Министерство образования и науки Российской федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
_____ К.А. Домбровский
«___»_____ 2017 г.

Разработка модуля голосового управления персональным компьютером

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 09.03.01.2017.382.ПЗ ВКР

Руководитель работы,
доцент каф. «Электронные
вычислительные машины»
_____ Ю.Г. Плаксина
«___»_____ 2017 г.

Авторы работы
студенты группы КЭ-445
_____ М.В. Чулков
_____ Р.А-Р. Хазбулатов
«___»_____ 2017 г.

Нормоконтролёр, ст. преп. каф.
«Электронные вычислительные
машины»

_____ В.В. Лурье
«___»_____ 2017 г.

Челябинск 2017

АННОТАЦИЯ

Чулков М. В., Хазбулатов Р. А-Р. Разработка модуля голосового управления персональным компьютером. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2017, 47 с., 26 ил., библиогр. список – 18 наименований.

В выпускной квалификационной работе представлена технология разработки модуля голосового управления для персонального компьютера.

В данной работе были рассмотрены аналоги и альтернативные решения разрабатываемого модуля. Были определены критерии, которым должен соответствовать разрабатываемый модуль.

В ходе выполнения выпускной квалификационной работы были спроектированы основные модули приложения, разработаны формы отображения, добавления, редактирования и удаления команд, а также реализована функция, преобразующая голосовое сообщение в текст.

Данная работа является актуальной, так как позволяет пользователям (в том числе и с ограниченными возможностями) удаленно управлять персональным компьютером с помощью голосовых команд.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время возрастает сложность управления ИТ-инфраструктурой. Для людей, имеющих ограниченные возможности, выполнение простых операций с персональным компьютером затруднительно. В этом случае им может помочь программа, позволяющая управлять компьютером с помощью голоса.

Разработка таких программ началась в конце 90-х годов. Например, отечественный программный комплекс для управления компьютером «Горыныч» созданный в 1997 году. Уже тогда, данный программный продукт обладал функциями диктовки, голосовым управлением отдельных функций операционной системы Windows, управлением функциями текстовых редакторов и прикладных программ [1].

Со временем, интерес к такого рода программам стал расти. Он подкреплялся выпуском голосовых ассистентов для мобильных устройств на платформах Android, iOS и Windows Phone, таких, как Google Now, Siri и Cortana, соответственно.

На персональных компьютерах, в настоящее время, таких программ существует относительно немного и все они имеют определенные недостатки, поэтому создание программы такого рода является актуальной задачей.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Цель выпускной квалификационной работы

Целью выпускной квалификационной работы является разработка модуля голосового управления для персонального компьютера. Разрабатываемое приложение нацелено на аудиторию, которая активно пользуется персональным компьютером, но по тем или иным причинам, испытывают трудности в управлении.

Приложение должно иметь возможность гибкой настройки выполняемых функций.

1.2 Задачи выпускной квалификационной работы

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ существующих аналогов и альтернативных решений;
2. Составление технического задания;
3. Проектирование и разработка приложения:

3.1. Построение диаграммы использования;

3.2. Построение объектной модели;

3.3. Выбор инструментов для разработки с учетом необходимости гибкой архитектуры и возможностью последующего расширения приложения;

3.4. Разработка программного кода приложения;

3.5. Тестирование работы приложения.

1.3 Конечный результат выпускной квалификационной работы

Результатом выполнения выпускной квалификационной работы будет программа с функцией удаленного управления функционалом компьютера.

Программа должна удовлетворять следующим требованиям:

- однопользовательское настольное приложение;
- редактируемый словарь команд;
- быстрое и точное распознавание голоса;
- возможность диктовки текста;
- возможность последующего расширения системы.

2 АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

Для разработки модуля были проанализированы следующие аналоги:

1. Программа Typle [2];
2. Программа Speaker [3];
3. Программа Braina [4];
4. Программа Горыныч [2][3];
5. Программа Cortana [3][5].

2.1 Программа Typle

Typle дает возможность выполнять некоторые операции на компьютере, используя голосовые команды. Программа поддерживает операционные системы Windows XP, Windows 2003, Windows Vista.

Интерфейс приложения представлен на рисунке 2.1

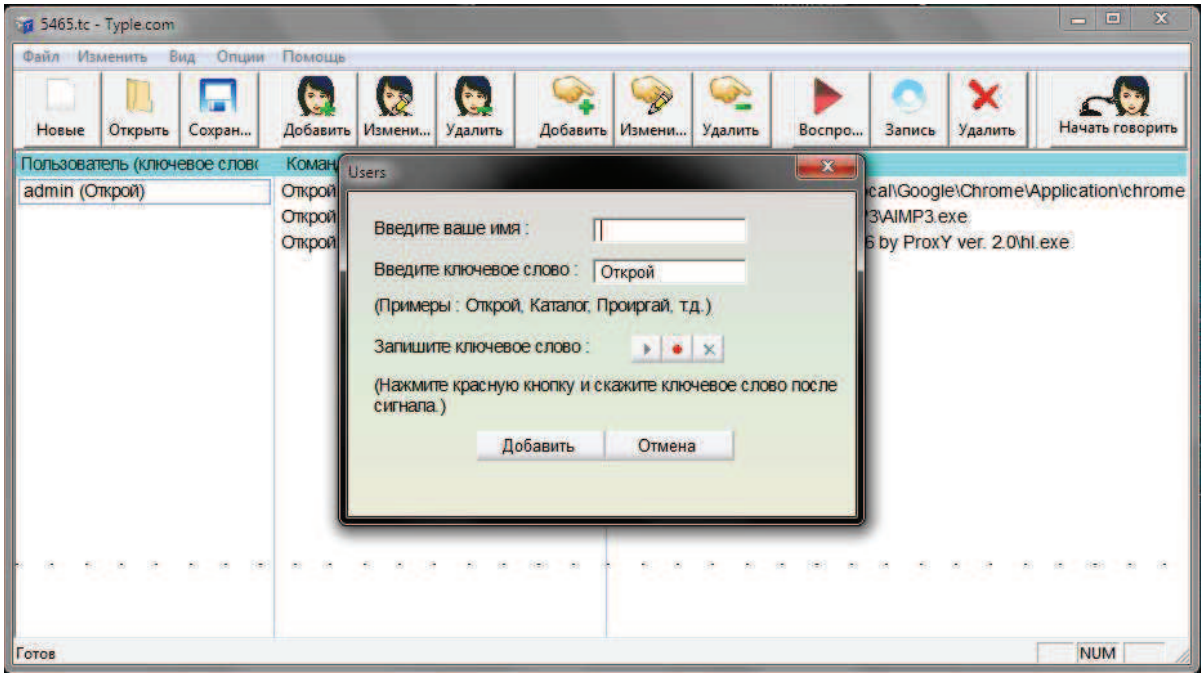

Рисунок 2.1 – Приложение Typle

Преимущества:

– не нужны знания английского языка;

– есть бесплатная версия;
– неограниченное количество команд.

Недостатки:

– сложный алгоритм создания команды;
– ограниченный функционал, можно лишь открывать утилиты и страницы в интернете;
– программа иногда воспринимает посторонние шумы как команды, из-за этого персональный компьютер может работать некорректно;
– программа работает корректно только с пользователем, который записал команду;
– неудобный интерфейс, так как на панели управления есть много различных функций, некоторые - с одинаковыми названиями. Надо ориентироваться по картинке, а не по надписи.
– программа работает с русским языком, но не всегда правильно его распознаёт. Говорить надо громко, чётко, механическим голосом[2].

2.2 Программа Speaker

Программа Speaker позволяет управлять компьютером с Windows 10 или 7 с помощью голосовых команд. Она имеет следующий функционал:

– создание снимков состояния экрана;
– переключение раскладки клавиатуры;
– завершение работы Windows 7;
– запуск приложения;
– открытие файла.

Преимущества:

– не нужны знания английского языка;
– есть бесплатная версия.

Недостатки:

– процесс считывания и распознавания поступающей на микрофон информации запускается после нажатия на заданную клавишу;
– на обработку, распознание речи и выполнение команды уходит достаточно много времени – более пяти секунд;
– ключевые слова задаются текстом, а не словами, потому распознанная речь сравнивается уже с введенным текстом, что делается далеко не идеально[3].

Интерфейс приложения предоставлен на рисунке 2.2.

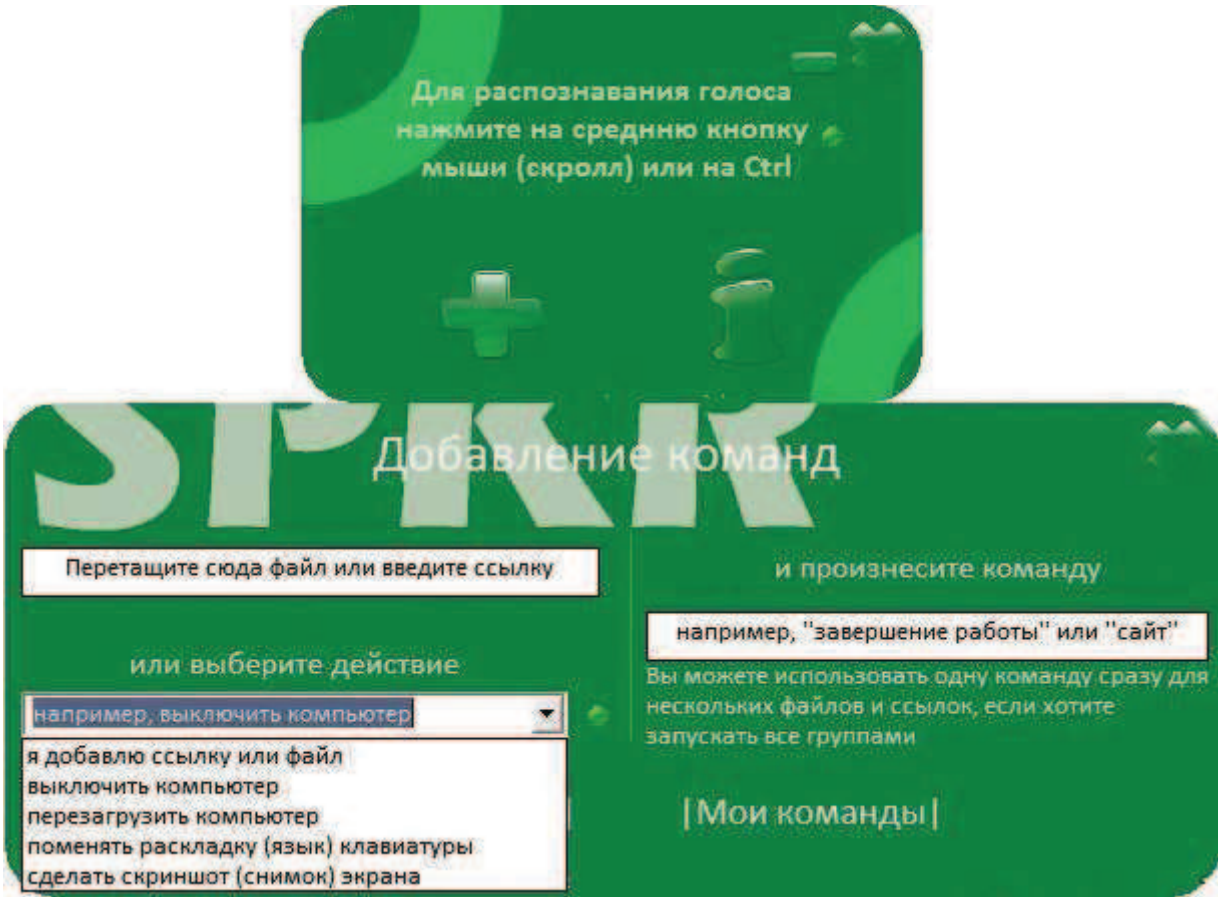

Рисунок 2.2 – Приложение Speaker

2.3 Программа Braina

Braina – программа распознавания голоса с возможностью преобразования голосовых сообщений в текст. Кроме функции диктовки, Braina обладает командами, позволяющими:

– делать запросы в поисковых системах;
– открывать файлы, программы, веб-сайты;
– находить информацию;
– устанавливать напоминания;
– делать заметки.

