МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
/ Б.М. Суховилов /
«_____»_____20____ г.

_____

_____

(наименование темы работы (проекта)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2021.301/24.ВКР

Руководитель, должность
_____/ Г.А. Поллак /
«_____»_____2021 г.

Автор
студент группы ЭУ – 402
_____/ А.О. Воронов /
«_____» _____2021 г.

Нормоконтролер, доцент
_____/Е.А. Конова/
«_____»_____2021 г.

Челябинск 2021

АННОТАЦИЯ

Воронов А.О. Компьютерный голосовой помощник. – Челябинск: ЮУрГУ, ЭУ-402, 46 с., 17 ил., 13 табл., библиогр. список – 8 наим., 2 прил.

Дипломный проект выполнен с целью создания компьютерного голосового помощника для пользования людьми в повседневной жизни.

В дипломном проекте проведен сравнительный анализ голосовых помощников от отечественных и зарубежных компаний с целью определения необходимого функционала и возможностей популярных решений. Рассмотрено большинство существующих решений. По окончанию сравнительного анализа, была поставлена цель по созданию своего компьютерного голосового помощника.

С помощью интегрированной среды разработки для языка программирования Python и программных пакетов реализован компьютерный голосовой помощник «Василиса». Благодаря голосовому помощнику пользователь может голосом задавать команды, которые «Василиса» умеет выполнять, не отвлекаясь от своих занятий.

Компьютерный голосовой помощник упростит повседневную жизнь и позволит рационально использовать время, выполняя различные задачи вместо пользователя, избавив его от рутинной работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее с каждым годом количество поисковых запросов и задач, выполненных ассистентом, растёт в геометрической прогрессии. Голосовые помощники продолжают заполнять нашу жизнь.

Пользователь, который использует голосовые помощники, имеет возможность эффективнее получать необходимую информацию, выполнять обыденные задачи и использовать сохраненное время для своих нужд.

По данным Searchengineland, почти половина потребителей используют технологию голосового поиска, чтобы быстро находить ответы на вопросы в интернете. Голосовой запрос намного удобнее и проще. Данными критериями обуславливается тенденция голосовых помощников.

Статистика показывает, что большинство пользователей предпочитают использовать голосовой поиск дома, так как дома достаточно конфиденциально, в отличие от общественных мест.

Так же, по данным PricewaterhouseCoopers, 35% потребителей узнают погоду с помощью голосового ассистента. Другие 15% делают это вручную.

В будущем рынок голосовых помощников будет только расти. Статистики показывают, что множество людей доверяют им выполнение простых задач, таких как поиск в интернете, конвертация единиц измерения, анализ погоды.

Пурна Вирджи, старший менеджер в компании Microsoft, в своем блоге сообщила, что 25% поисковых запросов на рабочем столе операционной системы Windows 10 выполняются голосом с помощью ассистента по имени Cortana. Однако, Cortana не поддерживает русский язык.

Подводя итоги, на сегодняшний день, разработка и создание своего голосового помощника является актуальной задачей.

Цель выпускной квалификационной работы – разработать и создать голосовой помощник.

Задачи работы:

1) выполнить сравнительный анализ отечественных и зарубежных решений различных компаний с целью определения необходимого функционала и возможностей популярных решений.;

3) определить функционал разрабатываемого голосового помощника;

4) определить структуру и основу для разработки ассистента;

5) разработать голосовой помощник;

6) провести тесты для проверки разработанного приложения и отладить;

7) подготовить голосовой помощник к выводу на рынок.

Объект выпускной квалификационной работы – программа для десктопных устройств, выполняющая задачи по требованию пользователя.

Разработанный голосовой помощник будет применяться для выполнения обыденных и простых задач с целью рационального использования времени и повышения эффективности и производительности деятельности пользователя.

1 СРАВНЕНИЕ ОТЕЧЕСТВЕННЫХ И ЗАРУБЕЖНЫХ ТЕХНОЛОГИЙ И РЕШЕНИЙ

1.1 Сравнительный анализ голосовых помощников

В настоящее время голосовые помощники пользуются большим спросом. Они позволяют пользователю автоматизировать выполнение простых задач, сократить до минимума затрачиваемое на них время, повысив его производительность, и сосредоточить свое внимание на том, что наиболее важно.

Например, открытие браузера, ввод нужного запроса в поисковую строку, займет у пользователя от 10 до 20 секунд. В данной ситуации факторами являются: скорость ввода текста возможности компьютера. Голосовой помощник сможет это сделать от 5 до 10 секунд. На это влияют возможности компьютера, и как быстро пользователь скажет команду. Каждый человек говорит быстрее чем, печатает.

Большинство домохозяек ищут рецепты блюд в интернете. В процессе готовки им может потребоваться перевести определенное количество чайных ложек в граммы или обратно.

Многим людям требуется ставить напоминания на телефоне, чтобы не забыть про предстоящие события.

Особенно актуальны голосовые ассистенты для людей с ограниченными возможностями.

В качестве объекта исследования, выбраны следующие голосовые помощники:

1) Алиса – виртуальный голосовой помощник, созданный компанией «Яндекс». Алиса является самым известным ассистентом на российском рынке;

2) Microsoft Cortana – личный помощник от Microsoft, наибольшей популярностью Cortana пользуется в англоязычных странах;

3) Laitis – программа для управления компьютером голосом и записи под диктовку, разработанная российским программистом энтузиастом Сергеем Миколайтис

4) Typle – приложение для запуска программ.

5) Speaker – приложение для голосового управления компьютера, открытия веб-страниц и запуска программ

6) Speechka – голосовой помощник, который способен открывать веб-сайты, папки, файлы, документы.

В приложении А представлены изображения главных страниц голосовых помощников.

Предмет исследования – функционал рассматриваемых голосовых помощников и дизайн их интерфейса.

Цель исследования – проанализировать готовые решения отечественных и зарубежных компаний, в частности, дизайн и функциональную часть.

Задачи:

1) определить критерии сравнения голосовых помощников в каждом из направлений: функции программы, интерфейсные решения;

2) провести оценку программ по критериям;

3) выявить сильные и слабые стороны реализации отечественных и зарубежных голосовых ассистентов по выделенным направлениям;

4) сделать выводы и применить полученные данные в разработке собственного голосового помощника.

Критерии сравнения основных характеристик рассматриваемых голосовых помощников (ГП):

- наличие решения «Установка напоминаний»;
- наличие решения «Конвертация»;
- наличие функции перевода фраз;
- наличие решения «Анализ погоды»;

- возможность поиска и выдачи определений из Википедии;
- наличие решения «Открытие программ»;
- наличие функции вычисления математических выражений;
- наличие решения «Поиск в интернете»;
- возможность обучения ГП;
- поддержка русского языка.
- Критерии сравнения интерфейсных решений голосовых помощников (ГП):
- возможность выбора устройства ввода;
- наличие решения «Ввод текста вручную»;
- возможность установки автозапуска ГП;
- качество интерфейса (единое оформление и современность).

Методы исследования: сравнительный анализ, синтез.

Инструмент исследования: сравнительные таблицы наличия признаков.

Условные обозначения:

- «+» – реализация решения удовлетворяет критерию,
- «–» – реализация решения не удовлетворяет критерию.

Ход исследования.

Результаты исследования основных характеристик отечественных и зарубежных голосовых помощников представлены в таблице 1.

Таблица 1 – Основные характеристики отечественных и зарубежных голосовых помощников

№	Название сервиса	Наличие решения «Установка напоминаний»	Наличие решения «Конвертация»	Наличие функции перевода фраз	Наличие решения «Анализ погоды»	Возможность поиска и выдачи определений из Википедии	Наличие решения «Открытие программ»	Наличие функции вычисления математических выражений	Наличие решения «Поиск в интернете»	Возможность обучения ГП	Поддержка русского языка
1	Алиса	-	+	+	+	+	+	+	+	-	+
2	Microsoft Cortana	+	-	-	+	-	+	-	+	-	-
3	Laitis	-	-	-	-	-	+	+	+	+	+
4	Typle	-	-	-	-	-	+	-	+	+	+
5	Speaker	-	-	-	-	-	+	-	+	-	+
6	Speechka	-	-	-	-	-	+	-	+	+	+

Результаты исследования интерфейсных решений, реализованных в голосовых помощниках, представлены в таблице 2.

Таблица 2 – Интерфейсные решения, реализованные в голосовых помощниках от отечественных и зарубежных компаний

№	Название сервиса	Возможность выбора устройства ввода	Наличие решения «Ввод текста вручную»	Возможность установки автозапуска ГП	Качество интерфейса	
					Единообразный	Современный
1	Алиса	-	+	+	+	+
2	Microsoft Cortana	-	+	+	+	+
3	Laitis	+	-	+	+	+
4	Typle	+	-	+	-	-

Окончание таблицы 2

№					Качество интерфейса

	Название сервиса	Возможность выбора устройства ввода	Наличие решения «Ввод текста вручную»	Возможность установки автозапуска ГП	Единообразный	Современный
5	Speaker	+	-	+	-	-
6	Speechka	+	+	+	-	-

Результаты исследования

Проведенная оценка голосовых помощников отечественных и зарубежных компаний, предоставляет возможность подвести итоги и выводы по каждому из выбранных критериев.

