ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

СОГЛАСОВАНО
Научный руководитель,
заместитель директора
по работе с НИУ ВШЭ, 1С

_____ Н. Ю. Старичков
«___» _____ 2023 г.

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, канд. техн. наук

_____ В. В. Шилов
«___» _____ 2023 г.

ГОЛОСОВАЯ КЛАВИАТУРА

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.02.02-01 81 01-1-ЛУ

Исполнитель
студент группы БПИ202
_____ / Д. А. Шагаров /
«____»_____ 2023 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

Москва 2023

УТВЕРЖДЕН
RU.17701729.02.02-01 81 01-1-ЛУ

ГОЛОСОВАЯ КЛАВИАТУРА

Пояснительная записка

RU.17701729.02.02-01 81 01-1

Листов 25

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

Москва 2023

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

1.1 Наименование программы

Наименование программы: «Локальный сервер для приложения Голосовая клавиатура**».**

Наименование программы на английском языке: «Local server for the Voice keyboard application».

Наименование программы для пользователя: «Voice Keyboard server».

1.2 Краткая характеристика области применения программы

Программа является локальным сервером для приложения Голосовая клавиатура.

Локальный сервер позволяет приложению имитировать нажатия клавиш на клавиатуре, прослушивать микрофон и распознавать речи пользователя.

Также сервер предоставляет интерфейс для приложения по взаимодействию командами: добавление клавиш, голосовых команд, их изменение и удаление.

Программа поставляется в виде исполняемых файлов для операционных систем Windows [24], MacOS [16].

2 НАЗНАЧЕНИЕ РАЗРАБОТКИ

2.1 Функциональное назначение

Программа предназначена для обеспечения основной функциональности приложения “Голосовая клавиатура” на различных операционных системах.

Программа позволяет имитировать нажатия клавиш на клавиатуре, прослушивать микрофон и распознавать речи пользователя; хранить информацию о сочетаниях клавиш и голосовых командах и представляет интерфейс для взаимодействия с ней.

2.2 Эксплуатационное назначение

Программа должна эксплуатироваться в качестве локального сервера при разработке интерфейсной части приложения “Голосовая клавиатура”. Другие виды эксплуатации данной программы не рассматриваются.

3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1 Постановка задачи для разработки программы

Сервер должен обеспечивать возможность выполнения следующих функций:

1. Хранение данных приложения в локальной файловой системе
2. Обработка запроса на добавление команды и сочетания клавиш, которое активируется данной командой.
3. Обработка запроса редактирования команд, назначенных сочетанию клавиш.
4. Обработка запроса редактирования сочетания клавиш, назначенного командам.
5. Обработка запроса удаления команды и назначенного ей сочетания клавиш.
6. Обработка запроса получения всех сохраненных команд и сочетаний клавиш.
7. Распознавание произнесенной пользователем команды, начинающейся с специального ключевого слова, и активация сочетания клавиш, назначенного этой команде.
8. Печать распознанного текста на русском языке после произнесения специальной команды “напечатай”.
9. Экспорт настроек команд и сочетаний клавиш в файл в формате JSON [14].
10. Импорт словаря команд и сочетаний клавиш из файла формата JSON [14], созданного данным приложением.

Перечисленные требования должны поддерживать работу с речью на русском языке.

3.2 Описание и обоснование алгоритма функционирования программы

3.2.1 Описание алгоритма функционирования программы

При запуске локальный сервер принимает один аргумент (platform) – операционная система, на которой запускается сервер (возможные варианты - windows или macos). Программа считывает конфигурационный файл, в котором содержатся основные параметры приложения, связанные с настройкой захвата аудио, конфигурацией модели распознавания голоса в текст и сетевой адрес, на котором будет работать сервер.

В программе есть два основных потока, работу которых контролирует отдельный класс ThreadController. Этот класс осуществляет корректный запуск потоков и их мягкую остановку при получении сигнала завершения процесса.

В первом потоке происходит основное взаимодействие с операционной системой пользователя, на которой запущена программа. Взаимодействие основных классов происходит внутри класса VoiceKeyboard. В методе run() данного класса происходит захват

аудиопотока, его распознавание в текст и выполнение распознанной команды. Архитектура построена так, что эти три действия делегируются трем отдельным классам.