Преимущества:

– высокая скорость работы (скорость отклика от 0,9 до 1,5 секунд);
– неограниченное количество команд.

Недостатки:

– программа поддерживает только английский язык;
– программа распространяется на платной основе, 59 долларов в год;
– пробная версия на 7 дней имеет ограниченный функционал [4].

Интерфейс приложения предоставлен на рисунке 2.3.

Рисунок 2.3 – Приложение Braina.

2.4 Программа Горыныч.

Российский модуль системы «Горыныч» предназначен для совместной работы с американской программой Dragon Dictates. Данная программа позволяет диктовать на русском и английском языке для быстрого введения текстов в компьютер в любом редакторе под Windows.

Также при помощи программы можно полностью контролировать компьютер речевыми командами. Программа приспосабливается к голосу пользователя и со временем качество распознавания команд и текста становится выше, до очередной деинсталляции программы или переустановки операционной системы без сохранения пользовательских данных.

С системных функций программный комплекс легко выполняет запуск приложений, создание новых текстовых файлов[2].

Преимущества:

- Есть поддержка русского и английского языков.
- Распознавание текста, голосовой ввод в любые редакторы.

Недостатки:

- Необходимо самостоятельно создавать команды для каждого процесса. В буквальном смысле придётся записывать словарь.
- Разработчик прекратил поддержку программы[3].

Интерфейс приложения предоставлен на рисунке 2.4.

Рисунок 2.4 – Приложение Горыныч.

2.5 Программа Cortana.

Cortana – голосовой помощник, разработанный корпорацией Microsoft для Windows 10, Phone и Android с дальнейшим распространением проекта на XBox и iOS. Она заменяет классическую поисковую строку и выполняет множество действий, в первую очередь, связанных с поиском информации и системных команд, получая их от пользователя в виде голосовых команд.

Преимущества:

- может находить информацию как в поисковых системах, так и среди личных файлов пользователя;
- имеет гибкие настройки интерфейса;
- может поддержать беседу.

Недостатки:

- глубокая интеграция в Windows 10;
- отсутствие русского языка;
- сбор фактически всей информации о пользователе с отправкой на серверы Microsoft;
- отсутствие финальной версии [3][5].

Интерфейс приложения предоставлен на рисунке 2.5.

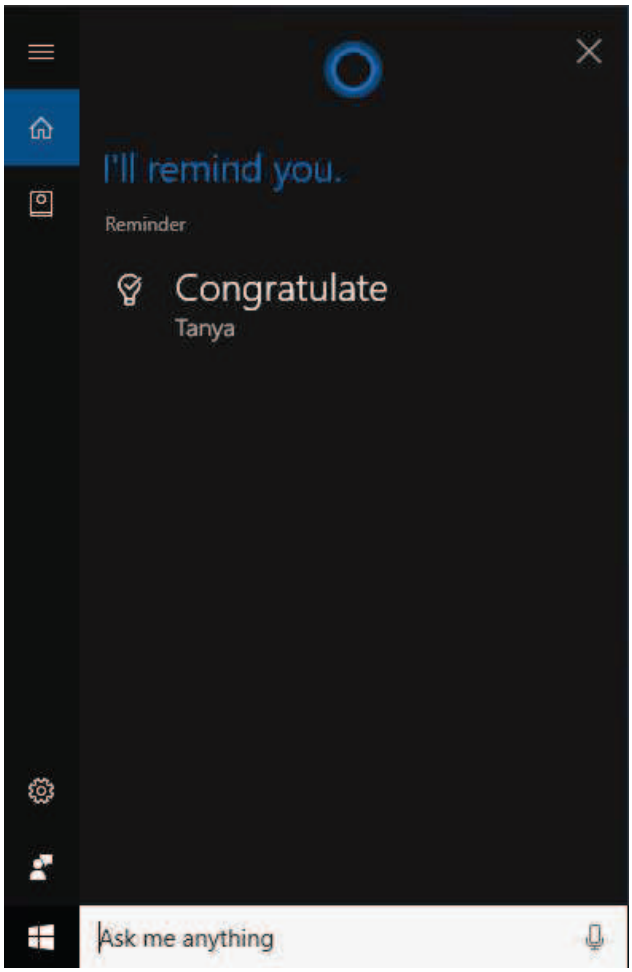

Рисунок 2.5 – Приложение Cortana.

В ходе проведенного анализа аналогов были выявлены следующие критерии, которыми должна обладать программа удаленного управления с помощью голосовых команд, такие как:

- скорость работы программы;
- отсутствие ограничений на количество команд;

- поддержка разработчиком;
- отсутствие зависимости результата от точной команды;
- удобный и современный пользовательский интерфейс;
- полностью бесплатное использование;
- поддержка русского языка.

Сравнительный анализ программ представлен в таблице 2.1.

Критерий/Название	Typle	Speaker	Braina	Горыныч	Cortana
Скорость, с.	1,5 – 2,5	2,5 - 4	0,8 – 1,5	2,5 - 4	0,8 – 1,5
Отсутствие ограничений	-	+	+	+	+
Поддержка	-	-	+	-	+
Отсутствие зависимости результата	-	-	-	-	-
Удобный интерфейс	-	-	+	-	+
Бесплатность	+	+	-	+	+
Поддержка русского языка	+	+	-	+	-

Таблица 2.1 – Сравнение программных продуктов по критериям

3 ТЕХНИЧЕСКОЕ ЗАДАНИЕ МОДУЛЯ

Необходимо разработать программу, позволяющую осуществить управление персональным компьютером с помощью голосовых команд, а также иметь возможность диктовки текста в любую активную форму приложения. Программа должна работать под ОС Windows.

Программа должна иметь следующий функционал:

1. Добавление информации о голосовой команде, а именно:
   1.1 Ключевое слово или словосочетание в текстовой форме;
   1.2 Выбранное действие, которое нужно произвести при произнесении ключевого слова или словосочетания;
   1.3 Выбрать программу, с которой будут производиться действия.
2. Редактирование информации о голосовой команде;
3. Удаление информации о голосовой команде;
4. Возможность выполнять сочетания клавиш, в этом случае, программа должна обеспечить удобный ввод сочетания клавиш путем нажатия на соответствующие клавиши;
5. Возможность просмотра словаря добавленных команд;
6. Возможность просмотра произнесенной фразы;
7. Функцию диктовки текста, которая активируется и деактивируется по заданной команде;
8. Информирование пользователя о таких ошибках, как:
   8.1 Отсутствие интернет соединения;
   8.2 Отсутствие подключенного микрофона;
   8.3 Некорректно заданный путь к исполняемому файлу.

4 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

При разработке модуля голосового управления можно выделить следующие этапы:

- построение диаграммы использования;
- построение объектной модели;
- выбор инструментов для разработки модуля голосового управления;
- разработка модуля голосового управления.

4.1 Построение диаграммы использования

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе.

Диаграммы вариантов использования предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы. Диаграммы вариантов использования говорят о том, что система должна делать, не указывая сами применяемые методы[6].

В результате анализа технического задания была построенная следующая диаграмма использования.

Построенная диаграмма использования представлена на рисунке 4.1.

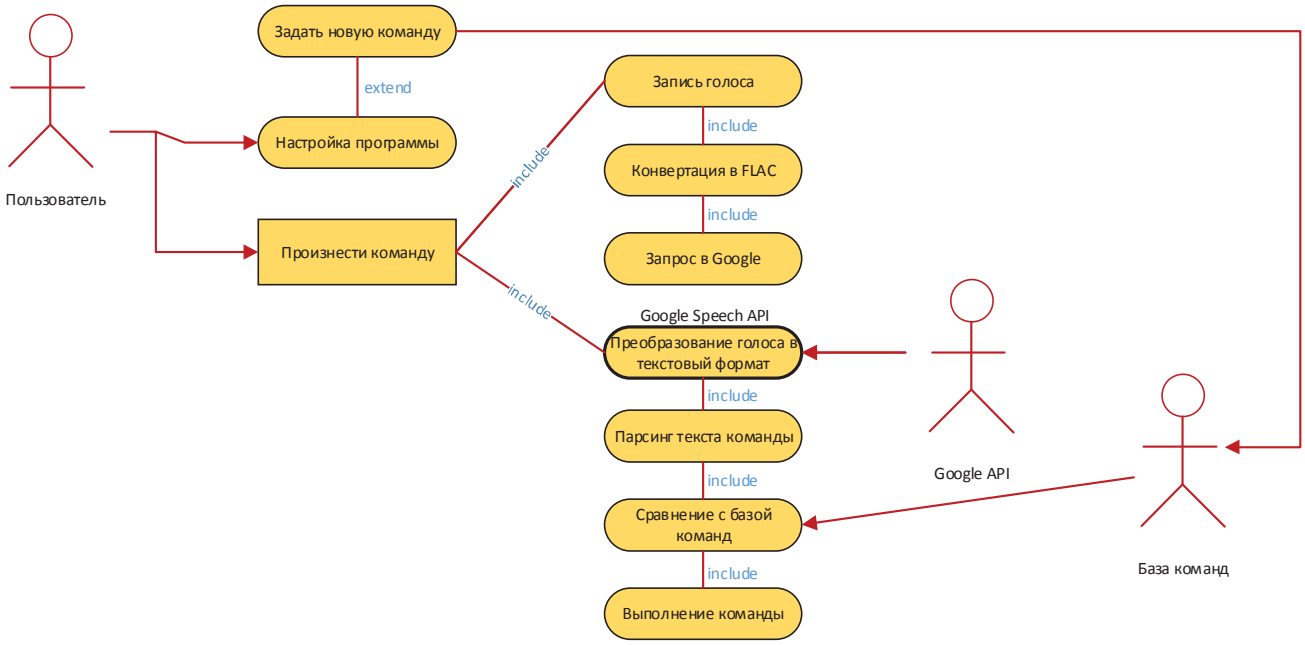

Рисунок 4.1 – Диаграмма использования.