1. Наличие решения «Установка напоминаний». Из всех рассмотренных голосовых помощников только Microsoft Cortana от компании Microsoft обладает возможностью планировщика событий. С помощью Cortana пользователю необязательно все запоминать. Он может обсудить с ней и распределить задачи на свой день и подтвердить, когда желательно предупредить его. Остальные программы (Алиса от Яндекса, Laitis Сергея Миколайтис, Typle, Speaker и Speechka) не умеют добавлять, изменять и редактировать напоминания.

Наличие решения «Установка напоминаний» является важным критерием по причине того, что напоминания позволяют максимально рационально распределить события пользователя и напомнить, в случае если он забудет о предстоящих делах. Благодаря этому решению пользователь будет чаще использовать голосовой помощник.

2. Наличие решения «Конвертация». Критерию соответствует только один голосовой помощник, им является Алиса от известной компании Яндекс. Алиса способна конвертировать денежные валюты, в соответствии с нынешним курсом. Остальные рассмотренные приложения для персонального компьютера не имеют функцию перевода из одной единицы измерения в другую.

3. Наличие функции перевода фраз. Реализовано только в голосовом помощнике Алиса.

4. Наличие решения «Анализ погоды». Анализ погоды позволяет пользователю определить, какая на данный момент погода и температура в определенном городе. Решение «Анализ погоды» реализовано в Microsoft Cortana и Алиса.

5. Возможность поиска и выдачи определений из Википедии. Реализовано только в программе Алиса.

6. Наличие решения «Открытие программ». С помощью функции открытия программ пользователь получает возможность не тратить время на поиск и запуск постоянно используемых программ. Ему требуется один раз заполнить строку пути программы и её название, чтобы помощник мог открывать их по голосовой команде, например, «Открой CCleaner». Функция реализована практически во всех рассматриваемых ассистентах, кроме Алисы.

7. Наличие функции вычисления математических выражений. Вычисление математических выражений позволяет быстро получать решения простых для ассистента и трудоемких для пользователя примеров. Вычисление реализовано в Алисе и в Laitis.

8. Наличие решения «Поиск в интернете». Решение «Поиск в интернете» представляет собой возможность открывать браузер с поисковым запросом по голосовой команде пользователя. Реализовано во всех рассматриваемых помощниках.

9. Возможность обучения ГП. Позволяет добавлять свои дополнительные речевые шаблоны. Реализовано в Laitis, Typle и Speechka.

10. Поддержка русского языка. Не все рассмотренные голосовые ассистенты поддерживают русский язык, Cortana не понимает его.

11. Возможность выбора устройства ввода. На некоторых компьютерах используются несколько микрофонов, поэтому выбор устройства ввода, через

которое помощник будет слушать пользователя, является достаточно важным критерием. Реализовано в Laitis, Typle, Speaker, Speechka.

12. Наличие решения «Ввод текста вручную». Ввод текста вручную, позволит пользователям, у которых неисправен или отсутствует микрофон, или имеется медленная скорость соединения с интернетом, пользоваться программой. Только в трех помощниках реализовано это решение (Алиса, Microsoft Cortana и Speechka).

13. Возможность установки автозапуска ГП. Благодаря этой функции пользователь сократит до нуля своё время, затрачиваемое на запуск голосового ассистента. Реализовано во всех рассматриваемых программах.

14. Качество интерфейса. Интерфейс разработан и реализован достаточно качественно в Алисе, Microsoft Cortana и Laitis. В остальных программах интерфейс выполнен на низком уровне.

Выводы из исследования

Проведенный сравнительный анализ позволяет сделать вывод, что на текущий момент времени на рынке России не имеется голосовой помощник для персонального компьютера, который бы соответствовал всем выбранным критериям в том или ином виде.

Большинство рассмотренных программ не имеют реализованных функций: конвертация единиц измерения, установка, редактирования и удаление напоминаний, перевод небольших фраз с русского на английский, выдача определений терминов по запросу пользователя, вычисление математических примеров, анализ погоды. Однако, можно выделить из них голосового ассистента под названием Алиса. Она соответствует десяти из 15 критериям.

1.2 Функционал и возможности голосового помощника «Василиса»

Сравнительный анализ голосовых помощников от известной компании Яндекс и Microsoft и разработанных программистами энтузиастами Laitis, Typle,

Speaker и Speechka позволил выделить основной функционал голосового помощника «Василиса».

Все возможности ассистента будут доступны для пользователя, который приобрел и установил его на компьютер.

Основные функции и решения голосового помощника «Василиса», которые будут предоставлены для пользователя:

- «Обучение ГП»;
- «Установка, редактирование и удаление напоминаний»;
- «Конвертация единиц измерения»;
- «Переводчик»;
- «Поиск определений»;
- «Вычисление математических примеров»;
- «Поиск в интернете»;
- «Анализ погоды»;
- «Открытие программ»;
- «Секундомер»;
- «Выбор устройства ввода»;
- «Ввод текста вручную»;
- «Автозапуск ГП».

1.3 Характеристики и возможности известных отечественных голосовых помощников

Алиса позволяет ускорить процесс поиска информации благодаря восприятию голосовых команд пользователя. Достаточно лишь задать нужный вопрос ассистенту, и ответ моментально высветится на экране. Пример поиска информации приведен на рисунке 1.

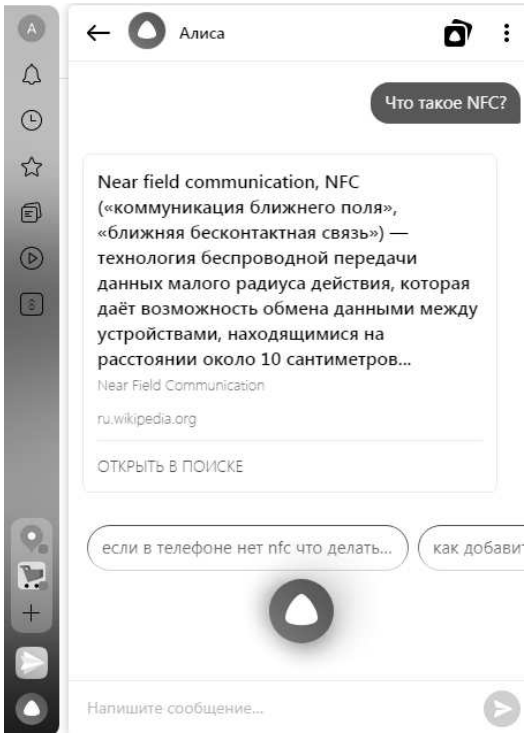

Рисунок 1 – «Поисковый запрос»

Если пользователь спросит Алису: «Что такое NFC?», Алиса найдет определение термина в Википедии и проговорит его.

Можно спросить Алису о погоде, и она выдаст актуальную информацию о погоде на сегодня или на несколько дней вперед сразу в окне приложения. Пример поиска информации о погоде приведен на рисунке 2.

Рисунок 2 – «Поиск погоды»

Алиса выводит информацию о погоде на сегодня и сколько градусов в настоящее время в Челябинске.

Быстрый перевод денежных единиц из одной валюты в другую является достаточно полезной функцией для многих пользователей. Алиса способна выполнять такие операции практически мгновенно, конвертируя, к примеру, доллары в рубли, учитывая актуальный курс. Кроме того, ассистент с легкостью вычислит сумму определенных значений, посчитает квадратный корень и найдет синус угла. Пример конвертации приведен на рисунке 3.

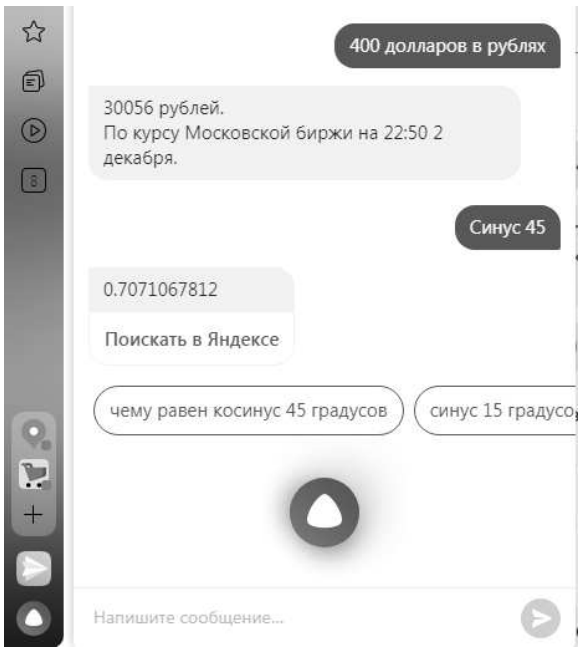

Рисунок 3 – «Конвертация и калькулятор»

Голосовой ассистент от компании Яндекс обладает еще и возможностью перевода фраз на любой язык. Пример перевода фраз приведен на рисунке 4.

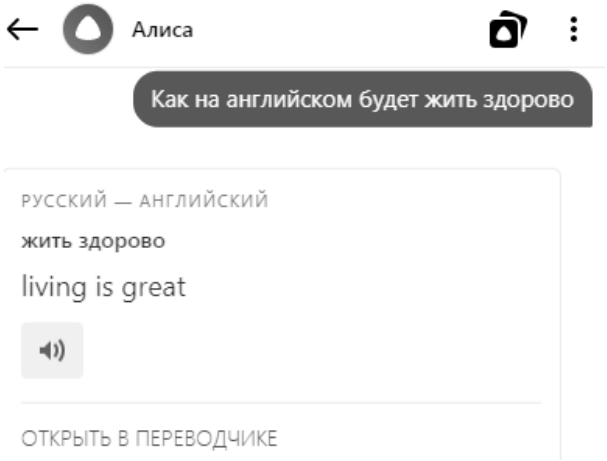

Рисунок 4 – «Перевод фраз»

Алиса в десктопной версии не обладает функцией установки напоминаний. Доказательство этому приведено на рисунке 5.