Класс VoskRecognizer реализует интерфейс, описанный в абстрактном классе Recognizer. VoskRecognizer делегирует прослушивание микрофона пользователя классу MicrophoneListener.

MicrophoneListener реализует интерфейс абстрактного класса Listener и позволяет реактивно передавать аудиопоток микрофона пользователя внутри приложения при помощи библиотеки PyAudio [19], которая использует драйвер PortAudio [17].

VoskRecognizer читает этот аудиопоток и с помощью offline-модели для распознавания речи в текст Vosk Russian small [21] преобразует аудиопоток в строку Python [21]. Если данная строка содержит одно или более вхождение слова- активации “клава”, то полученная строка передается классу PynputKeyboard, который реализует интерфейс абстрактного класса Keyboard.

Когда экземпляр PynputKeyboard был создан, объект считал файл, в котором содержится отображение клавиш кодов виртуальной клавиатуры на названия клавиш физической клавиатуры (файл специфичен для каждой операционной сисетмы, за его выбор отвечает параметр platform, описанный выше). Так же при создании читается файл с сохраненными на данный момент голосовыми командами и сочетаниями клавиш.

Когда PynputKeyboard получает строку с распознанной речью пользователя, находятся все команды, расположенные после слова активации “клава”. Далее происходит сопоставление каждой команды с словарем известных команд и сочетаний клавиш. Сопостовление происходит перебором распознанной команды и каждой команды из словаря, при котором к двум строкам применяется алгоритм подсчета расстояния Левенштейна [28]. Если сходство по расстоянию Левенштейна [28] для каждой комманды ниже порога, определенного в конфигурационном файле, никакая команда не выполняется. Иначе выбирается сочетание клавиш команды с наивысшим сходством с точке зрения расстояния Левеншейна [28]. При равном сходстве для нескольких команд, выбор команды не детерминирован.

Выбранное сочетание клавиш отображается на коды виртуальной клавиатуры, полученные при создании из файла. Эти коды используются внутри PynputKeyboard в библиотеке Pynput [20] для активации клавиш на устройстве пользователя.

Во втором потоке запускается gRPC [26] сервер, который прослушивает запросы к указанному в конфигурационном файле порту от приложения-клиента. Каждый запрос к серверу обрабатывается в отдельном потоке.

