // Бизнес новости. - Режим доступа: http://www.forbes.ru

Приложение А

Класс для работы с базой данных

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class MQTTDBHelper extends SQLiteOpenHelper {
    public MQTTDBHelper(Context context) {
        super(context, "mqtt", null, 4);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table MQTT ("
                + "_id integer primary key autoincrement,"
                + "NAME text,"
                + "VALUE text,"
                + "ACTIVE integer,"
                + "DASHBOARD integer,"
                + "ALTER_NAME text,"
                + "SUBSCRIBE integer,"
                + "ACTION_VALUE text" + ");");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS MQTT");
        onCreate(db);
    }
}
```