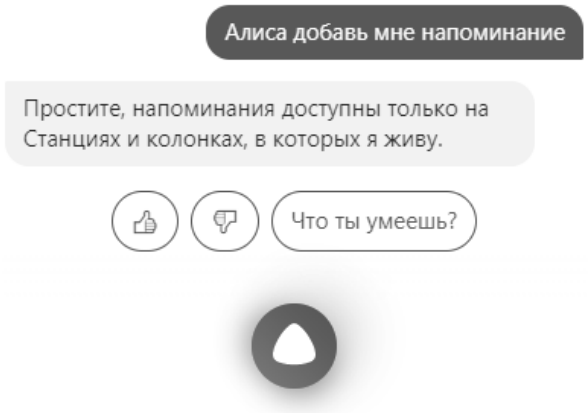

Рисунок 5 – «Напоминания»

На запрос «Алиса, добавь мне напоминание» Алиса отвечает, что данная функция недоступна в версии для персонального компьютера.

Вывод: отличный голосовой ассистент с комфортным интерфейсом и голосом. Имеет множество функций и возможностей. Из минусов: Алиса не работает без Yandex браузера, не каждый человек желает устанавливать его, в десктопной версии Алиса не умеет устанавливать напоминания и секундомер.

Laitis одна из известных программ с приятным интерфейсом.

После установки программы пользователя встречает окно приветствия, которое сообщает о требованиях программы. Laitis не может работать без интернета. Окно приветствия изображено на рисунке 6.

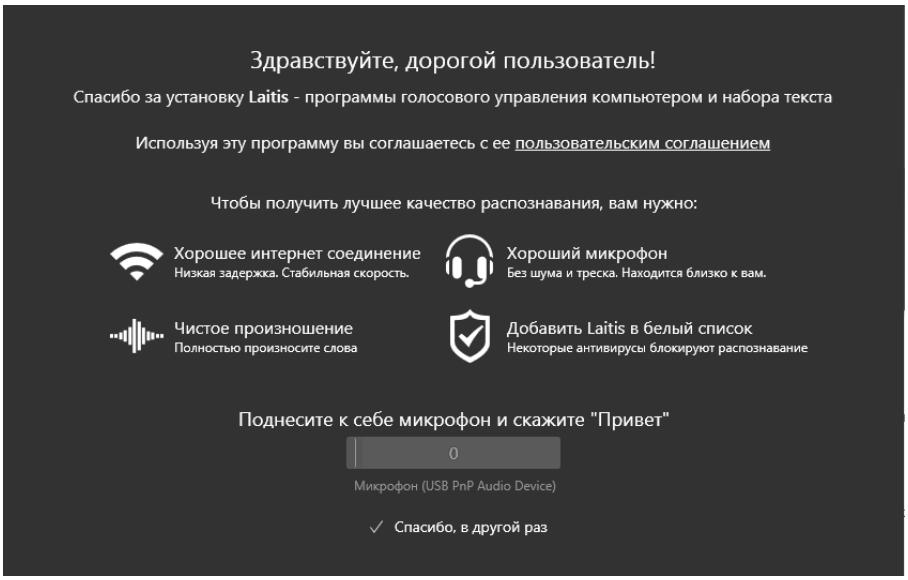

Рисунок 6 – «Окно приветствия»

Laitis умеет управлять браузером. Поэтому перед запуском программы, требуется установить расширение для используемого браузера. Окно настройки браузера представлено на рисунке 7.

Рисунок 7 – «Окно выбора браузера»

Программа способна создавать новое окно браузера, создавать новую вкладку передвигаться по вкладкам, открывать закладки, открывать веб-сайты.

На вкладке «Настройки» можно редактировать настройки программы такие как: запись голоса, сервис распознавания голоса, сервис произношения текста. Laitis, можно использовать для голосового набора текста. Окно с открытой вкладкой «Настройки» представлено на рисунке 8.

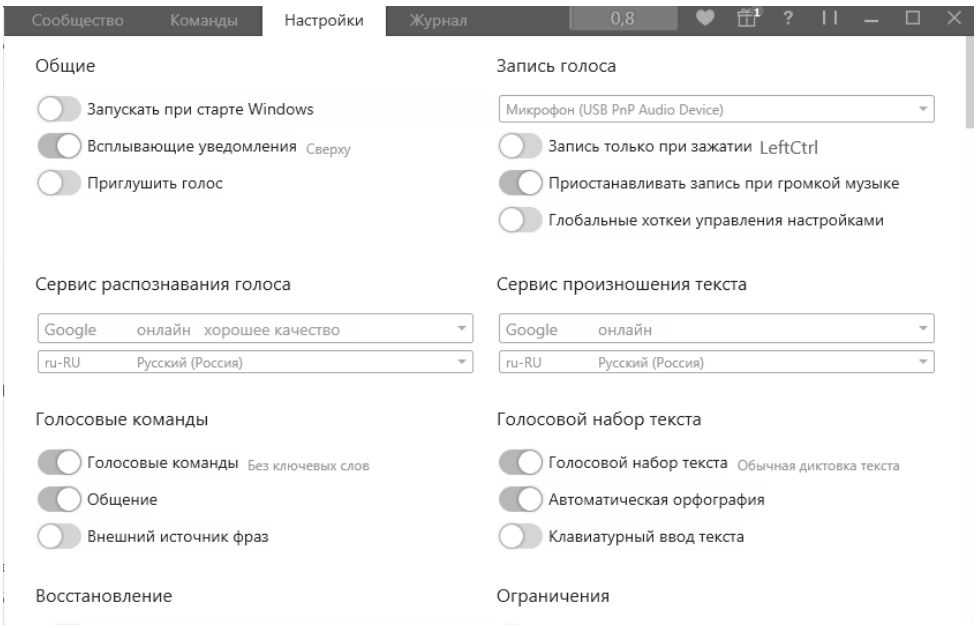

Рисунок 8 – «Вкладка настройки»

На вкладке команды программы просмотреть все возможные функции Laitis. Окно с открытой вкладкой «Команды» представлено на рисунке 9.

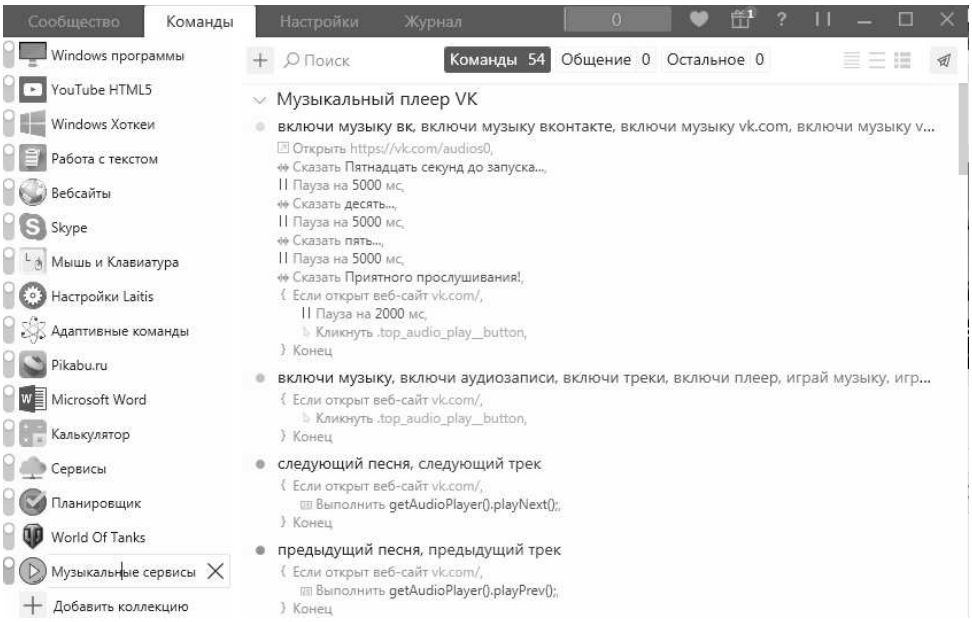

Рисунок 9 – «Команды Laitis»

Laitis умеет открывать различные Windows программы, открывать известные каналы на YouTube, нажимать горячие клавиши, работать с текстом, открывать различные Веб-сайты, открывать Skype, принимать звонки, сбрасывать

их, может найти элемент на экране и кликнуть на него, вычислять простые выражения, запускать таймер, запускать секундомер, работать с плеером в VK

В Laitis можно добавлять свои новые команды, редактировать и удалять старые. Окно добавления команд изображено на рисунке 10.