На основе диаграммы использования выделим бизнес-сущности и бизнес-логику.

Бизнес-сущности: Пользователь, Голосовая команда, Аудиофайл.

Бизнес-логика: Класс «Исполнитель», Интерфейсы: Google Speech API, Аудио конвертер, База команд.

Голосовая команда – это директива для запуска голосового управления. В аудиофайле хранится запись с командой пользователя.

Аудио конвертер предназначен для конвертирования аудиофайла в необходимый нам формат. Далее, аудиофайл передаётся в Google Speech API и принимается ответ, который сравнивается с базой команд.

4.2 Построение объектной модели

Объектная модель описывает структуру объектов, составляющих систему, их атрибуты, операции, взаимосвязи с другими объектами. В объектной модели должны быть отражены те понятия и объекты реального мира, которые

важны для разрабатываемой системы. В объектной модели отражается прежде всего прагматика разрабатываемой системы, что выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы[7].

В построенной объектной модели наглядно продемонстрированы объекты, составляющие программу, такие как:

- пользователь;
- интерфейс: аудио конвертор;
- аудио файл;
- интерфейс: Google Speech API;
- база команд.

Так же, показаны связи между объектами. Структура каждого объекта, показанного на модели, отображает выполняемые операции и установленные атрибуты. Объектная модель представлена на рисунке 4.2.

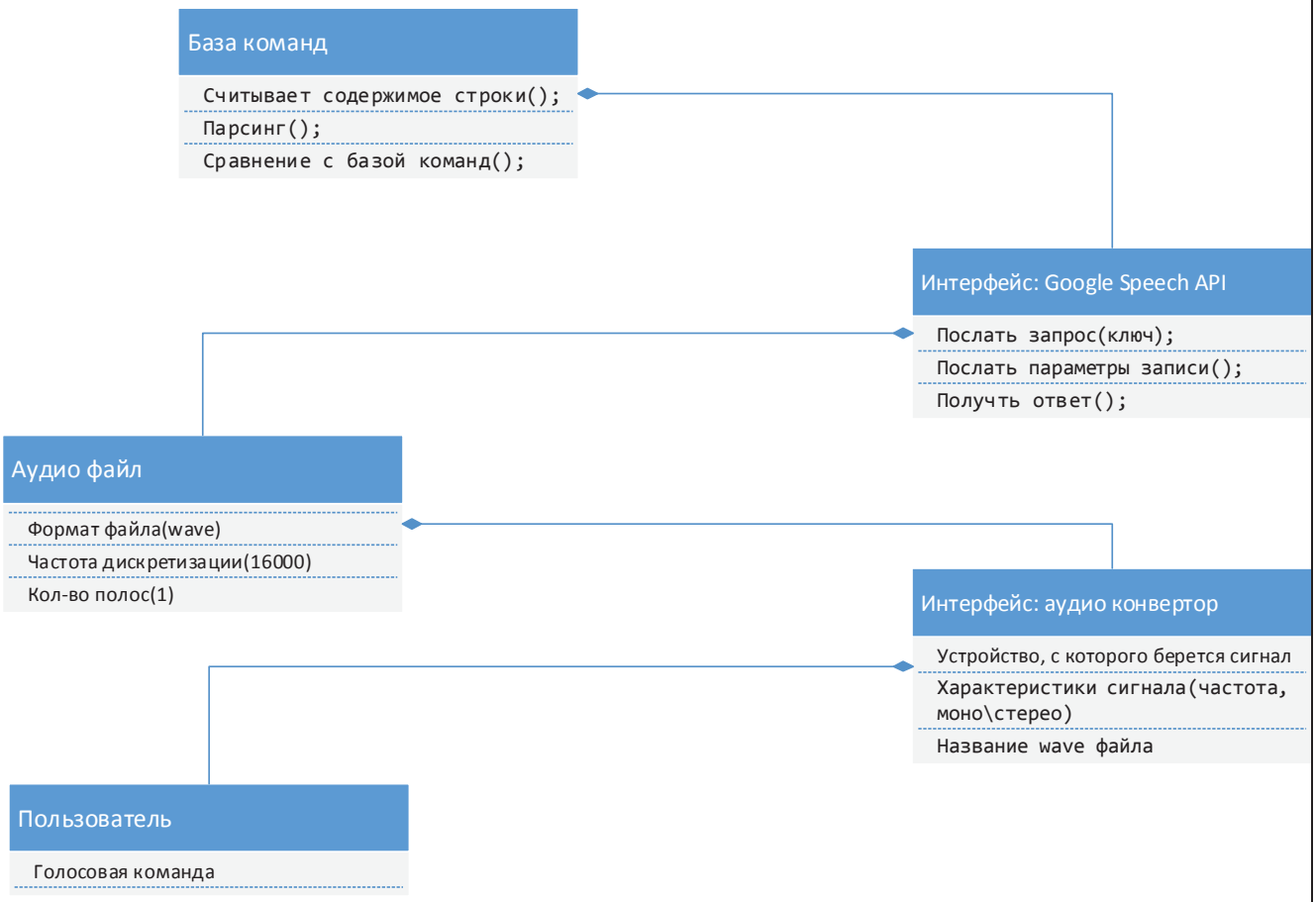

Рисунок 4.2 – Объектная модель

4.3 Выбор инструментов для разработки модуля для голосового управления

В качестве языка программирования был выбран C#. В настоящее время в сфере разработки программного обеспечения преобладают языки программирования Java и C#. Рейтинг языков программирования представлен на рисунке 4.3 [18]. Оба языка имеют общего предка – язык C (C++) и схожий синтаксис, но язык C# корпорации Microsoft появился примерно на 10 лет позже Java, и при его разработке были учтены и исправлены многие недостатки Java[8].

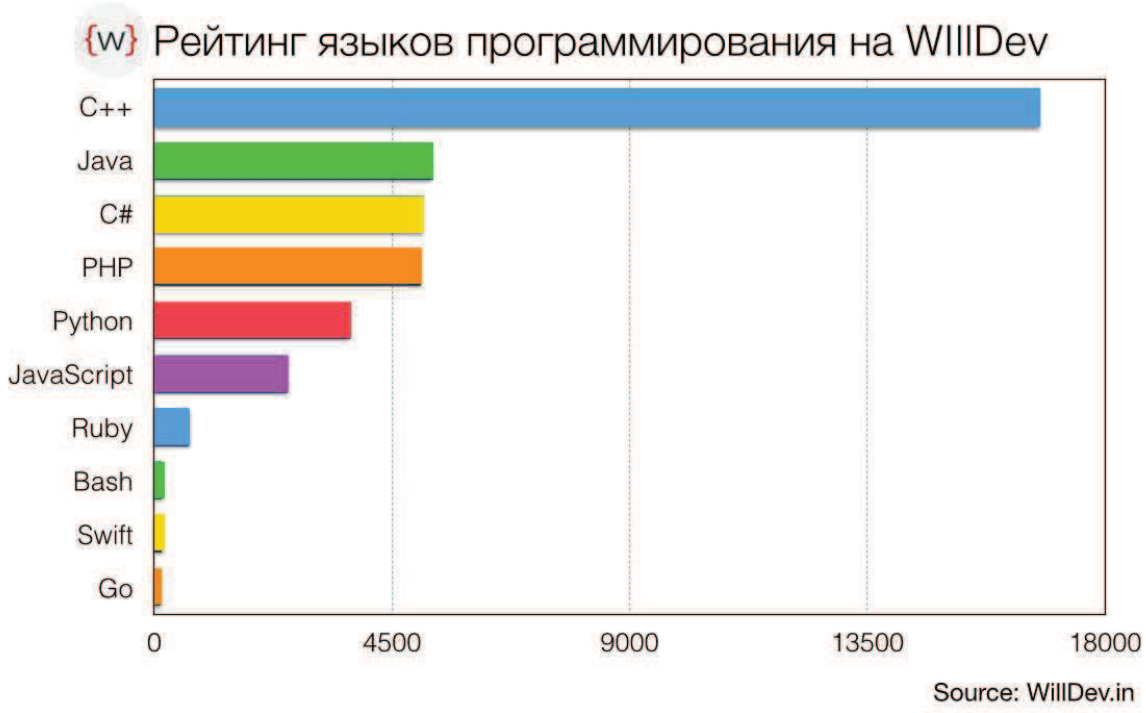

Рисунок 4.3 – Рейтинг языков программирования по рассчитанный по количеству сабмитов в системе контроля версий gihub

Язык C# является основным языком программирования платформы .NET. Использование C# позволяет разрабатывать[9]:

– любые приложения для настольных компьютеров (Windows Forms);

– веб-приложения любой сложности для и интернет (ASP.Net MVC, MVVM);

– серверные технологии доступа к данными (ADO.NET, LINQ, Entity Framework);

– распределённые системы на основе веб-сервисов;

– приложения для планшетов (Windows 10);

– приложения для смартфонов (Windows Phone);

– кроссплатформенные приложения для операционных систем iOS, Android (VS 2015, Xamarin);

– игры для Windows, X-Box (Unity3D);

– приложения облачных технологий (Windows Azure);

– приложения для встраиваемых систем (интернет вещей).

Корпорация Microsoft активно разрабатывает новые технологии и обеспечивает их поддержку платформой .NET.

Выделим ряд преимуществ языка C#:

- расширяемость системы (в C# возможно подгружать любые исполняемые файлы, импортировать классы и объекты из других программ);
- кроссплатформенность (mono, концепция NET);
- сложность разработки и сопровождения (подбор кадров, читаемость кода, документированность языка);
- степень открытости исходных текстов библиотек, исполняемых программ, количество литературы и помощь (MSDN);
- защищенность и контроль версий подключаемых алгоритмов (концепция NET);
- трудоемкость написания (концепция NET);
- скорость работы (распределение процессов, распределение данных скорость работы с данными).

Исходя из вышеперечисленного, можно сделать вывод, что посредством языка C# можно реализовать заявленный функционал разрабатываемого модуля голосового управления.

В качестве среды разработки используем Visual Studio 2012. Данная IDE обладает огромным набором средств и возможностей: позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения и веб-службы для всех поддерживаемых платформ: Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone, .NET Compact Framework и Silverlight[10].