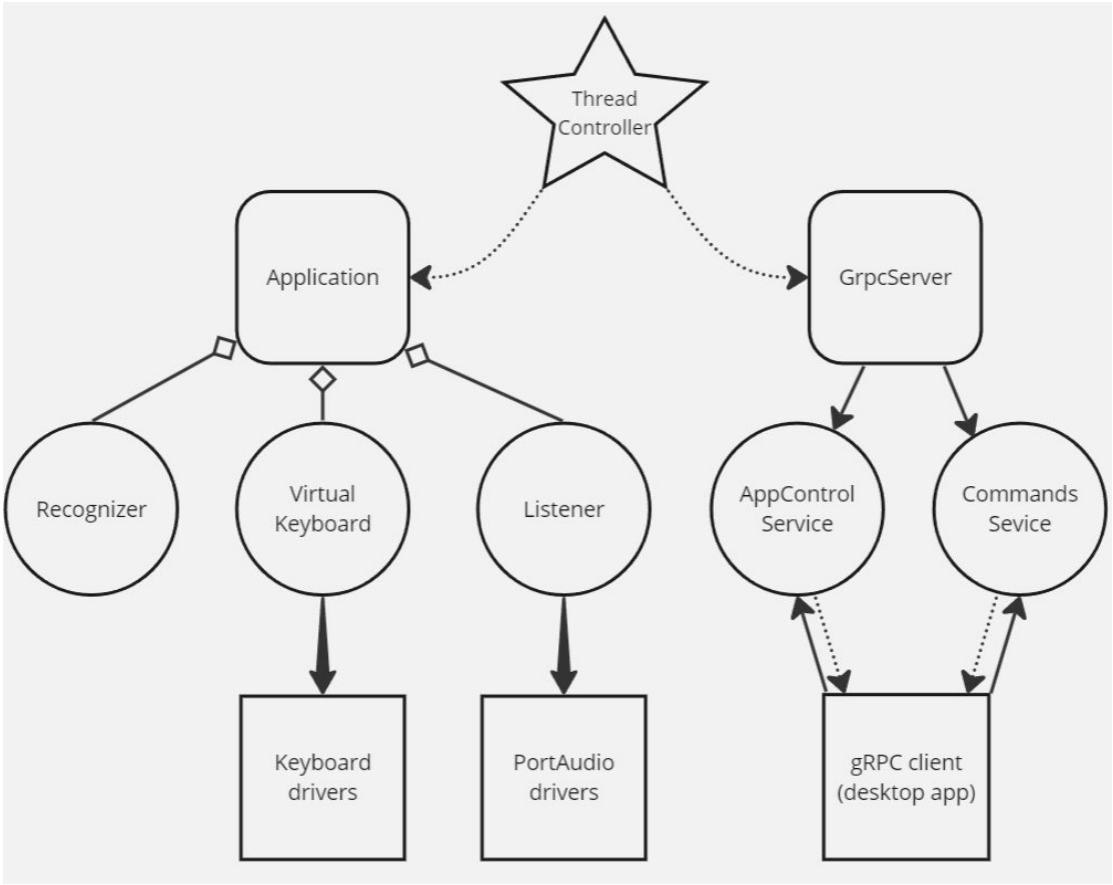
Сервер использует два сервиса. Первый позоваляет добавлять, удалять и читать команды и сочетания клавиш, импортировать и экспортировать файл с командами и сочетаниями клавиш. Этот функционал предоставлен классом CommandsService.

Второй класс сервиса – AppControlService. Он предоставляет непосредственно управление данной программой и позволяет включать и выключать распознавание команд пользователя приложением.

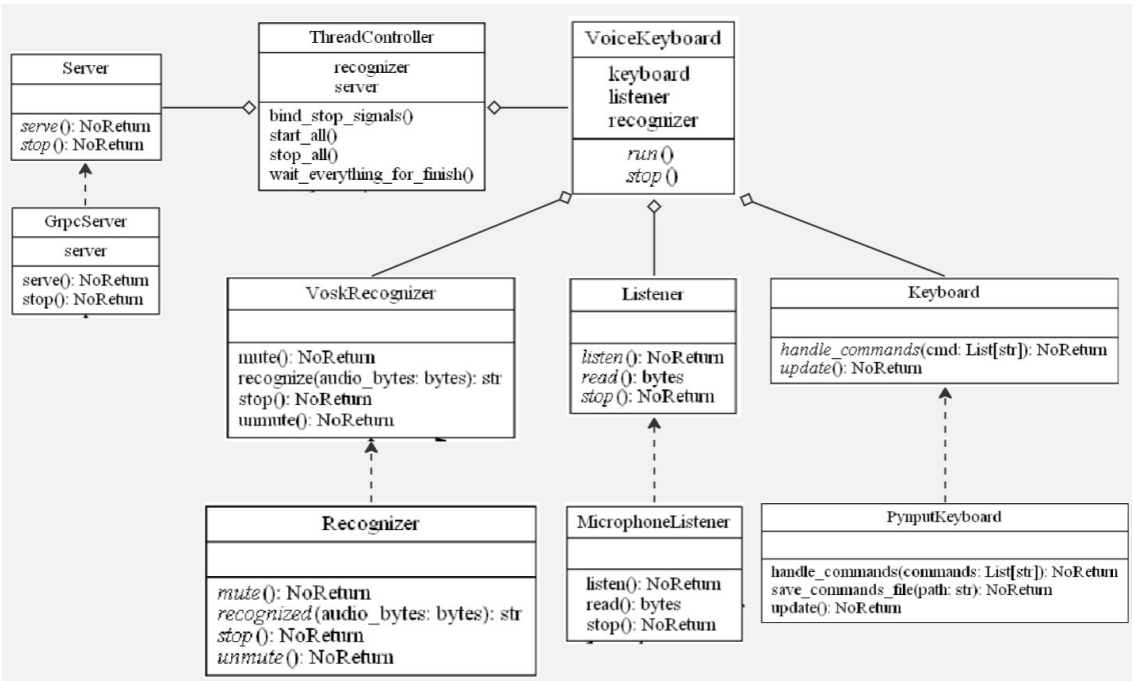
Программа при запуске читает файл, расположенный в специальной папке (папка создается, если ее нет), расположение которой зависит от операционной системы пользователя. Данный файл хранит отображение списка команд пользователя на сочетания клавиш. Это необходимо для сохранения состояния приложения при выходе из него.