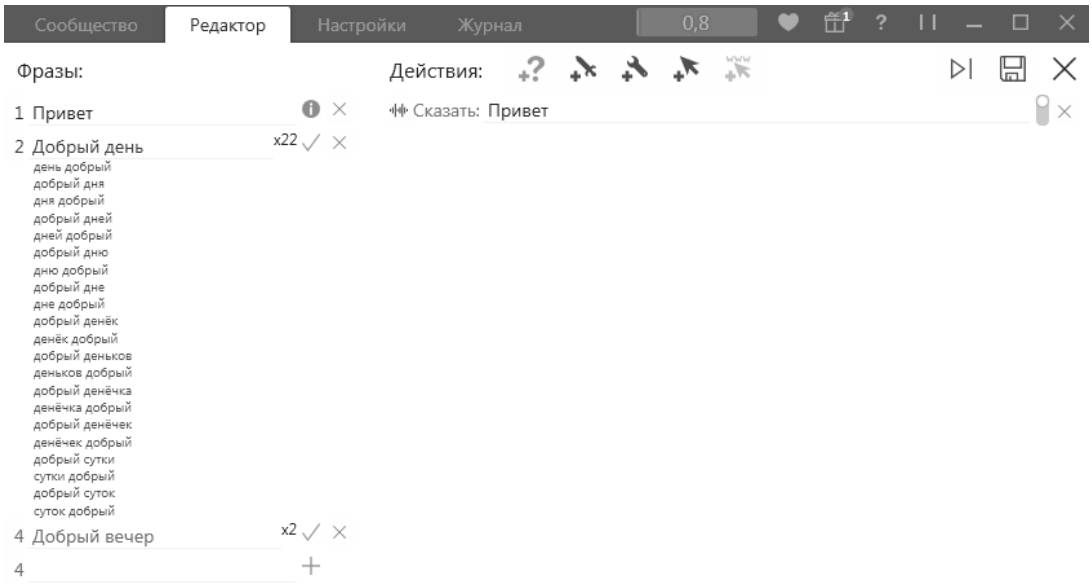

Рисунок 10 – «Добавление команды»

На вкладке «Редактор» можно ввести фразы, на которые помощник будет выполнять выбранные команды.

Вывод: Laitis программа со стильным интерфейсом. Один из её недостатков – загруженность интерфейса. Удобно, что можно добавлять свои команды, редактировать, удалять, но не каждый пользователь захочет в этом разбираться. Laitis не умеет устанавливать напоминания, говорить определения, переводить фразы и конвертировать, и этому её невозможно научить.

Выводы по разделу один

Выполнен сравнительный анализ отечественных и зарубежных аналогов – рассматривалось шесть голосовых помощников. Благодаря проведенному анализу выделены основные функции и решения для реализации голосового помощника.

2 РАЗРАБОТКА ГОЛОСОВОГО ПОМОЩНИКА

2.1 Современные тенденции в построении и дизайне голосового помощника

На сегодняшний день каждый сможет определить, что такое искусственный интеллект.

Искусственный интеллект позволяет программным обеспечениям учиться, адаптироваться к новым входным данным и выполнять задачи. Множество известных примеров искусственного интеллекта основаны на глубоком обучении и понимании естественного языка. С помощью данных методов программы способны выполнять конкретные задачи в определенной сфере, определяя закономерности и шаблонность получаемых данных.

Любая программа имеет свой собственный пользовательский интерфейс. Интерфейс обеспечивает обмен информацией между пользователем и компонентами компьютерной системы. Он играет основную роль в оценке программы пользователем.

21 век является информационным веком. Популярность минимализма продолжает расти. Порядок и простота привлекают людей. Поэтому чем проще и подсознательно понятнее будет интерфейс, тем выше он оценит программу. Элементы должны располагаться в привычных местах для потребителя. Благодаря этому, пользователь будет быстрее ориентироваться в интерфейсе помощника.

2.2 Платформа для разработки голосового ассистента

Голосовой ассистент разработан на основе языка программирования Python в интегрированной среде Pycharm.

Выбран язык программирования Python, так как он является гибким и логичным и имеет библиотеку PyTorch, которая использована для разработки искусственного интеллекта лежащего в основе ассистента.

2.3 Нейронная модель голосового помощника

Голосовой ассистент определяет какие функции выполнять в зависимости от фразы пользователя и имеет возможность дополнять шаблоны фраз и на основе их переобучаться. Для этого построена нейронная модель на основе библиотеки PyTorch.

PyTorch – актуальная библиотека глубокого обучения, разработанная от компании Facebook. С помощью этого модуля можно перестраивать статический граф при изменении набора шаблонов модели. Граф строится динамически каждый раз при прямом проходе для того, чтобы затем иметь возможность сделать проход обратный. Подобный подход позволяет переобучать нейронную сеть, добавляя дополнительные поля для обучения.

В классе нейронной модели определены три слоя. Каждый слой класса «nn.Linear» принимает на вход два параметра: количество входов и количество выходов. Количество входов и выходов вычисляются из словаря обучения.

Словарь является динамическим и хранится в файле «config.json», так как требуется иметь возможность вносить в него изменения. После выполнения каждой команды пользователя ассистент будет узнавать, правильно ли он понял и выполнил команду пользователя, если нет, тогда помощник запишет эту фразу в словарь, перед этим, спросив, что он должен был выполнить.

Словарь шаблонов имеет элементы, приведенные в таблице 2.1

Таблица 2.1 – Описание элементов словаря

№	Название элемента	Назначение элемента	Тип значения
1	tag	Содержит строку, которая обозначает набор элементов	String
2	patterns	Содержит список шаблонов задач, произнесенных пользователем	Array
3	responses	Содержит список ответов пользователю, которые выбираются в случайном порядке ассистентом.	Array

Перед обучением словарь требуется нормализовать. Для этого используются методы: токенизация и лемматизация.

Токенизация предложения – это разделение его на слова с удалением стоп-слов.

Лемматизация – это приведение слова к его нормальной словарной форме.

Выполняются методы с помощью библиотеки «nltk».

NLTK (Natural Language Toolkit) – это платформа для создания программ с обработкой естественного языка для анализа компьютером человеческого языка.

Затем создаётся мешок слов для дальнейшего обучения нейронной сети.

Мешком слов является модель текстов, в которой каждый текст представляет собой неупорядоченный набор слов, не имеющий данные об их связи между собой. Мешок слов можно представить в качестве матрицы. Каждый столбец такой матрицы содержит в себе соответствующее слово, а каждая строка определяет текст. Пересечение строки и столбца содержит частоту слова в данном тексте.

Функция «training()» используется для переобучения нейронной сети на основе измененного словаря. Она получает в виде аргументов массив слов из нормализованных фраз и теги, по которым программа будет ориентироваться, какую задачу просит выполнить пользователь. После этого библиотека «PyTorch» строит граф и вычисляет веса. Результат обучения записывается в файл «data.pth», чтобы в дальнейшем не требовалось переобучать помощника без изменений словаря. Код функции «training()» приведен в листинге А.1.

Изначальное количество эпох обучения стоит 2000.

Через нейронную сеть нельзя разом пропустить весь объем данных. Поэтому эпохи делятся на некоторое число тренировочных объектов. В реализованном голосовом ассистенте таких объектов 10.

Количество входов первого слоя будет равно длине массива слов нормализованных фраз, количество выходов равно 8.

Количество входов и выходов скрытого слоя равно 8.

Количество входов последнего слоя, выходного, равно количеству выходов скрытого слоя, а количество выходов последнего слоя равно количеству тегов нашего словаря.

Затем в каждой эпохе в итерации происходит обучение нейронной сети и сохранение конечного результата обучения. Результатами обучения является вычисленный граф.

2.4 Определение задачи голосовым помощником

Для считывания голоса пользователя с микрофона используется библиотека «Speech Recognition» от компании Google. Первая секунда звука с микрофона используется для его калибровки по уровню шума с помощью метода «abjust_for_ambient_noise()» библиотеки. Затем начинается запись звука, если он отличается от выявленного шума. Далее полученный экземпляр типа «AudioData» и название языка отправляется запросом в Google методом «recognize_google()», который возвращает текстовый вид записанной речи. Код функции «get_voice()», отвечающей за считывание речи, приведен в листинге А.2.

Требуется произвести удаление возможных имен голосового ассистента и второстепенных слов – «озвучь», «скажи», «произнеси» и т.д. Слова, от которых необходимо очистить входящий текст, хранятся в файле «options.json».

Перед определением команды необходимо произвести токенизацию и составление мешка слов из полученной фразы. Переменная передается в обученную нейронную сеть, которая выдаст матрицу вероятностей по тегам из словаря шаблонов.

Перед выполнением каждой функции требуется удалять второстепенные слова, для выделения ключевых слов запроса пользователя.

Голосовой ассистент произносит фразы с помощью библиотеки «pyttsx3». Этот пакет использует родные драйверы речи, когда они доступны, и работает в оффлайн режиме. Для озвучивания фраз была создана функция «speak()». Код функции «speak ()» приведен в листинге А.3.

2.5 Добавление, редактирование и удаление напоминаний

Отсортированные напоминания, начиная от ближайших, хранятся в файле «reminder.json». Во время запуска голосового ассистента запускается функция «check_rem()». Код функции «check_rem()» приведен в листинге А.4. В ней считывается массив из файла с напоминаниями и напоминания проходят проверку на истечение срока. Если срок напоминания истёк, то оно удаляется, иначе запускается таймер в параллельном потоке с помощью пакета threading и его метода «Timer()», получающий на вход название функции и аргументы, которые будут переданы по истечению количества секунд в функцию «voice_rem()». Код функции «voice _rem()» приведен в листинге А.5. В конце функции «check_rem()» файл с напоминаниями перезаписывается.