В использовании Visual Studio есть и недостатки, например, невозможность работать на платформах, отличных от Windows, однако ~~наш~~ проект изначально разрабатывается для платформы Windows. Visual Studio распространяется на платной основе, но Microsoft предоставляет программу бесплатного распространения для студентов.

В качестве системы управления версиями файлов с исходным кодом используется Git.

Преимущества и недостатки git по сравнению с централизованными системами управления версиями (такими как, например, Subversion) типичны для любой распределённой системы. Если же сравнивать git с «родственными» ей распределёнными системами, можно отметить, что git изначально идеологически ориентирован на работу с изменениями, а не с файлами, «единицей обработки» для него является набор изменений, или патч. Эта особенность прослеживается как в структуре самой системы (в частности — в структуре репозитория), так и в принципах построения команд; она отражается на производительности системы в различных вариантах её использования и на достоинствах и недостатках git по сравнению с другими DVCS[11].

Для хранения исходного кода используется веб-сервис BitBucket – веб-сервис для хостинга проектов и их совместной разработки, основанный на системе контроля версий Git. Главным преимуществом данного сервиса является предоставление бесплатных закрытых репозиториев[12].

Для управления задачами используется приложение Redmine. Данный продукт предоставляет следующие возможности [13]:

- ведение нескольких проектов;
- гибкая система доступа, основанная на ролях;
- система отслеживания ошибок;
- диаграммы Ганта и календарь;
- ведение новостей проекта, документов и управление файлами;
- оповещение об изменениях с помощью RSS-потоков и электронной почты;
- вехи для каждого проекта;
- форумы для каждого проекта;
- учёт временных затрат;
- настраиваемые произвольные поля для инцидентов, временных затрат, проектов и пользователей;
- лёгкая интеграция с системами управления версиями (SVN, CVS, Git, Mercurial, Bazaar и Darcs);
- создание записей об ошибках на основе полученных писем;
- поддержка множественной аутентификации LDAP;
- возможность самостоятельной регистрации новых пользователей;
- многоязычный интерфейс (в том числе русский).

Для решения задач записи и обработки звука была выбрана библиотека NAudio. NAudio - это библиотека с открытым исходным кодом .NET, предназначенная для работы с аудио и MIDI, содержащая десятки полезных аудио-классов, предназначенных для ускорения разработки утилит, связанных с аудио в .NET. NAudio находится в разработке с 2002 года и её возможности выросли до широкого спектра функций:

1. Воспроизведение аудио с использованием различных API:
   1.1 WaveOut;
   1.2 DirectSound;
   1.3 ASIO;
   1.4 WASAPI.

2. Чтение аудио из многих стандартных форматов файлов:
   2.1 WAV;
   2.2 AIFF;
   2.3 WMA;
   2.4 Файлы SoundFont (SF2);
3. Декодирование многих популярных типов сжатия аудио:
   3.1 MP3 (с использованием ACM, DMO или MFT);
   3.2 G.711 mu-law и a-law;
   3.3 ADPCM;
   3.4 G.722;
   3.5 Speex (с использованием NSpeex);
   3.6 WMA, AAC, MP4 и другие.
4. Преобразование различных форм несжатого аудио:
   4.1 Изменение количества каналов;
   4.2 Изменение битовой глубины.
5. Кодирование звука с использованием любого кодека ACM или Media Foundation, установленного компьютере:
   5.1 Создание MP3-файлов в Windows 8 и выше;
   5.2 Создание AAC / MP4 в Windows 7 и выше;
   5.3 Создание файлов WMA;
   5.4 Создание WAV-файлов, содержащих G.711, ADPCM, G.722 и т.д.
6. Смешивание и управление аудиопотоками с помощью 32-битного микширования:
   6.1 Анализ уровней выборки для измерения или визуализации осциллограмм;
   6.2 Задержка, loop и затухание звука через вход или выход;
   6.3 Изменение тона звука с помощью фазового вокодера.
7. Запись аудио с использованием WaveIn, WASAPI или ASIO;
8. Работа со звуковыми картами.

NAudio был создан так как библиотека классов платформы, которая поставляется с .NET 1.0 не имеет поддержки для воспроизведения аудио. Пространство имен System.Media, представленное в .NET 2.0 предоставляло не много поддержки, и MediaElement в WPF и Silverlight было не многим лучше. Цель NAudio предоставить комплексный набор классов связанных с аудио, позволяющих легко разрабатывать утилиты, которые воспроизводят или записывают аудио, или манипулируют аудио файлами каким-то способом.

NAudio находится под лицензией Microsoft Public License (Ms-PL), которая означает, что возможно использование в любом проекте.

NAudio достаточно производителен, чтобы обрабатывать множественные WAV файлы вместе, включая их обработку различными эффектами и кодеками, воспроизводить безошибочно с задержкой около 50мс на достаточно скромных ПК [14].

Для разработки графического интерфейса были выбраны Windows Forms. Windows Forms позволяет разрабатывать интеллектуальные клиенты. Интеллектуальный клиент – это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows.

Windows Forms - это технология интеллектуальных клиентов для .NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью среды разработки типа Visual Studio можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Если существующий элемент

управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl [15].

Для решения проблемы хранения структурированных данных был использован формат JSON. JSON - простой, основанный на использовании текста, способ хранить и передавать структурированные данные. С помощью простого синтаксиса возможно хранить данные любых типов, начиная от одного числа до строк, массивов и объектов, в простом тексте. Также можно связывать между собой массивы и объекты, создавая сложные структуры данных.

После создания строки JSON, отправка данных другому приложению или в другое место сети не предоставляет трудности, так как она представляет собой простой текст[16].

JSON имеет следующие преимущества:

- компактность;
- предложения, сгенерированные json, читаются и составляются как человеком, так и компьютером;
- простота преобразования в структуру данных для большинства языков программирования (числа, строки, логические переменные, массивы и так далее);
- многие языки программирования имеют функции и библиотеки для чтения и создания структур JSON.

Google Cloud Speech API позволяет разработчикам конвертировать аудио в текст, применяя мощные нейронные сети в удобном API. API распознает более 80 языков и диалектов, чтобы поддерживать глобальную пользовательскую базу. Существует функция создания транскрипции текста пользователей, диктующих через микрофон, включения команд управления голосом или создание транскрипций звуковых файлов. С помощью API возможно распознать звук, загрузить в запрос и интегрировать его с хранилищем аудио в Google Cloud Storage, используя ту же технологию, которую Google использует для собственных продуктов.

В Speech API применяются самые передовые алгоритмы машинного обучения и нейронных сетей для распознавания речи. Точность распознавания речи улучшается с течением времени.

Speech API может передавать текстовые результаты, возвращая результаты частичного распознавания по мере их появления. Кроме того, Speech API может возвращать распознанный текст из аудиофайла, хранящегося в файле.

Перед отправкой аудио в Speech API не потребуется предварительная обработка сигнала или подавление шумов. Служба может успешно обрабатывать шумные звуки в различных средах.

Распознавание речи можно адаптироваться к контексту, предоставляя отдельный набор словарных подсказок при каждом вызове API. Особенно полезен для использования в приложениях управления приложениями[17].

Алгоритм преобразования речи в текст с помощью Google Speech API представлен на рисунке 4.4

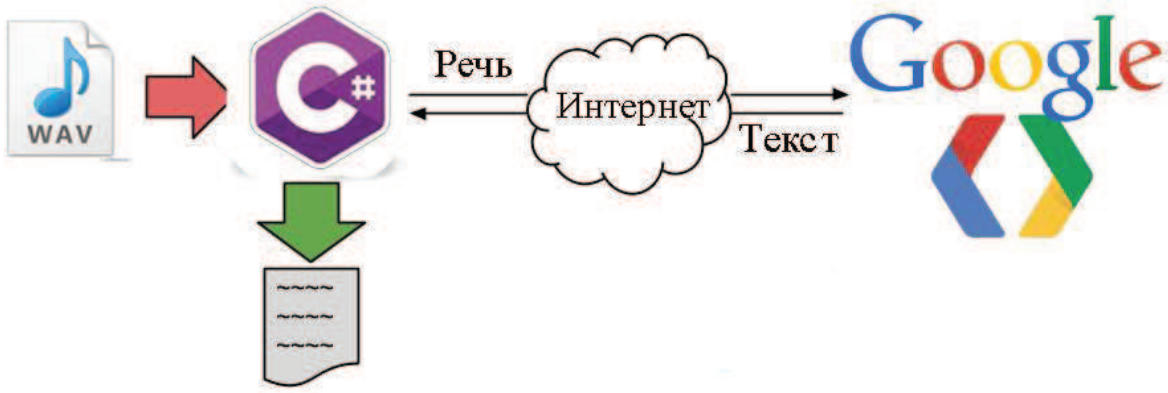

Рисунок 4.4 – Преобразование речи в текст с помощью Google Speech API

4.4. Разработка модуля голосового управления

Разработка приложения начинается с реализации функции записи звука с микрофона с применением библиотеки NAudio. NAudio.dll дает более чистый звук, без помех, в отличии от Winmm.dll.

Для начала, инициализируем поток для записи WaveIn и класс для записи в файл WaveFileWriter. Фрагмент кода с инициализацией показан в листинге 4.1.

Листинг 4.1. – Фрагмент кода с инициализацией данных для библиотеки NAudio.

```
public partial class Form1 : Form
    {
        // WaveIn - поток для записи
        WaveIn waveIn;
        //Класс для записи в файл
        WaveFileWriter writer;
        //Имя файла для записи
        string outputFilename = "имя_файла.wav";

        public Form1()
        {
            InitializeComponent();
        }
```

Далее, реализуем начало и остановку записи по кнопке. Начинаем запись с обработчика нажатия кнопки. Затем определяем стандартное устройство для записи (если оно имеется). В операционной системе Windows, стандартным устройством для записи является микрофон под номером 0. Прикрепляем к событию DataAvailable обработчик, возникающий при наличии записываемых данных. Прикрепляем обработчик завершения записи. Далее задаем формат аудио, в котором будет записываться звук. Используется формат Wav, так как именно этот формат потребуется для дальнейшей обработки. Так же, необходимо задать дополнительные параметры, такие как частота дискретизации и количество каналов. Для задачи достаточно 8000Гц и одноканальной записи звука. Затем инициализируем объект WaveFileWriter и начинаем запись. Фрагмент кода с началом записи показан в листинге 4.2.