Любой некорректный ввод, полученный при вызове gRPC [26] метода обрабатывается на стороне сервера. В этом случае клиент получает в ответе ошибку с детальной информацией о ней. Аналогично сервер информирует о невозможности совершения той или иной операции.

Изображение 2. Краткая схема работы приложения.



Изображение 3. Диаграмма классов приложения.



3.2.2 Обоснование выбора алгоритмов

Алгоритм передачи потока данных и управления внутри программы реализован с использованием идей из книги “Чистая архитектура” [27] Р. Мартина. Классы VoskRecognizer, MicrophoneListener, PynputKeyboard реализуют интерфейс, описанный в соответствующих абстрактных классах. Это нужно для того, чтобы в классах использовать принцип инверсии зависимостей внутри класса VoiceKeyboard.

Такой алгоритм передачи потока данных и управления части программы являются слабосвязанными и легкозаменяемыми (классы распознавания речи в текст, виртуальной клавиатуры и прослушки аудио легко подменить на другие реализации в случае необходимости).

В приложении используется распознавание речи с помощью алгоритмов библиотеки Vosk [22], так как она не требует подключения к интернету, а также предоставляет несколько моделей разной сложности. Для приложения была выбрана маленькая версия модели для того, чтобы распознавание было как можно быстрее, а размер программы как можно меньше, что повышает удобство пользования программой.

Для фразы пользователя с командами из словаря использован алгоритм вычисления сходства строк на основе расстояния Левенштейна [28]. Алгоритм выбран как наиболее подходящий алгоритм для нахождения команды, наиболее похожей на распознанную из речи пользователя, так как возможны ошибок на этапе распознавания речи в текст и возможных оговорок в речи пользователя. Ошибки такого рода наибольшим образом на ошибки, выявляемые подсчетом редакционного расстояния между строками.

Помимо этого, алгоритм подсчета расстояния Левенштейна [28] между двумя строками имеет хорошую временную сложность и сложность по памяти. В данной программе используется библиотека Levenshtein [15], реализованная на языке C, что позволяет данному алгоритму работать быстрее, нежели другие возможные реализации алгоритмов на языке Python [21].

3.3 Описание и обоснование метода организации входных и выходных данных.

3.3.1 Описание организации входных и выходных данных

Входные данные для приложения представляют собой:

1) Аудиопоток, захватываемый микрофоном пользователя.

2) Файлы форматов JSON [14], YAML [25].

3) Строковый параметр, указываемый при запуске приложения.

4) Сериализованные в бинарный формат при помощи механизма Protocol Buffers [18] структурированные сообщения, передаваемые в соответствии с концепцией gRPC [26].

Выходные данные представляют собой:

1) Эмулированные с помощью виртуальной клавиатуры нажатия клавиш клавиатуры.

2) Файлы формата JSON [14].

3) Сериализованные в бинарный формат при помощи механизма Protocol Buffers [18] структурированные сообщения, передаваемые в соответствии с концепцией gRPC [26].

3.3.2 Обоснование выбора метода организации входных и выходных данных

Описанные методы позволяют предоставить эффективное и удобное взаимодействие между клиентом и сервером благодаря технологии gRPC [26].

Файлы формата JSON [14] позволяют удобно хранить состояние приложения, файл формата YAML [25] используется в качестве конфигурационного, что является популярным, удобным и надежным решением.