Файл с напоминаниями имеет элементы, приведенные в таблице 2.2

Таблица 2.2 – Описание элементов файла напоминаний

№	Название элемента	Назначение элемента	Тип значения
1	name	Содержит название напоминания	String
2	time	Содержит дату и время, когда нужно напомнить, в виде строки (%Y-%m-%d %H:%M)	String

После истечения времени таймер запустит функцию «voice_rem()», в которой голосовой ассистент сообщает о сохраненном напоминании и удаляет его из файла, затем снова запускает функцию «check_rem()»

Добавление напоминания происходит в функции «add_rem()».

Если пользователь скажет: «Напомни мне через 2 дня купить лимонад», то голосовой ассистент спросит: «Во сколько вам напомнить?» и после валидного ответа пользователя запишет напоминание в файл.

Учитывается так же, что пользователь может сказать просто «Установи напоминание через 1 час», тогда ассистенту нужно будет только узнать, что именно необходимо напомнить пользователю.

Помощник берёт во внимание такие слова как послезавтра, завтра и месяц, если пользователь его назовёт.

Если пользователь просто скажет: «Поставь напоминание», то голосовой ассистент уточнит, на какое время, какое число и какой месяц, а затем, что сказать в это время.

После определения времени и текста напоминания голосовой ассистент уточнит, все ли верно, если пользователь согласится, тогда напоминание будет записано в файл со всеми напоминаниями, иначе отменит действие.

Так же для того, чтобы голосовой ассистент мог произносить дату и время напоминания, была создана функция «speak_date». Она преобразует числовую дату в текстовую. Синтезатор речи плохо выговаривает окончание «ого», поэтому принято решение данное окончание заменить на «ово».

Функции удаления и изменения напоминаний схожи. Для поиска напоминаний используется функция «search_rem()», которая возвращает массив найденных напоминаний из файла «reminder.json». Пользователь может вызвать методы изменения или удаления напоминания тремя возможными способами.

Пользователь скажет короткую фразу «Василиса, удали напоминание» или «Василиса, измени напоминание». Тогда в функции поиска Василиса попросит сказать день и месяц напоминания или его название. Затем будет определено, что назвал пользователь дату или наименование. И выполнен поиск по записанным в файл напоминаниям.

Пользователь может дополнить одну из коротких вышеописанных фраз наименованием или датой. В случае, если дата является валидной будет произведен поиск в файле.

Возвращенный массив функцией «search_rem()». Код функции «search_rem()» приведен в листинге А.6. Если массив имеет более одного элемента, тогда Василиса продиктует найденные напоминания и попросит пользователя назвать порядковый номер напоминания, которое нужно удалить или изменить. Если массив имеет один элемент, то Василиса продолжит без выбора напоминания.

В функции изменения, после поиска и выбора требуемого напоминания, пользователь может изменить дату, время, число или название. Василиса

удостоверится всё ли верно она поняла, когда пользователь согласится, она удалит или изменит выбранное напоминание.

2.6 Конвертация единиц измерения

Конвертация единиц измерения основывается на заранее созданном словаре, который содержит в себе различные измерения (длина, масса, время, информация, сила, объем, площадь). Так же был добавлен элемент со степенными приставками. В каждом измерении все элементы имеют значение в виде коэффициента отношения к одной определенной единице измерения. Например, в элементе «long» ею является метр.

Элементы словаря единиц измерения приведены в таблице 2.3

Таблица 2.3 – Описание элементов словаря с единицами измерения

№	Название элемента	Назначение элемента	Тип значения
1.	name	Содержит название измерения.	String
2.	numb	Содержит список из перечисленных единиц измерения, которые имеют собственные значения	Array

После инициализации словаря производится поиск числа в полученной строке от пользователя. Если число найдено, оно одно, тогда определяется единица измерения в словаре. В случае, если единица измерения является метром, граммом, битом и т.д. производиться поиск в этом слове приставки для дальнейшей записи степени.

Затем, происходит поиск в тексте уже другой единицы измерения, по такому же принципу.

Если были определены две единицы измерения, и они из одного измерения, тогда, используя полученные данные, программа конвертирует по формуле 1.

$$res = \frac{\left(\frac{n \cdot 10^{rate_1}}{f_1}\right) \cdot f_2}{10^{rate_2}} \quad , \tag{1}$$

где res – искомое количество нужной единицы измерения;

n – данное количество единицы измерения;

$rate_1$ – степень числа 10, в зависимости от приставки единицы измерения;

f_1 – коэффициент отношения данной единицы измерения к определенной;

f_2 – коэффициент отношения нужной единицы измерения к определенной;

$rate_2$ – степень числа 10, в зависимости от приставки нужной единицы измерения.

2.7 Переводчик

Существует пакет «Google Translate API», разработанный компанией Google. С помощью него производится перевод сказанной фразы пользователем. В функции «translate()» создается объект класса «Translator». После этого в качестве параметров методу «translate()» передается текст, язык назначения и исходный. Возвращаемый объект имеет поле «text» со значением переведенного текста.

Поля объекта класса «Translator» приведены в таблице 2.4

Таблица 2.4 – Описание полей объекта класса «Translator»

№	Название элемента	Назначение элемента	Тип значения
1	src	Исходный язык	String
2	dest	Язык назначения	String
3	origin	Строка оригинального текста, который требуется перевести	String
4	text	Строка переведенного текста	String
5	pronunciation	Строка произношения переведенного текста	String

2.8 Поиск определений

Поиск определений на сервере Wikipedia осуществляется библиотекой «Wikipedia», который позволяет голосовому ассистенту отправлять запрос с ключевым словом, а затем получать с сервера массив объектов, в котором есть элемент с определением термина.

Сначала необходимо инициализировать объект класса «Wikipedia». С помощью метода «search()».В качестве двух аргументов передается сам термин и количество возвращаемых результатов. Метод возвращает массив названий статей, упорядоченных по схожести с переданным текстом.

После проверки массива на пустоту методом «summary()», в который передается название найденной статьи и количество возвращаемых предложений из статьи помощник получает резюме статьи.

2.9 Вычисление математических примеров

Для расчета произнесенного пользователем выражения была создана функция «cmath()». В ней производится циклический поиск в полученной строке цифр и знаков. В следствии, составляются массив цифр и массив знаков, с помощью условных операторов происходит: умножение, деление, сложение, вычитание и выделение корня, в зависимости от знака в тексте. Код функции «cmath()» приведен в листинге А.7.

2.10 Поиск в интернете

Поиск в интернете осуществляется функцией «open_search_in()» с помощью модуля «webbrowser». Полученная фраза проходит обработку: удаляются второстепенные слова, пробелы заменяются на «%20». Затем модулем «webbrowser» запускается браузер установленный по умолчанию на компьютере, и открывается вкладка с полученной ссылкой.

2.11 Анализ погоды

Для анализа погоды требуется иметь API ключ на сервисе погоды, чтобы получать ответы с прогнозами по указанным параметрам. В настоящее время известным бесплатным сервисом, который подходит для выполнения поставленной задачи, является OpenWeatherMap.

Функция «Weather()» создана для анализа погоды. Ее аргументом является фраза пользователя, из которой по предлогу «в» находится нужный город. Но для

поиска города на сервисе погоды требуется название города на английском языке, для этого используется вышеописанная функция «translate()». Отправляется «GET» запрос с названием города, чтобы получить его id. Для получения данных с сайта OpenWeatherMap используется метод «json» модуля «requests», так как сервис погоды возвращает данные в json формате. Если город был найден и получен его id, отправляется повторный запрос с аргументами, такие как адрес сайта, id города, единица измерения, язык, на котором будет возвращены данные, и API ключ в json формате. Код функции «Weather()» приведен в листинге А.8.

2.12 Открытие программ

Открытие программ и файлов происходит в функции «launch()». В полученной фразе производится поиск ключевых слов, которые хранятся в файле «path.json» с путями к определенным файлам. Все ключевые слова и пути к файлам пользователь должен добавить вручную, так как ассистент не имеет возможности понять, что от него требует запустить пользователь. Далее программа или файл будет запущены методом «startfile()» встроенного модуля «os», который отвечает за управление системой.

2.13 Секундомер

Для работы секундомера при запуске голосового помощника инициализируются две глобальные переменные. Первая переменная «flag_sec», она имеет булевый тип данных. Если ее значение равно истине, то значит секундомер был уже запущен и во вторую глобальную переменную «tmp_sec» был записан объект класса «datetime» в момент старта.

Функция «run_stopwatch()», если секундомер не был запущен записывает текущую дату и время в глобальную переменную «tmp_sec» и переводит «flag_sec» в значение истина, иначе, Василиса уведомит пользователя, что прошлый секундомер не был остановлен, и спросит нужно ли это сделать. После утвердительного ответа будет запущена функция «stop_stopwatch()».

Функция «stop_stopwatch()» отвечает за остановку секундомера и вычисление сколько всего секунд прошло с момента его запуска. Если их общее количество больше 60, то программа считает сколько это минут. Далее в зависимости от количества секунд и минут подбираются падежи слов «минута» и «секунда», и Василиса сообщает. Код функции «stop_stopwatch()» приведен в листинге А.9.