Листинг 4.2. – Фрагмент кода с началом записи.

```
 //Начинаем запись - обработчик нажатия кнопки
        private void button1_Click(object sender, EventArgs e)
        {
        try
        {
            MessageBox.Show("Start Recording");
            waveIn = new WaveIn();
```

```
                //Дефолтное устройство для записи (если оно имеется)
                //встроенный микрофон ноутбука имеет номер 0
                waveIn.DeviceNumber = 0;
               //Прикрепляем к событию DataAvailable обработчик, возникающий при
наличии записываемых данных
                waveIn.DataAvailable += waveIn_DataAvailable;
                //Прикрепляем обработчик завершения записи
                waveIn.RecordingStopped += new EventHandler(waveIn_RecordingStopped);
                //Формат wav-файла - принимает параметры - частоту дискретизации и
количество каналов(здесь mono)
                waveIn.WaveFormat = new WaveFormat(8000, 1);
                //Инициализируем объект WaveFileWriter
                writer = new WaveFileWriter(outputFilename, waveIn.WaveFormat);
                //Начало записи
                waveIn.StartRecording();
                }
                catch (Exception ex)
                {MessageBox.Show(ex.Message)}
            }
```

Таким образом, была реализована запись голоса используя кнопку. Так же, исходя из технического задания, необходимо реализовать инициирование записи голоса без нажатия на кнопки. Во время реализации этой функции возникла следующая проблема: при записи звука иногда возникает необходимость записи только человеческого голоса, не записывая при этом какие-то посторонние звуки, шумы и т.д. Для этого используются так называемые VAD-фильтры (Voice Activity Detection), которые позволяют выявлять, если ли голос на некотором звуковом отрезке. Используя NAudio API создадим детектор голоса.

Итак, функция ProcessData возвращает булево логическое значение, показывающее определена или нет речь на некотором отрезке звука. Чтобы получить звук, передаем аргумент типа WaveInEventArgs. Этот тип принадлежит библиотеки NAudio. Он нужен, потому что содержит буфер звука, который в последствии будет использован: e.Buffer. То есть фактически, у ~~нас~~ появилась возможность, в качестве параметра, передать тот массив звуковых данных, который был нами получен с микрофона.

Следует отметить, что используется некоторый порог, равный определенному значению. Если уровень громкости звука станет больше

порогового значения, установится триггер, который начнет работу алгоритма записи звука. В данном случае порог равен 0.02 (private double porog = 0.02;) и может варьироваться и устанавливаться экспериментальным путем в зависимости от ситуации.

Получая значение типа short, а затем double сравниваем полученное значение с порогом и на основе сравнения узнаем, есть ли звук на данном речевом отрезке или нет.

Работа триггера по пороговому значению представлена на рисунке 4.5.

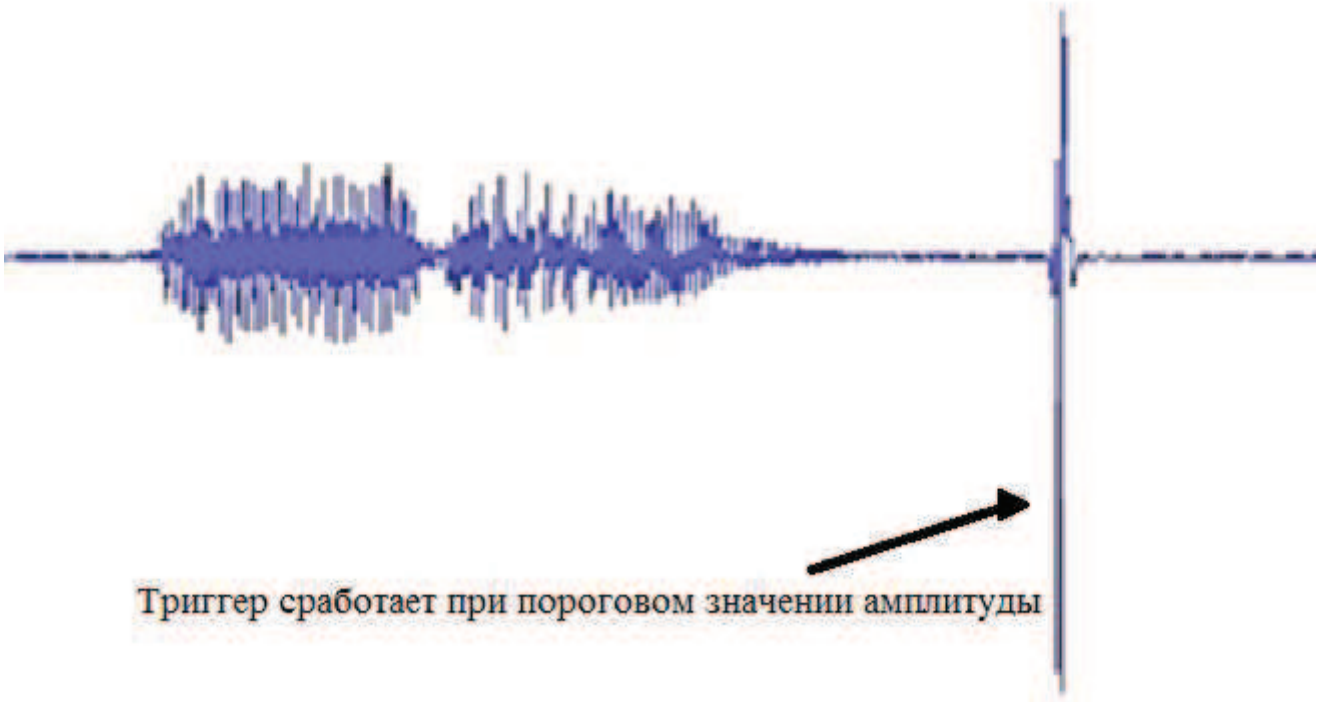

Рисунок 4.5. - Работа триггера по пороговому значению амплитуды.

Исходный код нашего фильтра голоса представлен в листинге 4.3.

Листинг 4.3. – фильтра голоса.

```
// Обработчка речи - вычисляем, есть ли сама речь на звуковом отрезке
        private bool ProcessData(WaveInEventArgs e)
        {
            // Порог для вычисления наличия речи
            private double porog = 0.02;
            bool result = false;
            bool Tr = false;
```

```
                double Sum2 = 0;
                int Count = e.BytesRecorded / 2;
                for (int index = 0; index < e.BytesRecorded; index += 2)
                {
                    double Tmp = (short)((e.Buffer[index + 1] << 8) | e.Buffer[index +
0]);
                    Tmp /= 32768.0;
                    Sum2 += Tmp * Tmp;
                    if (Tmp > porog)
                        Tr = true;
                }
                Sum2 /= Count;
                if (Tr || Sum2 > porog)
                { result = true; }
                else
                { result = false; }
                return result;
            }
```

В качестве механизма преобразования речи в текстовый формат был использован Google Speech API. Для того, чтобы иметь возможность использовать сервис, необходимо зарегистрироваться как разработчик. Далее был получен персональный ключ для аутентификации в формате JSON. Этот ключ используется для формирования веб-запроса, который отправляется на сервер сервиса.

Фрагмент кода, отвечающий за отправку запроса показан в листинге 4.4.

```
    private void Responsing(out HttpWebResponse response, out StreamReader reader)
            {
                WebRequest request = WebRequest.Create("https://www.google.com/speech-
api/v2/recognize?output=json&lang=ru-RU&key=AIzaSyBaJbO7tnfrt83wxzmZrSVL9F_MEqWZFac");
                request.Method = "POST";
                byte[] byteArray = File.ReadAllBytes(outputFilename);
                request.ContentType = "audio/l16; rate=8000"; //"8000";
                request.ContentLength = byteArray.Length;
```

Листинг 4.4. - Отправка запроса.

В поле WebRequest request задается путь отправки запроса по персональному ключу.

В поле request.ContentType, задается режим кодирования, в рамках проекта используется «audio/l16», что означает схему кодирования LINEAR16, которая обеспечивает кодирование без потерь для лучшей производительности. Так

же, в данном поле обязательно задается частота дискретизации. В рамках проекта частота дискретизации равна 8000Гц, что обеспечивает наилучшую производительность и скорость работы за счёт ухудшения качества записи.

Далее, следует получить ответ на запрос от сервиса. Google Speech API возвращает нам результат в таком виде:

"{\"result\":[]}\n{\"result\":[{\"alternative\":[{\"transcript\":\"Я говорю для теста\",\"confidence\":0.98718762}],\"final\":true}],\"result_index\":0}\n".

Это текст в формате JSON, который следует разобрать для получения произнесенной фразы. Для этого удаляем первые 56 символа, а так же удаляем все символы после первого «\». В результате получаем произнесенную фразу.

Так же, необходимо перевести буквы в нижний регистр, это следует сделать по той причине, что корректная работа функции, отвечающей за сравнение фразы со словарем команд, гарантируется только при том условии, что исходный текст будет находиться в нижнем регистре.

Фрагмент кода, отвечающий за редактирование результата работы сервиса Google Speech API представлен в листинге 4.5.

```
string source = label1.Text;
     string kek = source.Remove(0, 56);
     int first = kek.IndexOf('"');
     string profit = kek.Remove(first).ToLower();
```

Листинг 4.5. - Отправка запроса.

Согласно техническому заданию, приложение должно обеспечить возможность выполнять сочетания клавиш, в таком случае, программа должна обеспечить удобный ввод сочетания клавиш путем нажатия на соответствующие клавиши. Для этого используем оператор SendKeys, который посылает сообщения о нажатии клавиш активному приложению. Так как идет работа с оператором SendKeys, необходимо реализовать функцию по замене кодировки горячих клавиш. ReplaceHotKeys заменяет Ctrl, Alt, Shift на «^», «%», «+» соответственно.

Фрагмент кода, отвечающий за замену кодировки управляющих клавиш представлен в листинге 4.6.

```
       private string ReplaceHotKeys(string hotkey)
            {
                hotkey = hotkey.Replace(" + ", "").Replace("Shift",
"+").Replace("Shift", "+").Replace("Ctrl", "^").Replace("Alt", "%");
                if (char.IsUpper(hotkey[0]))
                    hotkey = hotkey.Remove(0, 1);
                int first = hotkey.Length-1;
                hotkey = hotkey.Insert(first, "{");
                int last = hotkey.Length;
                hotkey = hotkey.Insert(last, "}");
                hotkey = hotkey.ToLower();

                return hotkey;
            }
```

Листинг 4.6. - Замена кодировки управляющих клавиш.

Был реализован класс globalKeyboardHook. Данный класс нужен для того, чтобы отлавливать выбранную клавишу для начала или конца записи. Это позволяет управлять программой с помощью клавиши даже когда программа свернута или находится в области уведомлений.