Строковый параметр при запуске приложения необходим для обеспечения надежности выбора операционной системы, на которой используется программа.

Захват аудиопотока и эмуляция нажатий клавиш являются частью функциональных требований к системе.

3.4 Описание и обоснование выбора технических и программных средств

3.4.1 Состав технических и программных средств

Для работы приложения необходимо устройство с операционной системой Windows [24] 10 и выше или MacOS [16] 13.0 и выше, характеристики которого позволяют работать и пользоваться всеми основными функциями данного приложения.

Для надежной и бесперебойной работы программы требуется следующий состав технических средств:

1. Оперативная память не меньше 2 Гб.

2. Свободное место на встроенной памяти не меньше 1 Гб.
3. Микрофон

Для корректной работы приложения на персональном компьютере пользователя должно быть установлено: пакет PortAudio [17].

Также должны быть установлены драйверы для работы с микрофоном и клавиатурой.

3.4.2 Обоснование выбора технических и программных средств

Вышеописанные условия обусловлены функциональностью программного дополнения, примерным размером исполняемых файлов, а также минимальными системными требованиями целевых операционных систем.

Наличие драйверов для работы с микрофоном и клавиатурой необходимо для обеспечения захвата аудиопотока с помощью микрофона пользователя и эмулирования нажатия клавиш клавиатуры соответственно.

4 ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1 Предполагаемая потребность

Программа предназначена для эксплуатации в качестве локального сервера при разработке интерфейсной части приложения “Голосовая клавиатура”.

Предполагаемый конечный пользователь – разработчики приложения для настольного компьютера “Голосовая клавиатура”.

4.2 Ориентировочная экономическая эффективность

В рамках данного программного проекта расчет экономической эффективности не предусмотрен.

4.3 Технические преимущества по сравнению с аналогами

Программа предназначена исключительно для использования в приложении “Голосовая клавиатура”. Иных образцов или аналогов с необходимым для данного приложения функционалом выявлено не было.

5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

4) ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001;

5) ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001;

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

7) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

8) ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

9) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

10) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

11) ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

12) ГОСТ 19.602-78. ЕСПД. Правила дублирования, учета и хранения программных документов, выполненных печатным способом. — Москва: Стандартинформ, 2005

13) ГОСТ 19.603-78 Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001;

14) JSON. [Электронный ресурс] – URL: https://www.json.org/json-en.html режим доступа: свободный (дата обращения: 21.01.2023).

15) Levenshtein. [Электронный ресурс] – URL: https://pypi.org/project/Levenshtein/ режим доступа: свободный (дата обращения: 13.02.2023).

16) MacOS (операционная система). [Электронный ресурс] – URL: https://developer.apple.com/macos/ режим доступа: свободный (дата обращения: 03.01.2023).

17) PortAudio. [Электронный ресурс] – URL: http://www.portaudio.com/ режим доступа: свободный (дата обращения: 17.01.2023).

18) Protocol Buffers. [Электронный ресурс] – URL: https://protobuf.dev/ режим доступа: свободный (дата обращения: 02.02.2023).

19) PyAudio. [Электронный ресурс] – URL: https://people.csail.mit.edu/hubert/pyaudio/ режим доступа: свободный (дата обращения: 17.01.2023).

20) Pynput. [Электронный ресурс] – URL: https://pypi.org/project/pynput/ режим доступа: свободный (дата обращения: 12.01.2023).

21) Python (язык программирования). [Электронный ресурс] – URL: https://www.python.org/ режим доступа: свободный (дата обращения: 03.01.2023).

22) Vosk Russian small. [Электронный ресурс] – URL: https://alphacephei.com/vosk/models режим доступа: свободный (дата обращения: 07.01.2023).

23) Vosk. [Электронный ресурс] – URL: https://alphacephei.com/vosk/ режим доступа: свободный (дата обращения: 07.01.2023).

24) Windows (операционная система). [Электронный ресурс] – URL: https://support.microsoft.com/ru-ru/windows режим доступа: свободный (дата обращения: 03.01.2023).

25) YAML. [Электронный ресурс] – URL: https://yaml.org/ режим доступа: свободный (дата обращения: 21.01.2023).