Выводы по разделу два

Реализован весь основной функционал, определенный в результате сравнительного анализа отечественных и зарубежных голосовых помощников. На языке программирования Python реализован голосовой помощник, основанный на нейронной сети и имеющий возможность переобучаться. Голосовой помощник обладает всеми основными функциями, выделенными в результате проведенного сравнительного анализа отечественных и зарубежных голосовых помощников. Функции программы позволяют пользователю автоматизировать выполнение простых задач.

3 РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ ВНЕДРЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Экономическая эффективность

Разработка каждого программного обеспечения должна сопровождаться экономическим обоснованием соей целесообразности, так как разработка сопровождается с капитальным вложением на такие, как приобретение программного обеспечения, необходимой техники и разработку проекта, и выполнение подготовительных работ.

Экономическая эффективность представляет собой соотношение итогов, которые были получены в результате разработки, и ресурсов, затраченных для достижения результатов. Расчет экономической эффективности позволяет в дальнейшем максимизировать прибыль.

3.2 Затраты, затраченные на разработку

Для расчета экономической сначала необходимо определить вложения и вычислить затраты, которые требуются для разработки голосового ассистента.

Описание единовременных затрат приведен в таблице 3.1

Таблица 3.1 – Описание единовременных затрат

Наименование технического средства и ПО	Тип или модель	Стоимость
Операционная система	Windows 10	-
Ноутбук	IRBIS NB NB550, NB550	14 790
Мышь	DEFENDER Optimum MB-270, проводная	140
Стол компьютерный	Стол письменный Kiwi белый	1 650
Кресло	Компьютерное кресло Бюрократ CH-1201NX, черное	2 800
Итого		19 380

Расчет единовременных затрат:

$$З_е = 14790 + 140 + 1650 + 2800 = 19380 \text{ руб.}$$

Определены расходы по оплате труда сотрудника, который занимается созданием и разработкой голосового помощника и расходы по оплате машинного времени при отладке кода.

Календарный план-график основных этапов создания и разработки ассистента представлен в таблице 3.2

Таблица 3.2 – Календарный план-график основных этапов создания и разработки ассистента

Наименование этапа	Дата начала	Длительность, недель
1. Подготовительный (изучение необходимой литературы, сравнительный анализ готовых решений, составление списка основных требований)	01.02.2021	1
2. Теоретическая разработка (разработка теоретической части и решение поставленных задач)	8.02.2021	2
3. Проектирование и выполнение технического задания на ЭВМ	22.02.2021	3
4. Консультации с руководителем проекта	29.03.2021	1
5. Произведение расчетов на ЭВМ и отчёт в электронном виде	05.04.2021-19.04.2021	2

Затраты на теоретическую часть и работу с литературой рассчитываются по формуле 2

$$З_т = С_р \cdot Т_т, \quad (2)$$

где $С_р$ = 150 руб./час – тарифная ставка Python разработчика;

$Т_т$ – время, затраченное на работу с литературой и теоретический анализ.

$$З_т = 150 \cdot 56 = 8400 \text{ руб.}$$

Затраты на теоретические разработки рассчитываются по формуле 3

$$З_{тр} = С_р \cdot Т_{тр}, \quad (3)$$

где $Т_{тр}$ – время, затраченное на теоретические разработки.

$$З_{тр} = 150 \cdot 112 = 16800 \text{ руб.}$$

Затраты на проектирование и выполнение технического задания на ЭВМ рассчитываются по формуле 4

$$З_п = С_п \cdot Т_п, \quad (4)$$

где $Т_п$ – время, затраченное на проектирование и выполнение технического задания на ЭВМ.

$$З_п = 150 \cdot 168 = 25200 \text{ руб}$$

Затраты на оплату машинного времени рассчитываются по формуле 5

$$З_м = С_м \cdot Т_м, \quad (5)$$

где $С_м$ = 30 руб./час – стоимость одного часа машинного времени;

$Т_м$ =168 + 112 час – время использования машины.

$$З_м = 30 \cdot 280 = 8400 \text{ руб}$$

Затраты на консультацию с руководителем рассчитываются по формуле (6)

$$З_{кр} = С_{кр} \cdot Т_{кр}, \quad (6)$$

где $С_{кр}$ = 200 руб./час – тарифная ставка ведущего Python разработчика;

$Т_{кр}$ – время консультаций с руководителем.

$$З_{кр} = 200 \cdot 56 = 11200 \text{ руб}$$

Сумма полученных затрат рассчитывается по формуле 7

$$З_{сум} = З_{т} + З_{тр} + З_{п} + З_{м} + З_{кр}, \quad (7)$$

$$З_{сум} = 8400 + 16800 + 25200 + 8400 + 11200 = 70000 \text{ руб.}$$

Полные затраты по оплате труда сотрудника, по оплате машинного времени при отладке кода представлены в таблице 3.3

Таблица 3.3 – Полные затраты по оплате труда сотрудника, по оплате машинного времени при отладке кода

Наименование затрат	Обозначение	Сумма	
		в рублях	в %
1. Теоретическая часть и работа с литературой	$З_{т}$	8 400	12
2. Теоретические разработки	$З_{тр}$	16 800	24
3. Проектирование и выполнение технического задания на ЭВМ	$З_{п}$	25 200	36
4. Оплата машинного времени	$З_{м}$	8 400	12
5. Консультация с руководителем	$З_{кр}$	11 200	16
ИТОГО	$З_{сум}$	22 940	100

На основании значений, полученных по формулам, составляется календарный план-график работы над разработкой программы.

Затраты на накладные расходы, связанные с проектированием и отладкой программного обеспечения, в том числе стоимость используемых материалов (бумаги, картриджей к принтерам и т.п.), описаны в таблице 3.4.

Таблица 3.4 – Затраты на накладные расходы

Материалы	Потребность	Стоимость одной ед., руб.	Общая стоимость, руб.	Примечания
Бумага	100 листов	1	100	Печать текстов
Расходные материалы для лазерного принтера	1 шт.	1156	1 156	Для печати необходимых данных
Перезаписываемый диск флеш-накопитель	1 шт.	340	340	Резервирование данных и материалов
Использование сети Internet	36 часов	9,5	342	Поиск информации и литератур.
Итого			2 038	

Оплата за пользование электричеством составляет

$$З_э = 0{,}5 \text{ кВатт} \cdot 280 \text{ час} \cdot 3{,}8 \text{ руб.} = 532 \text{ руб.}$$

Себестоимостью разработки голосового ассистента является сумма всех рассчитанных текущих затрат. Себестоимость вычисляется по формуле 8

$$С = З_е + З_{сум} + З_н + З_э, \quad (8)$$

Текущие затраты описаны в таблице 3.5

Таблица 3.5 – Текущие затраты

Наименование статей затрат	Сумма
1. Единовременные затраты ($З_е$)	19 380
2. Полные затраты ($З_{сум}$)	22 940
3. Накладные расходы ($З_н$)	2 038
4. Расходы на электричество при пользовании техникой ($З_э$)	532
Итого затрат (С)	44 890

Можно сделать вывод, что для создания и разработки голосового ассистента потребуется 44890 руб. В дальнейшем, для поддержания проекта и повышения его конкурентоспособности будут полные затраты, накладные расходы и расходы на электричество. Все затраты на разработку программного средства описаны в таблице 3.6

Таблица 3.6 – Затраты на разработку программного средства

Затраты и ожидаемые доходы от внедрения ИС	Период			
Затраты	1 квартал	2 квартал	3 квартал	4 квартал
1. Единовременные затраты	19 380			
2. Полные затраты:	22 940	22 940	22 940	22 940
3. Накладные расходы	2 038	2 038	2 038	2 038
4. Расходы на электричество	532	532	532	532
5. Расходы на рекламу	30 000	15 000	15 000	15 000
Итого затрат:	74 890	40 510	40 510	40 510

Расчет экономической эффективности проводится по следующим показателям:

а) чистый дисконтированный доход (ЧДД) или интегральный эффект;

б) индекс доходности (ИД);

в) внутренняя ставка доходности (ВСД);

г) срок окупаемости;

Чистый дисконтированный доход – превышение интегральных результатов над интегральными затратами. Определяется как сумма текущих эффектов за весь расчетный период, приведенная к начальному шагу.

Метод дисконтированных денежных потоков предполагает анализ потоков инвестиционного капитала, причем как затратных, так и прибыльных, с учетом их временно-стоимостной оценки.

Выбор данного метода был обусловлен следующими причинами:

- необходимость инфляционной корректировки показателей;
- обеспечение наглядности и простоты расчета срока окупаемости;
- стабильность и предсказуемость денежных потоков;
- равномерность денежных потоков.

Этот метод нашел широкое применение в зарубежной практике в ходе оценки инвестиций в материальные и нематериальные активы, особенно в случаях нестабильной макроэкономической ситуации (инфляция, возможность кризисных ситуаций и т. д.).

Суть этого метода заключается в отнесении предполагаемых денежных потоков (и положительных, и отрицательных) к временным интервалам, начиная с момента начала инвестирования. Как правило, при малых показателях рентабельности продукта сроком анализа этого метода выбирают нормативный «срок жизни» продукта, или срок окупаемости продукта-субститута. В данном случае для установления срока окупаемости достаточно ограничиться сроком в 3 года.

К недостаткам метода можно отнести то, что он не позволяет учесть случайные риски (как внешние, так и внутренние) и известную условность расчета срока окупаемости, связанную с выбором ставки дисконтирования.