Фрагмент кода, реализующий класс globalKeyboardHook, представлен в листинге 4.7.

```
            public int hookProc(int code, int wParam, ref keyboardHookStruct lParam)
            {
                if (code >= 0)
                {
                    Keys key = (Keys)lParam.vkCode;
                    if (HookedKeys.Contains(key))
                    {
                        KeyEventArgs kea = new KeyEventArgs(key);
                        if ((wParam == WM_KEYDOWN || wParam == WM_SYSKEYDOWN) &&
(KeyDown != null))
                        {
                            KeyDown(this, kea);
                        }
                        else if ((wParam == WM_KEYUP || wParam == WM_SYSKEYUP) &&
(KeyUp != null))
                        {
                            KeyUp(this, kea);
                        }
                        if (kea.Handled)
                            return 1;
                    }
                }
                return CallNextHookEx(hhook, code, wParam, ref lParam);
            }
```

Листинг 4.7. - Класс globalKeyboardHook.

Команды, добавленные в программу сериализуются в json файл. Это нужно для того, чтобы после перезапуска программы все команды сохранялись и подгружались обратно в программу с помощью написанного десериализатора.

Фрагмент кода, реализующий сериализацию и десериализацию данных, представлен в листинге 4.8.

```
    public void serialJSON(string fName)
                {
                    DataContractJsonSerializer serializer = new
DataContractJsonSerializer(this.GetType());
                    FileStream stream = new FileStream(fName, FileMode.Create);
                    serializer.WriteObject(stream, this);
                    stream.Close();
                }
    public static Dictionary deserialJSON(string fName)
                {
                    Dictionary dictionary = new Dictionary();
                    DataContractJsonSerializer serializer = new
DataContractJsonSerializer(dictionary.GetType());
                    FileStream stream = new FileStream(fName, FileMode.Open);
                    dictionary = (Dictionary)serializer.ReadObject(stream);
                    stream.Close();
                    return dictionary;
                }
```

Листинг 4.8. - Сериализация и десериализация данных.

При выполнении программы выполняется, все данные о командах хранятся в контейнере List. Каждой команде присвоен отдельный параметр, который отвечает за вид команды. Параметры делятся на три вида:

- открыть, имеет параметр «-o»;
- закрыть, имеет параметр «-c»;
- выполнить сочетание клавиш, имеет параметр «-p».

Это позволяет увеличить скорость работы программы, так как сокращается количество проходов программы по словарю для нахождения совпадения.

У каждой команды могут быть несколько вариантов произнесения фразы, на которую она реагирует, например, «открыть проигрыватель» и «открыть плеер» равнозначны и выполняют одно и то же действие. Чтобы сравнивать произнесенную фразу с каждом вариантом, нужно разбить строку на несколько подстрок в зависимости сколько есть вариантов произнесения команды. Так как команды для произнесения заполняются через запятую, то

нужно разбить эти команды на массив строк. Фрагмент кода, реализующий разделение команды на массив строк, представлен в листинге 4.9.

```
    public string[] devideComands(string comands)
                {
                    comands = comands.ToLower();
                    String[] devidecomands = comands.Split(',');
                    return devidecomands;
                }
```

Листинг 4.9. - Разделение команды на массив строк.

После того как команда была разделена, фраза хранится в массиве строке. Далее, следует сравнить команду со словарем команд. Параметры присвоенные отдельным видам команд позволяют быстрее прогонять алгоритм по всем командам, отбрасывая ненужные. Например, если сказать «открыть блокнот», то программа знает, что во фразе присутствует «открыть» и следует пройти только по командам из соответствующего списка. Аналогично с остальными видами команд.

Фрагмент кода, реализующий проверку на совпадения со словарем, представлен в листинге 4.10.

```
    public void action(string profit)
                {

                    for (int i = 0; i < commands.Count; i++)
                    {
                        string[] devidecomands =
devideComands(commands[i].VoiceComand);

                        for (int p = 0; p < devidecomands.Length; p++)
                        {
                            if (profit.Contains(devidecomands[p]) && (commands[i].Param
== 'c') && (profit.Contains("закрыть") || profit.Contains("закрой")))
                            {
                                if (profit.Contains(devidecomands[p]))
                                {
                                    try
                                    {
                                        Process[] ps1 =
Process.GetProcessesByName(commands[i].Action);
                                        foreach (Process p1 in ps1)
                                        {
                                            p1.Kill();
                                        }
                                    }
                                    catch
                                    {
```

```
                                }
                            }
                        }

                        if (profit.Contains(devidecomands[p]) && (commands[i].Param
== 'o'))
                        {
                            Process.Start(commands[i].Action);
                        }

                        if (profit.Contains(devidecomands[p]) && (commands[i].Param
== 'p'))
                        {
                            SendKeys.Send(this.commands[i].Action);
                        }
                    }
                }
            }
```

Листинг 4.10. – Проверка на совпадения со словарем команд.

Для реализации диктовки текста требуется старт-слово и стоп-слово, которое задается пользователем в настройках. При произнесении старт-слова, включается режим диктовки. Диктовка текста реализована с помощью копирования в буфер обмена произнесенной фразы. Далее она вставляется в активную форму. Чтобы выйти из режима диктовки, требуется сказать стоп-слово.

Фрагмент кода, реализующий вставку распознанного текста в активную форму, представлен в листинге 4.10.

```
public void inputText(string profit)
            {
                Clipboard.SetText(profit+" ");
                SendKeys.SendWait("^v");
            }
```

Листинг 4.11. – Вставка распознанного текста в активную форму.

Интерфейс приложения выполнен в минималистичном стиле. Он максимально удобен и интуитивно понятен. При запуске приложение выводится форма с двумя крупными кнопками – одна начинает и останавливает запись, другая отвечает за настройки параметров программы.

Открытое окно с программой представлено на рисунке 4.6.

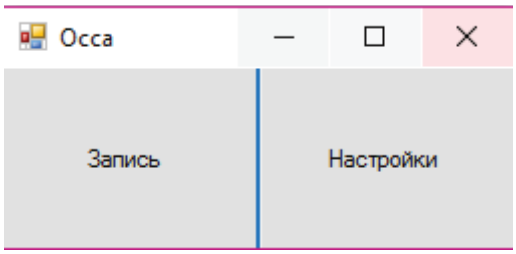

Рисунок 4.6 – Окно программы.

После нажатия настроек, открывается следующее окно, в котором предоставляется выбор между несколькими вкладками:

– добавить команду;
– отобразить команды;
– дополнительные настройки.

Открытое окно с настройками параметров представлено на рисунке 4.6.

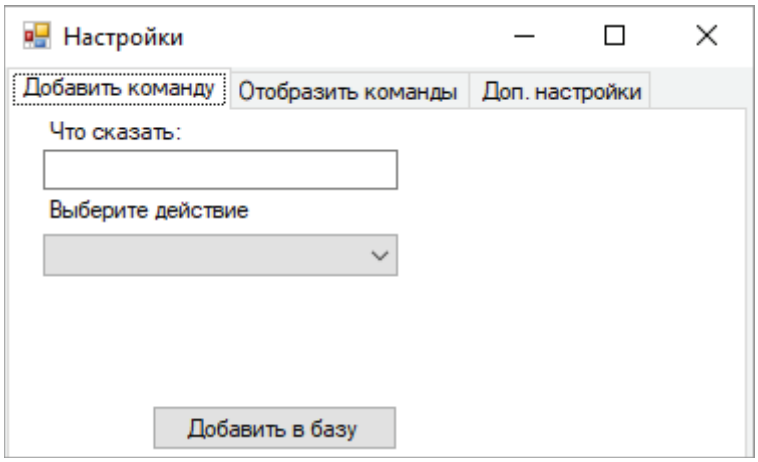

Рисунок 4.6 – Окно настроек.

Чтобы добавить команду для открытия файла или программы, нужно ввести желаемую фразу, которая будет открывать файл или программу, выбрать из раскрывающего списка желаемое действие «Открыть файл или программу» и ввести её в поле в ручную. Так же, можно нажать на кнопку «Выбрать» и указать путь до файла, тогда путь к исполняемому файлу установится автоматически. Для применения настройки требуется нажать «Добавит в базу».

Окно настроек, с введённым вручную исполняемым файлом представлено на рисунке 4.7.

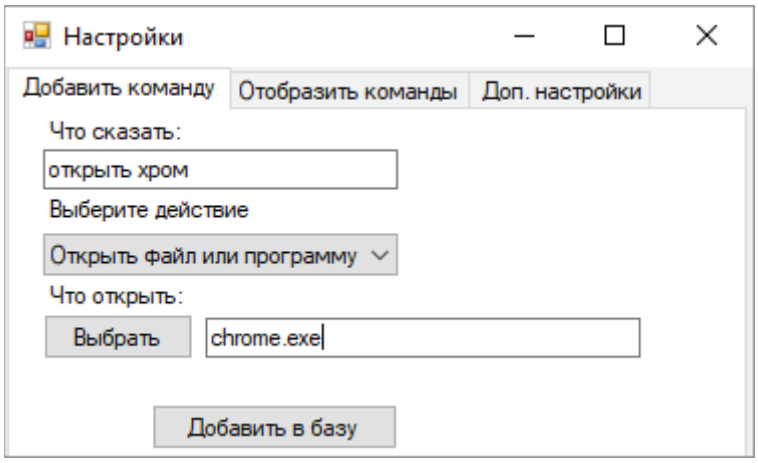

Рисунок 4.7 – Окно настроек, с введённым вручную исполняемым файлом.

Окно настроек, с введённым исполняемым файлом в автоматическом режиме представлено на рисунке 4.8.

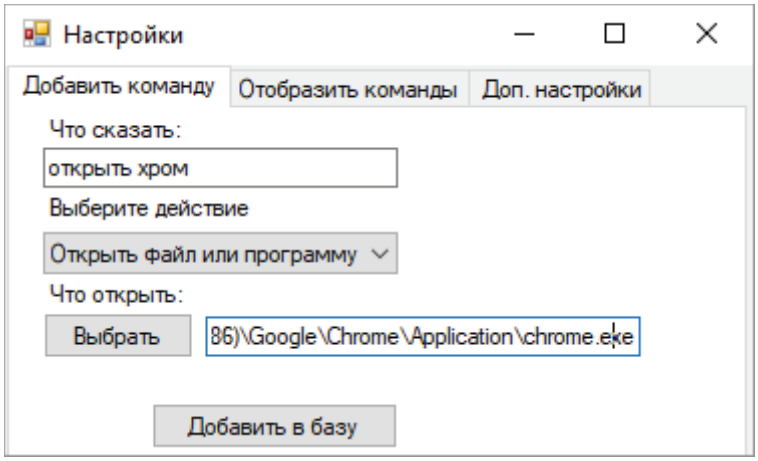

Рисунок 4.8 – Окно настроек, с введённым исполняемым файлом в автоматическом режиме.

Чтобы добавить команду для закрытия программы, нужно ввести желаемую фразу, которая будет закрывать программу, выбрать из раскрывающего списка желаемое действие «Закрыть программу» и в раскрывающемся списке выбрать процесс, затем нажать «Добавит в базу».