26) gRPC. [Электронный ресурс] – URL: https://opensource.google/projects/grpc/ режим доступа: свободный (дата обращения: 02.02.2023).

27) Мартин Р.М. (2018). Чистая архитектура (дата обращения: 10.11.2022).

28) Расстояние Левенштейна. [Электронный ресурс] – URL: https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance/ режим доступа: свободный (дата обращения: 13.02.2023).

ПРИЛОЖЕНИЕ 1 ТЕРМИНОЛОГИЯ

Персональный компьютер (ПК) — однопользовательская ЭВМ, имеющая эксплуатационные характеристики бытового прибора и универсальные функциональные возможности.

Сочетание клавиш — разновидность интерфейса взаимодействия с вычислительным устройством, представляющая собой нажатие кнопки/клавиши на клавиатуре, которому назначено некое действие - команды, исполняемые данной системой.

Импорт — добавление данных, вставка данных из внешних источников в текущий файл/документ/базу данных.

Экспорт — сохранение документа программы не в родном для программы формате файла, а в формате другой программы.

Мягкая остановка — остановка, при которой завершение работы происходит только после того, как все ожидающие процессы завершены (новые процессы не должны запускаться и новые запросы не должны приниматься) и все ожидающие процессы завершены. Так же все открытые подключения должны быть закрыты.

Модель распознавания речи (голоса) в текст — алгоритм на основе машинного обучения, способный из аудиопотока речи получать текст того, что в нем говорится.

ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ

В данном приложении описаны только те классы и функции, которые не являются реализациями сторонних библиотек и не являются сгенерированными сторонними библиотеками.

Таблица 1. Описание и функциональное назначение классов и отдельных функций.

Название	**Тип**	**Назначение**
ThreadController	Класс	Запуск и мягкая остановка двух основных потоков приложения за выполнение функциональных требований по эмуляции активаций сочетаний клавиш по голосовой команде
VoiceKeyboard	Класс	Основной класс приложения, отвечающий за
Recognizer	Абстрактный класс	Описывает интерфейс для распознавания речи и обработки команд
VoskRecognizer	Класс	Реализует интерфейс Recognizer. Использует для распознавания аудиопотока в текст библиотеку Vosk [22]
Listener	Абстрактный класс	Описывает интерфейс для захвата внешнего аудиопотока
MicrophoneListener	Класс	Реализует интерфейс Recognizer. Использует для прослушивания микрофона пользователя библиотеку PyAudio [19]
Keyboard	Абстрактный класс	Описывает интерфейс для эмуляции нажатия сочетания клавиш на клавиатуре и взаимодействию с файлом

		сохраненных команд и сочетаний клавиш
PynputKeyboard	Класс	Реализует интерфейс Keyboard. Использует для эмуляции нажатия клавиш виртуальной клавиатурой библиотеку Pynput
AudioConfig	Класс	Структура для хранения конфигурации для класса MicrophoneListener
Server	Абстрактный класс	Описывает интерфейс сервера
GrpcServer	Класс	Реализует интерфейс Server. В качестве протокола взаимодействия сервера и клиента используется gRPC [26].
AppControlService	Класс	Реализует gRPC [26] сервис для управления приложением
CommandsService	Класс	Реализует gRPC [26] сервис взаимодействия клиента с сохраненными командами и сочетаниями клавиш
InvalidCommandError	Класс	Ошибка при получении некорректной команды
InvalidHotkeyError	Класс	Ошибка при получении некорректного сочетания клавиш

ПРИЛОЖЕНИЕ 3
ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ, МЕТОДОВ И СВОЙСТВ

В данном приложении описаны только те классы и функции, которые не являются реализациями сторонних библиотек и не являются сгенерированными сторонними библиотеками.

Таблица 2. Описание и функциональное назначение полей, методов и свойств класса ThreadController.