Чистый дисконтированный доход определяется как сумма текущих эффектов за весь расчетный период и рассчитывается по формуле 9

$$ЧДД = \sum_{t=1}^{6}(Д_t - P_t) \cdot \frac{1}{(1+a)^t}, \quad (9)$$

где $Д_t$ – результаты, достигаемые на t-ом шаге расчета;

P_t – затраты, осуществляемые на том же шаге;

($Д_t$ – P_t) – эффект достигаемый на t-ом шаге.

$1/(1+ a)^t$ – коэффициент приведения по времени результатов и затрат;

а – норма дисконта, равная приемлемой для инвестора норме дохода на капитал = 0,14.

Предполагаемые доходы от продаж (Д) определяются по формуле 10

$$Д = Ц_{пр} \cdot n, \quad (10)$$

где $Ц_{пр}$ – цена продажи голосового помощника, руб.;

n – объем продаж по периодам в соответствии с исследованиями рынка, шт.

Расходы (Р) включают, кроме текущих затрат, расходы на тиражирование и рекламу. Расходы вычисляются по формуле 11

$$Р = З_{тир} \cdot n + З_р \cdot N, \quad (11)$$

где $З_{тир}$ – затраты на тиражирование;

$З_р$ – затраты на рекламу;

n – количество месяцев в рассматриваемом периоде.

Таблица 3.8 – Технико-экономические показатели работы

Периоды (месяц)	Количество продаж	Показатели		$1/(1+a)^t$	Достигнутый эффект	ЧДД
		Доходы	Расходы			

1 квартал 2021	40	32 000	74 890	0.89	-42 890	-38 172,1

Продолжение таблицы 3.8

Периоды (месяц)	Количество продаж	Показатели		$1/(1+a)^t$	Достигнутый эффект	ЧДД
		Доходы	Расходы			
2 квартал 2021	50	40 000	40 510	0.77	-510	-392,7
3 квартал 2021	65	52 000	40 510	0.67	11 490	7 698,3
4 квартал 2021	80	64 000	40 510	0.59	23 490	13 859,1
1 квартал 2022	100	80 000	40 510	0.46	39 490	18 165,4
2 квартал 2022	120	96 000	40 510	0.52	55 490	28 854,8
Итого:		364 000	277 440	3.37	201 674.77	30 013

Индекс доходности – представляет собой отношение суммы приведенных эффектов к величине капитальных вложений. Индекс доходности вычисляется по формуле 12

$$ИД = \frac{\sum_{t=1}^{6} Д_t \cdot \frac{1}{(1+a)^t}}{\sum_{t=1}^{6} P_t \cdot \frac{1}{(1+a)^t}} \quad (12)$$

Индекс доходности строится из тех же элементов, что и ЧДД. Если ЧДД положителен, то ИД > 1 и наоборот. Индекс доходности равен 1,21

Расчет ЧДД инвестиционного проекта показывает, является ли он эффективным при некоторой заданной норме дисконта.

Срок окупаемости – минимальный временной интервал (от начала осуществления проекта), за пределами которого интегральный эффект становится положительным и в дальнейшем остается неотрицательным. Это период (измеряемый в месяцах, кварталах или годах), начиная с которого первоначальные вложения и другие затраты, связанные с проектом, покрывается суммарными результатами его осуществления.

Сроком окупаемости проекта является 4 квартала при выделенных продажах голосового помощника по цене в 800 рублей.

Выводы по разделу три

После проведения анализа и расчета экономической эффективности создания и разработки голосового ассистента можно сделать следующие выводы:

- ежеквартальные затраты на разработку, повышение конкурентоспособности и рекламу равны 40510 руб.;
- единовременные затраты равны 19380 руб.;
- разработчик ГП покроет все свои затраты через год;

Создание и разработка голосового помощника является актуальным на данный момент времени и целесообразным проектом.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускного квалификационного проекта разработан голосовой помощник.

Выполнен сравнительный анализ отечественных и зарубежных аналогов – рассматривалось шесть голосовых помощников в них входили известные программы, такие как Алиса, Laitis и Microsoft Cortana.

С учетом полученной в ходе исследования информации определен функционал разрабатываемого голосового ассистента. В ходе исследования установлено, что на сегодняшний день отсутствуют решения, соответствующие всем выбранным критериям. Большинство рассмотренных программ не имеют функций, такие как конвертация единиц измерения, установка, редактирования и удаление напоминаний, перевод небольших фраз с русского на английский, выдача определений терминов по запросу пользователя, вычисление математических примеров, анализ погоды.

Принято решение весь функционал голосового помощника сделать доступными всем пользователям. Функции доступные потребителю: «Обучение ГП», «Установка, редактирование и удаление напоминаний», «Конвертация единиц измерения», «Переводчик», «Поиск определений», «Вычисление математических примеров», «Поиск в интернете», «Анализ погоды», «Открытие программ», «Секундомер», «Выбор устройства ввода», «Ввод текста вручную», «Автозапуск ГП».

Благодаря богатому инструментарию языка программирования Python реализован голосовой помощник, основанный на нейронной сети и имеющий возможность переобучаться. Все функции программы позволяют пользователю автоматизировать выполнение простых задач, сократить до минимума затрачиваемое время, тем самым, повысить его производительность и сосредоточить свое внимание на том, что наиболее важно.

Таким образом, цель работы достигнута, задачи – решены.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Python для чайников, Мюллер Д.П., 2019.
2. Изучаем Python. 3-е издание, Марк Лутц 2017. – 830 с.
3. Как устроен Python, Мэтт Харрисон 2019.
4. Python 3. Самое необходимое, Н.А.Прохоренок, В.А.Дронов, 2019.
5. Информационные системы и технологии в экономике и управлении Текст учеб. пособие для вузов по специальности "Прикл. информатика (по областям)" и др. В. В. Трофимов, О. П. Ильина, Е. В. Трофимова и др.; под ред. В. В. Трофимова ; Санкт-Петербург. гос. ун-т экономики и финансов. - 3-е изд.
6. Habr. Электронный ресурс: https://habr.com/ru/
7. Документация по PyTorch. Электронный ресурс: https://pytorch.org/
8. Документация по Python. Электронный ресурс: https://www.python.org/

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Основной код функций голосового помощника

Листинг А.1 – Код функции «training()»

```
def training(X_train, tags):
    dataset = BotDataset()

    # Гипер-параметры
    num_epochs = 2000
    batch_size = 10
    learning_rate = 0.001
    input_size = len(X_train[0])
    hidden_size = 8
    output_size = len(tags)

    train_loader = DataLoader(dataset=dataset,
                              batch_size=batch_size,
                              shuffle=True,
                              num_workers=0)

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    model = NeuralNet(input_size, hidden_size, output_size).to(device)

    # Потери и оптимизация
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
    # Тренировка модели
    for epoch in range(num_epochs):
        for (words, labels) in train_loader:
            words = words.to(device)
            labels = labels.to(dtype=torch.long).to(device)

            # Шаг вперед
            outputs = model(words)
            loss = criterion(outputs, labels)

            # Шаг назад и оптимизация
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        if (epoch + 1) % 100 == 0:
            print(f'Эпоха" [{epoch + 1}/{num_epochs}], Потеря: {loss.item():.4f}')
    print(f'Финальная Потеря: {loss.item():.4f}')
    data = {
        "model_state": model.state_dict(),
        "input_size": input_size,
        "output_size": output_size,
        "hidden_size": hidden_size,
        "all_words": all_words,
        "tags": tags
    }
    FILE = cwd + "/data.pth"
```

```
    torch.save(data, FILE)
    print(f'Тренировка завершена. Результаты сохранены в файл {FILE}')
    options["training"] = "False"
    with open(cwd + "/options.json", "w", encoding="utf-8") as outfile:
        json.dump(options, outfile, indent=4,
                  ensure_ascii=False,
                  separators=(',', ': '))
```

Листинг А.2 – Код функции «get_voice()»

```
def get_voice():
    r = sr.Recognizer()
    with sr.Microphone() as sourse:
        r.adjust_for_ambient_noise(sourse, duration=0.5)
        audio = r.listen(sourse)
    try:
        voice = r.recognize_google(audio, language = "ru-RU").lower()
        print("Распознано: " + voice)
        return voice
    except (sr.UnknownValueError):
        print("Голос не распознан!")
    except sr.RequestError as e:
        print("Неизвестная ошибка!")
```

Листинг А.3 – Код функции «speak()»

```
def speak(text): # Функция для произношения помощника
    tts.say(text)
    tts.runAndWait()
    tts.stop()
```

Листинг А.4 – Код функции «check_rem()»

```
def check_rem():
    now = datetime.now()
    with open(cwd + "/reminder.json", "r",encoding="utf-8") as rf:
        reminder = json.load(rf)
    i=0
    for m in reminder:
        data_time_obj = datetime.strptime(m["time"], '%Y-%m-%d %H:%M')
        delay = (data_time_obj - now).total_seconds()
        if delay<0:
            del reminder[i]
            i=i+1
        else: break
    if len(reminder)>0:
        data_time_obj = datetime.strptime(reminder[0]["time"], '%Y-%m-%d %H:%M')
        delay = (data_time_obj - now).total_seconds()
        if delay-300>0:
            delay=delay-300
        if delay< 400000:
            timer = threading.Timer(delay, voice_rem,args=(re-
minder[0]["name"],data_time_obj))
            timer.start()
    with open(cwd + "/reminder.json", "w",encoding="utf-8") as outfile:
        json.dump(reminder, outfile,indent=4,
                                    ensure_ascii=False,
                                    separators=(',', ': '))
```