Окно настроек, с параметрами закрытия программ представлено на рисунке 4.9 и рисунке 4.10.

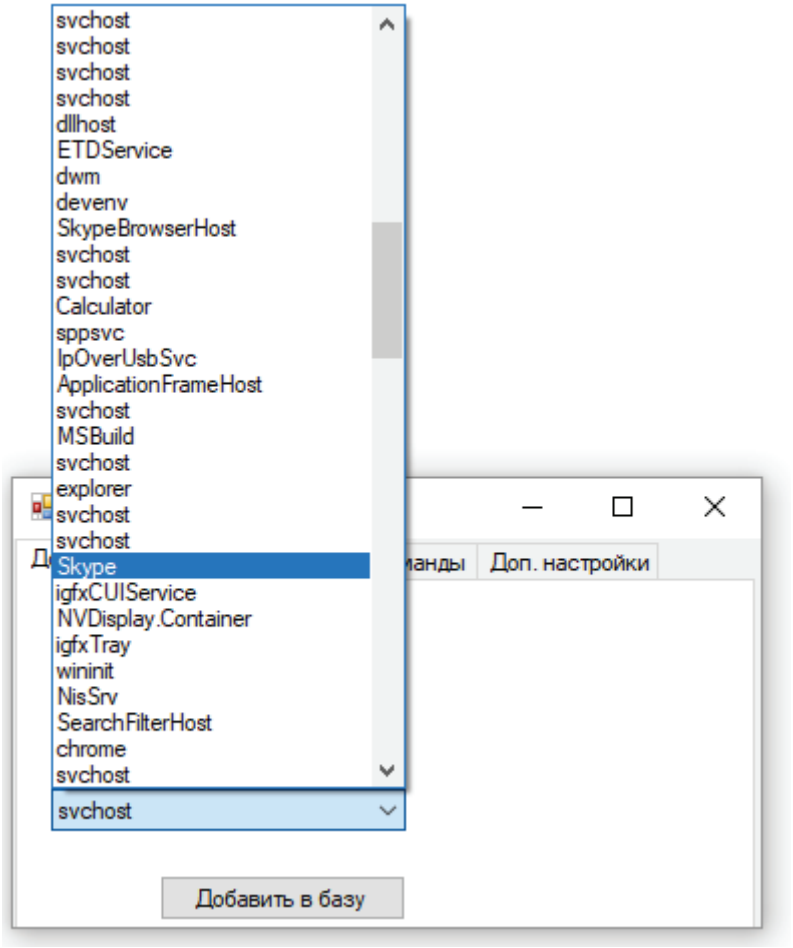

Рисунок 4.9 – Окно настроек, выбором необходимой программы для закрытия.

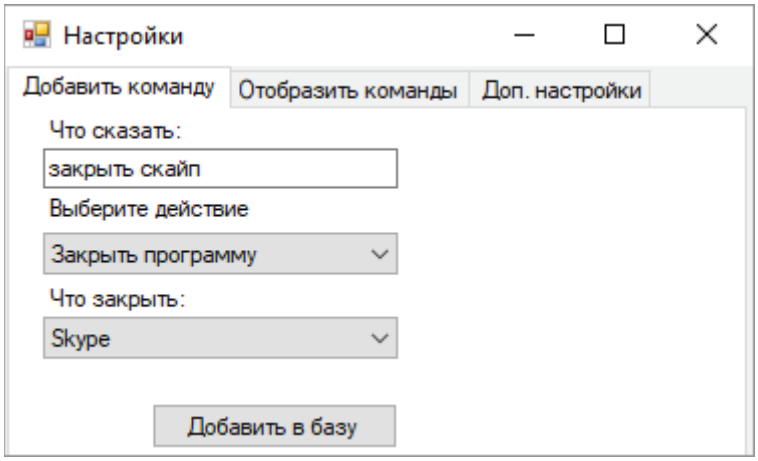

Рисунок 4.10 – Настроенное действие для закрытия программы.

Чтобы добавить команду для нажатия сочетания клавиш, нужно ввести желаемую фразу, которая будет эмулировать нажатие клавиш, выбрать из раскрывающего списка желаемое действие «Нажать сочетание клавиш», и в поле «Что нажать» нажать на клавиши, которые хотим сэмулировать, затем нажать «Добавить в базу».

Окно с настройкой сочетания клавиш представлено на рисунке 4.11.

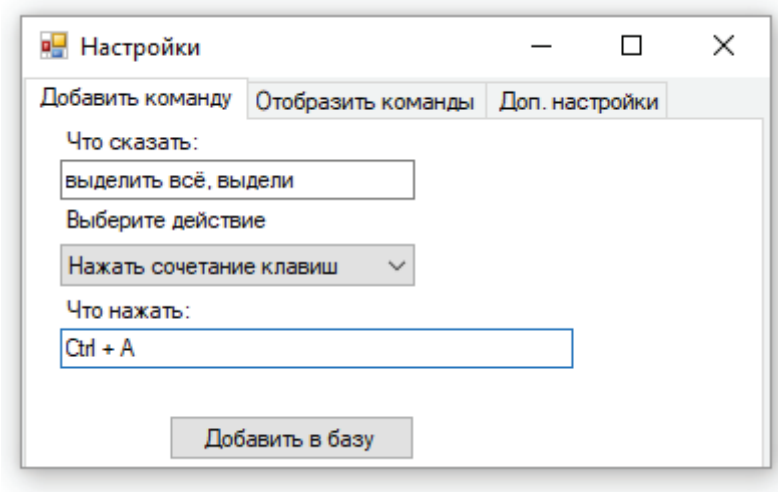

Рисунок 4.11 – Окно настройки сочетаний клавиш.

Во вкладке «Отобразить команды» представлены все команды, которые были добавлены. Он делится на 3 подсписка, это «Открыть», «Закрыть» и «Нажать», в каждой из которых есть 2 столбца, первый это «Что сказать», а второй «Что, открыть», «Что закрыть», «Что нажать» в соответствие с каждой вкладкой.

Окна с содержимым вкладки «Отобразить команды» представлено на рисунке 4.12, рисунке 4.13 и рисунке 4.14.

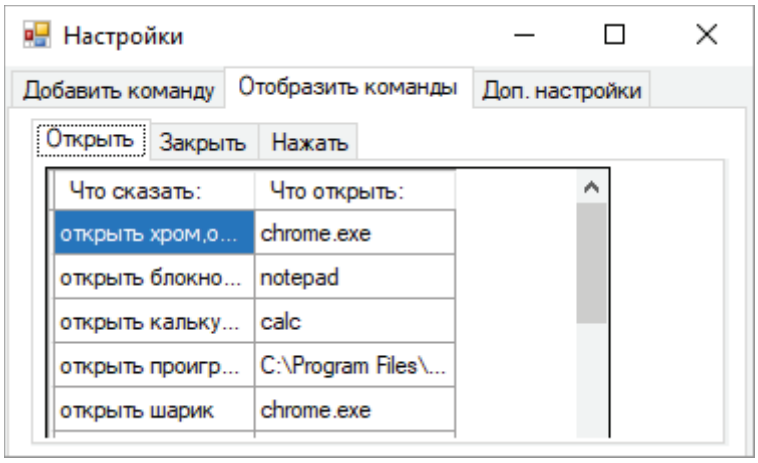

Рисунок 4.12 – Вкладка, отображающая список команд, предназначенных для запуска программ

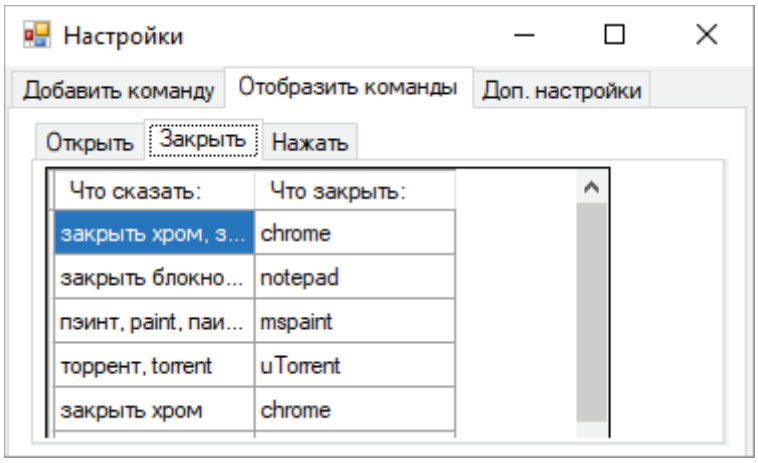

Рисунок 4.13 – Вкладка, отображающая список команд, предназначенных для закрытия программ

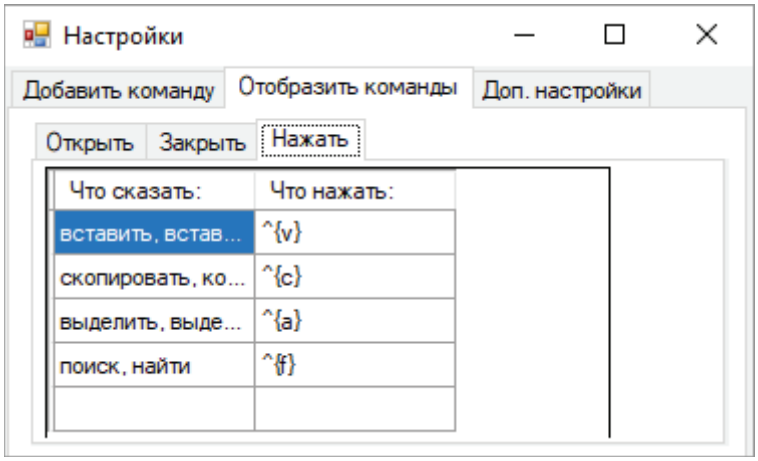

Рисунок 4.14 – Вкладка, отображающая список команд, эмулирующее нажатия сочетаний клавиш

В программном продукте были предусмотрена обработка следующих ошибок:

1. При попытке воспользоваться программой с отсутствующим интернет соединением, пользователь будет проинформирован соответствующей ошибкой.

Окно с оповещением об отсутствующем интернет-соединении представлено на рисунке 4.15.

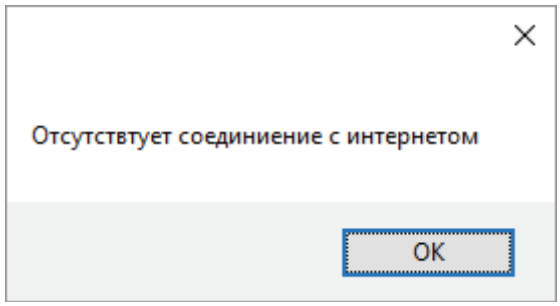

Рисунок 4.15 – Обработка ошибки интернет-соединения

2. При попытке воспользоваться программой с неподключенным к компьютеру микрофоном, пользователь будет проинформирован соответствующей ошибкой.