Наименование	**Тип**	**Назначение**
__app	Поле	Экземпляр класса, ответственного за выполнение функциональных требований по выполнению нажатий сочетаний клавиш по голосовой команде
__server	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса Server
__active_threads	Поле	Список активных потоков
__mu	Поле	Мьютекс для исключения состояния гонки при запуске и остановки потоков
__stop_poll_time	Поле	Частота проверки, стоит ли останавливать приложение
__exit	Поле	Событие остановки приложения
stop_all	Метод	Останавливает активные потоки
bind_stop_signals	Метод	Привязывает метод, исполняемый при получении сигнала завершения приложения
wait_everything_for_finish	Метод	Реализует мягкое завершение активных потоков

Таблица 3. Описание и функциональное назначение полей, методов и свойств класса Recognizer.

Наименование	**Тип**	**Назначение**
recognize	Метод	Распознает речь из аудиопотока и возвращает распознанный текст

Таблица 4. Описание и функциональное назначение полей, методов и свойств класса VoskRecognizer.

Наименование	**Тип**	**Назначение**
recognize	Метод	Распознает речь из аудиопотока и возвращает распознанный текст
__audio_cfg	Метод	Конфигурация для настройки библиотеки Vosk [23]
__recognizer	Метод	Экземпляр класса Recognizer библиотеки Vosk [23]

Таблица 5. Описание и функциональное назначение полей, методов и свойств класса Listener.

Наименование	**Тип**	**Назначение**
listen	Метод	Начинает захват аудиопотока
stop	Метод	Останавливает захват аудиопотока
read	Метод	Предоставляет поток для чтения байт из аудиопотока

Таблица 6. Описание и функциональное назначение полей, методов и свойств класса MicrophoneListener.

Наименование	**Тип**	**Назначение**
listen	Метод	Начинает захват аудиопотока
stop	Метод	Останавливает захват аудиопотока
read	Метод	Предоставляет внешним потребителям поток байт из аудиопотока для чтения из него
__audio	Поле	Объект захвата аудиопотока библиотеки PyAudio [19]
__audio_cfg		Конфигурация для настройки библиотеки PyAudio [19]
__stream	Поле	Поток байт из аудиопотока

Таблица 7. Описание и функциональное назначение полей, методов и свойств класса Keyboard.

Наименование	**Тип**	**Назначение**
handle_commands	Метод	Выполняет эмуляцию нажатия клавиш клавиатуры
update	Метод	Обновляет информацию об отображении команд на сочетания клавиш из файла

Таблица 8. Описание и функциональное назначение полей, методов и свойств класса PynputKeyboard.

Наименование	**Тип**	**Назначение**
handle_commands	Метод	Выполняет эмуляцию нажатия сочетаний клавиш клавиатуры
update	Метод	Обновляет информацию об отображении команд на сочетания клавиш из файла
__keyboard	Поле	Экземпляр контроллера виртуальной клавиатуры
__commands	Поле	Отображение команд на сочетания клавиш
__similarity_threshold	Поле	Порог сходства строк при котором команда не игнорируется
__commands_path	Поле	Путь к файлу с отображением команд на сочетания клавиш
__mu	Поле	Мьютекс для избегания состояния гонки при взаимодействии с отображением команд на сочетания клавиш
__vk_codes	Поле	Отображение названия клавиши на ее код в виртуальной клавиатуре
save_commands_file	Метод	Сохраняет текущее отображение команд на сочетания клавиш в файл
__read_commands_file	Метод	Обновляет текущее отображение команд на сочетания клавиш в соответствии с данными из файла
__handle_command	Метод	Выполняет эмуляцию нажатия

		сочетания клавиш клавиатуры
__is_type_command	Метод	Проверяет, является ли команда “напечатай"
__handle_type_command	Метод	Выполняет команду “напечатай”
__compare_commands_by_levenshtein	Метод	Сравнивает строки с использованием расстояния Левенштейна [28]
__check_keys	Метод	Проверяет поддерживаются ли клавиши виртуальной клавиатурой

Таблица 9. Описание и функциональное назначение полей, методов и свойств класса AudioConfig.

Наименование	**Тип**	**Назначение**
chunk_size	Поле	Размер (в байтах) части аудиопотока, отправляемой в методе read за один раз
sample_format	Поле	Формат аудиопотока для PyAudio [19]
channels	Поле	Количество каналов в захватываемом аудиопотоке
frequency	Поле	Частота дискретизации аудиопотока

Таблица 10. Описание и функциональное назначение полей, методов и свойств класса Server.