Листинг А.5 – Код функции «voice_rem()»

```
def voice_rem(name,time):
    with open(cwd + "/reminder.json", "r",encoding="utf-8") as rf:
        reminder = json.load(rf)
    i=0
    for m in reminder:
        m_time = datetime.strptime(m["time"], '%Y-%m-%d %H:%M')
        if m["name"]==name and m_time==time:
            del reminder[i]
            break
        i=i+1
    with open(cwd + "/reminder.json", "w",encoding="utf-8") as outfile:
        json.dump(reminder, outfile,indent=4,
                                    ensure_ascii=False,
                                    separators=(',', ': '))
    s_time = time.strftime("%H:%M")
    speak("Сегодня в " + s_time + " вам нужно" + name)
    check_rem()
```

Листинг А.6 – Код функции «search_rem()»

```
def search_rem(text):
    arr_month = ["январ", "феврал", "март", "апрел", "май", "мае", "мая", "июн",
"июл", "август", "сентябр",
                 "октябр", "ноябр", "декабр"]
    searching = []
    now = datetime.now()
    year = now.year
    month = now.month
    reminder = {"name": "name", "time": "name"}
    with open(cwd + "/reminder.json", "r", encoding="utf-8") as rf:
        reminder = json.load(rf)
    if len(text) < 5:
        speak("Скажите день и месяц или название напоминания")
        text = input("Вы: ").lower()
        if len(text) <= 10:
            flag = False
            month = 0
            for indx, x in enumerate(arr_month):
                if x in text:
                    if indx > 3 and indx < 7:
                        month = 5
                    else:
                        if indx > 6:
                            month = indx - 1
                        else:
                            month = indx + 1
                    flag = True
                    if month < now.month:
                        year = year + 1
                    break
            if flag == False:
                mon_yea = voice_month()
                month = mon_yea[0]
                year = mon_yea[1]
            match = re.search(r'\d{1,2}', text)
            flag = False
            if match:
                try:
```

```
                    day = int(match[0])
                    if day <= calendar.monthrange(year, month)[1] and day > 0:
                        flag = True
                    else:
                        speak("Не верная дата. В этом месяце " + str(calen-
dar.monthrange(year, month)[1]))
                except(Exception):
                    speak("Не верная дата")
            if flag == False:
                day = voice_day(month,year)
            s_day = str(day)
            s_month = str(month)
            if day < 10:
                s_day = "0" + str(day)
            if month < 10:
                s_month = "0" + str(month)
            time = "-" + str(s_month) + "-" + str(s_day)
            for x in reminder:
                if time in x["time"]:
                    searching.append(x)
        else:
            for x in reminder:
                ratio = similarity(x["name"], text)
                if ratio > 0.9:
                    searching.append(x)
    else:
        flag_m_d = False
        month = 0
        for indx, x in enumerate(arr_month):
            if x in text:
                if indx > 3 and indx < 7:
                    month = 5
                else:
                    if indx > 6:
                        month = indx - 1
                    else:
                        month = indx + 1
                flag_m_d = True
                if month < now.month:
                    year = year + 1
                break
        if flag_m_d:
            day = 0
            match = re.search(r'\d{1,2}', text)
            flag = True
            if match:
                try:
                    day = int(match[0])
                    if day < 0 or day > calendar.monthrange(year, month)[1]:
                        speak("Не верное число. В этом месяце " + str(calen-
dar.monthrange(year, month)[1]) + "дней")
                        flag = False
                except(Exception):
                    flag = False
                    speak("Не верная дата")
            if flag == False:
                day = voice_day(month,year)
            s_day = str(day)
            s_month = str(month)
            if day < 10:
                s_day = "0" + str(day)
```

```
            if month < 10:
                s_month = "0" + str(month)
            time = str(s_month) + "-" + str(s_day)
            for x in reminder:
                if time in x["time"]:
                    searching.append(x)
        else:
            for x in reminder:
                ratio = similarity(x["name"], text)
                if ratio > 0.9:
                    searching.append(x)
    return searching
```

Листинг А.7 – Код функции «cmath()»

```
def cmath(text):#функция вычисления
    for x in options['cmath_tbr']:
        text = text.replace(x, "").strip()
    text = text.replace(",", ".").strip()
    str = text.split()
    if len(str)<3 and len(str)%2 == 0:
        Exception
    tr={'0','1','2','3','4','5','6','7','8','9'}
    tr2 = {'x', '+', '-', '/'}
    arg = []
    sign = []

    try:
        i=0
        for x in str:
            flag = False
            for y in tr:
                result = re.search(y, x)
                if result != None:
                    flag = True
                    result = None
                    break
            if flag:
                arg.append(float(x))
                str.pop(i)
            i=i+1
        n = 0
        i=1
        if str[0]=='корень':
            str.pop(0)
            arg[0]=sqrt(arg[0])
        max = len(str)-1
        for idx, x in enumerate(str):
            if x == '-':
                if idx+1 <=max:
                    if str[idx+1] == 'корень':
                        arg[i] = sqrt(arg[i])
                arg[0] = arg[0] - arg[i]
            if x == '+':
                if idx + 1 <= max:
                    if str[idx + 1] == 'корень':
                        arg[i] = sqrt(arg[i])
                arg[0] = arg[0] + arg[i]
            if x == '/':
                if idx + 1 <= max:
```

```
                    if str[idx + 1] == 'корень':
                        arg[i] = sqrt(arg[i])
                arg[0] = arg[0] / arg[i]
            if x == 'x' or x == 'x' or x == '*':
                if idx + 1 <= max:
                    if str[idx + 1] == 'корень':
                        arg[i] = sqrt(arg[i])
                arg[0] = arg[0] * arg[i]
            i = i + 1
        speak("Готово! Ваше число %.0f" % arg[0])
        return True
    except (Exception):
        speak("Не верное выражение")
        return False
```

Листинг А.8 – Код функции «weather()»

```
def weather(place):#функция погоды
    for x in options['weather_tbr']:
        place = place.replace(x, "").strip()
    tmp = place.split()
    place = ""
    for id,x in enumerate(tmp):
        if x.lower()=="в" and id+1<len(tmp):
            place = tmp[id+1]
    if place != "":
        city = translate(place)
        s_city = city + ",RU"
        city_id = 0
        appid = "8e2269b81b7c3791ee7925767f560491"
        try:
            res = requests.get("http://api.openweathermap.org/data/2.5/find",
                               params={'q': s_city, 'type': 'like', 'units': 'met-
ric', 'APPID': appid})
            data = res.json()
            cities = ["{} ({})".format(d['name'], d['sys']['country'])
                      for d in data['list']]
            city_id = data['list'][0]['id']
        except Exception as e:
            speak("Ошибка. Город не найден")
            return False
        try:
            res = requests.get("http://api.openweathermap.org/data/2.5/weather",
                               params={'id': city_id, 'units': 'metric', 'lang':
'ru', 'APPID': appid})
            data = res.json()
            descr = data['weather'][0]['description']
            temp = data['main']['temp']
            s = "Сейчас в " + tmp + " " + descr
            speak(s + ". Температура - %.0f градусов" % temp)
            if temp <= -25:
                speak("Ужасные морозы. Не забудьте надеть шкуру мамонта! Ха-ха-ха-
ха")
            else:
                if temp <= 0:
                    speak("Оденьтесь потеплее. На улице зима всё же!")
                else:
                    if temp <= 10:
                        speak("На улице прохладненько. Наденьте лучше куртку")
                    else:
```

```
                        if temp < 20:
                            speak("На улице более менее тепло")
                        else:
                            if temp >= 20:
                                speak ("На улице жарища! Пора купаться!")
            return True
        except Exception as e:
            speak("Ошибка. Город не найден")
            return False
    else: speak("Я вас не поняла")
```

Листинг А.9 – Код функции «stop_stopwatch()»

```
def stop_stopwatch():
    global flag_sec
    global tmp_sec
    if flag_sec:
        time_sec = int((datetime.now() - tmp_sec).total_seconds())
        time_min = 0
        str_sec = ""
        str_min = ""
        str_and = ""
        while time_sec >= 60:
            time_sec = time_sec - 60
            time_min = time_min + 1
        if time_sec > 0:
            if time_sec%10 > 4:
                str_sec = str("{:.0f}".format(time_sec)) + " секунд"
            else:
                if time_sec%10 > 1:
                    str_sec = str("{:.0f}".format(time_sec)) + " секунды"
                else:
                    if time_sec % 10 > 0:
                        str_sec = str("{:.0f}".format(time_sec)) + " секунд"
        if time_min>0:
            if time_min%10 > 4:
                str_min = str("{:.0f}".format(time_min)) + " минут"
            else:
                if time_min%10 > 1:
                    str_min = str("{:.0f}".format(time_min)) + " минуты"
                else:
                    if time_min%10 > 0:
                        str_min = str("{:.0f}".format(time_min)) + " минута"
        if str_min != "" and str_sec != "":
            str_and = " и "
        speak("Секундомер остановлен. Прошло " + str_min + str_and + str_sec)
        flag_sec = False
        return True
    else:
        speak("Секундомер не был запущен. Запустить секундомер?")
        res = y_n()
        if res == "yes":
            run_stopwatch()
            return True
        else:
            speak("Хорошо")
            return True
```