Окно с оповещением об отсутствии подключенного микрофона представлено на рисунке 4.16.

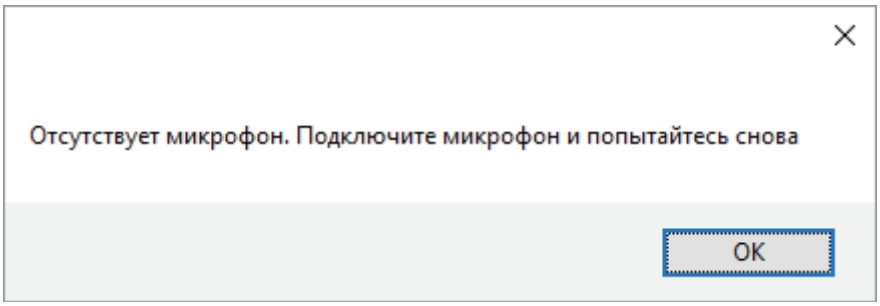

Рисунок 4.16 – Обработка ошибки неподключенного микрофона

3. При попытке ввести в путь файла данных символов, пользователь будет проинформирован следующей ошибкой.

Окно с оповещением о некорректном пути к файлу представлено на рисунке 4.17.

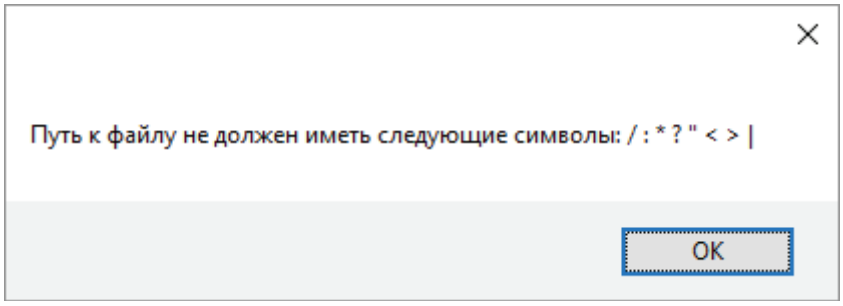

Рисунок 4.17 – Обработка ошибки некорректного пути к файлу

Так как в операционной системе Windows есть недопустимые символы в задание имени файла, а именно, звездочка (*), вертикальная черта (|), двоеточие (:), двойные кавычки ("), меньше (<), больше (>), вопросительный знак (?), косая черта (/), то в настройках в формах добавление команды на открытия файла не должно быть этих символов в пути файла.

Для настройки функций диктовки теста, поиска в интернет, а так же редактирования клавиши для записи существует вкладка «Дополнительные настройки». Во вкладке «Диктовка текста» можно задать желаемую фразу, которая будет включать режим диктовки. Настройка этого параметра представлена на рисунке 4.18. Во вкладке «Клавиша записи» есть возможность поменять клавишу, которая будет инициировать запись. Настройка этого параметра представлена на рисунке 4.19. Во вкладке «Поиск в интернете» можно задать желаемую фразу, которая будет активировать режим поиска в интернет. Настройка этого параметра представлена на рисунке 4.20.

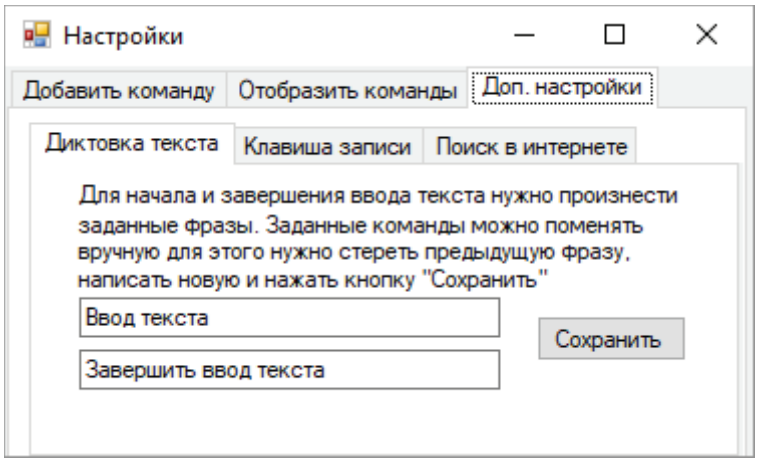

Рисунок 4.18 – Настройка режима диктовки текста

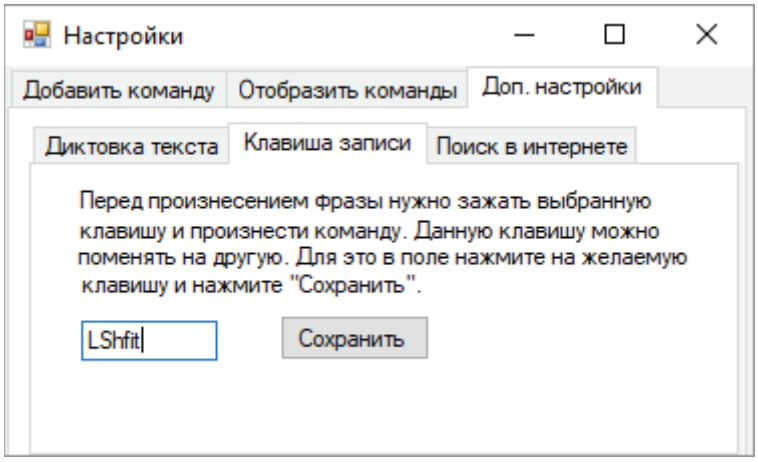

Рисунок 4.19 – Настройка клавиши для активации записи

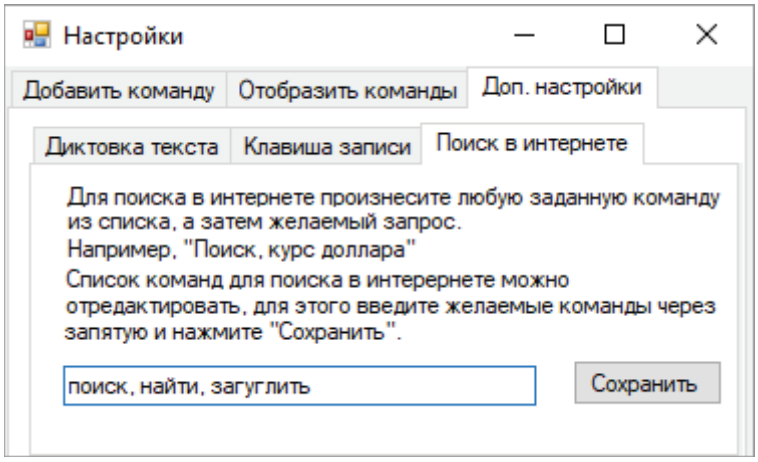

Рисунок 4.20 – Настройка поиска в интернет

ЗАКЛЮЧЕНИЕ

При разработке приложения был проведен анализ существующих приложений, позволяющих удаленное управление функциями персонального компьютера с помощью голосовых команд.

Проанализировав полученные данные, был составлен полный список требований к разрабатываемому приложению:

- высокая скорость работы программы (до 1.5 секунд задержки);
- отсутствие ограничений на количество команд;
- отсутствие зависимости результата от точной команды;
- удобный и современный пользовательский интерфейс;
- полностью бесплатное использование;
- поддержка русского языка.

Исходя из списка требований, была построена диаграмма использования и объектная модель приложения. Также, для реализации программного обеспечения была выбрана программная платформа C#. Данная платформа активно развивается и поддерживается, имеет хорошую документацию и средства быстрой разработки. Кроме того, в ходе проектирования была разработана структура интерфейса пользователя.

В итоге была разработана пробная версия продукта, которая удовлетворяет всем требования технического задания.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОРЫНЫЧ Система автоматического распознавания речи [Электронный ресурс]. – Режим доступа: http://www.rusdoc.ru/material/manual/gor/gor.html. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
2. Голосовое управление компьютером [Электронный ресурс]. – Режим доступа: http://nastroyvse.ru/programs/review/upravlenie-kompyuterom-s-pomoshhyu-golosa.html. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
3. Голосовое управление компьютером Windows 7 и 10 [Электронный ресурс]. – Режим доступа: http://windowsprofi.ru/win10/sposoby-golosovogo-upravleniya-kompyuterom-windows-7-i-10.html. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
4. Braina – Artificial Intelligence Software for Windows [Электронный ресурс]. – Режим доступа: https://www.brainasoft.com/braina/. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
5. Cortana | Your Intelligent Virtual & Personal Assistant | Microsoft [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/en-us/windows/cortana. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
6. Бабич, А. В. Введение в UML, 6. Лекция: Диаграммы прецедентов: крупным планом [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/1007/229/lecture/5962. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
7. Гайсарян, С.С. Объектно-ориентированное проектирование [Электронный ресурс]. – Режим доступа: http://www.mista.ru/oop_book/. – (Дата обращения: 28.05.2017).
8. Radeck, Kirk C# and Java: Comparing Programming Languages [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/en-

us/library/ms836794.aspx. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).

9. Рубанов, С. Почему C# [Электронный ресурс]. – Режим доступа: https://www.kv.by/content/337688-pochemu-c. – (Дата обращения: 28.05.2017).
10. Visual Studio IDE [Электронный ресурс]. – Режим доступа: https://www.visualstudio.com/ru. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
11. Git [Электронный ресурс]. – Режим доступа: https://git-scm.com. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
12. Bitbucket | The Git solution for professional teams [Электронный ресурс]. – Режим доступа: https://bitbucket.org. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
13. Andriy Lesyuk. Mastering Redmine./ Andriy Lesyuk// – Packt Publishing, 2013 – 1. – С. 343. – Англ.
14. NAudio - Documentation [Электронный ресурс]. – Режим доступа: https://naudio.codeplex.com/documentation. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
15. Общие сведения о Windows Forms [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/8bxxy49h(v=vs.110).aspx. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
16. JSON [Электронный ресурс]. – Режим доступа: http://www.json.org/json-ru.html. – Заглавие с экрана. – (Дата обращения: 28.05.2017).
17. Speech API – Speech Recognition | Google Cloud Platform [Электронный ресурс]. – Режим доступа: https://cloud.google.com/speech/. – Заглавие с экрана. – Англ. – (Дата обращения: 28.05.2017).
18. Рейтинг программистов [Электронный ресурс]. – Режим доступа: https://willdev.in/. – Заглавие с экрана. – (Дата обращения: 28.05.2017).