Наименование	**Тип**	**Назначение**
serve	Метод	Начинает работу сервера
stop	Метод	Останавливает работу сервера

Таблица 11. Описание и функциональное назначение полей, методов и свойств класса MicrophoneListener.

Наименование	**Тип**	**Назначение**
serve	Метод	Начинает работу сервера
stop	Метод	Останавливает работу сервера
__address	Метод	Адрес, на котором работает сервер

__server	Метод	Экземпляр gRPC [26] сервера
server	Свойство	Возвращает экземпляр gRPC [26] сервера

Таблица 12. Описание и функциональное назначение полей, методов и свойств класса AppControlService.

Наименование	**Тип**	**Назначение**
ChangeMicrophoneStatus	Метод	Управляет флагом, активно ли сейчас распознавание аудиопотока и выполнение эмуляции нажатия клавиш
__recognizer	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса Recognizer

Таблица 13. Описание и функциональное назначение полей, методов и свойств класса CommandsService.

Наименование	**Тип**	**Назначение**
__commands_path	Поле	Путь до файла с отображением команд на сочетания клавиш
__keyboard	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса Keyboard
__supported_vk_keys	Поле	Множество поддерживаемых клавиш виртуальной клавиатуры
__allowed_command_symbols	Поле	Множество поддерживаемых символов, из которых может состоять команда
__write_commands_file	Метод	Записывает файл с отображением команд на сочетания клавиш
__check_command_and_hotkey	Метод	Проверят команду и сочетание клавиш на корректность
__check_type_command	Метод	Проверяет похожа ли команда на команду “напечатай”
__check_command_is_correct	Метод	Проверяет команду на корректность
__check_keys_supported	Метод	Проверят что названия клавиш корректны
AddCommand	Метод	Добавляет команду и сочетание клавиш в отображение команд на сочетания клавиш

DeleteCommand	Метод	Удаляет команду и сочетание клавиш из отображения команд на сочетания клавиш
GetCommands	Метод	Возвращает отображение команд на сочетания клавиш
ImportCommands	Метод	Получает и применяет отображение команд на сочетания клавиш из файла
ExportCommands	Метод	Сохраняет отображение команд на сочетания клавиш в файл

Таблица 14. Описание и функциональное назначение полей, методов и свойств класса InvalidCommandError.

Наименование	**Тип**	**Назначение**
message	Поле	Сообщение об ошибке команды

Таблица 15. Описание и функциональное назначение полей, методов и свойств класса InvalidHotkeyError.

Наименование	**Тип**	**Назначение**
message	Поле	Сообщение об ошибке сочетания клавиш

Таблица 16. Описание и функциональное назначение полей, методов и свойств класса VoiceKeyboard.

Наименование	**Тип**	**Назначение**
run	Метод	Начинает распознавание аудиопотока и выполнение эмуляции нажатия клавиш
stop	Метод	Останавливает распознавание аудиопотока и выполнение эмуляции нажатия клавиш
mute	Метод	Приостанавливает выполнение эмуляции нажатия клавиш
unmute	Метод	Возобновляет выполнение эмуляции нажатия клавиш
__recognizer	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса

		Recognizer
__listener	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса Listener
__keyboard	Поле	Экземпляр класса, удовлетворяющего интерфейсу абстрактного класса Keyboard
__in_work	Поле	Флаг, работает ли сейчас распознавание аудиопотока и выполнение эмуляции нажатия клавиш
__mu	Поле	Мьютекс, чтобы исключить состояние гонки при запуске и остановке распознавания аудиопотока и выполнения эмуляции нажатия клавиш
__trigger	Поле	Слово, являющееся триггером для активации (началом команды пользователя)
__is_microphone_on	Поле	Флаг, активно ли сейчас распознавание аудиопотока и выполнение эмуляции нажатия клавиш

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

зм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					

Изм.	Лист	№ документа	Подпись	Дата
RU.17701729.02.02-01 81 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата