МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

**(ФГБОУ ВО «ВГУ»)**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

*Разработка и реализация Telegram бота для перевода текста AutoTranslatorRUS*

*Курсовая работа*

09.03.04 *Программная инженерия*

*Профиль «Информационные системы и технологии»*

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *С.Д. Махортов, д.ф.- м.н., доцент* \_\_.\_\_.20\_\_

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.В. Черных, 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.А.Вахтин, к.ф.-м.н, доцент*

**Содержание**

[Введение 3](#_Toc168876021)

[1 Анализ предметной области. 5](#_Toc168876022)

[1.1 Глоссарий. 5](#_Toc168876023)

[1.2 Постановка задачи. 6](#_Toc168876024)

[1.3 Предметная область проекта. 7](#_Toc168876025)

[1.4 Обзор аналогов. 9](#_Toc168876026)

[1.4.1 Multitran Bot (@multitran\_bot). 9](#_Toc168876027)

[1.4.2 Translation Chatbot(@TranslationChatBot). 10](#_Toc168876028)

[2 Проектирование приложения. 12](#_Toc168876029)

[2.1 Требования к приложению. 12](#_Toc168876030)

[2.2 Диаграмма вариантов использования. 13](#_Toc168876031)

[2.3 Интерфейс бота. 15](#_Toc168876032)

[3 Реализация приложения. 18](#_Toc168876033)

[3.1 Анализ существующих решений для реализации на рынке. 18](#_Toc168876034)

[3.2 Средства реализации приложения. 21](#_Toc168876035)

[3.3 Диаграмма последовательности. 23](#_Toc168876036)

[3.4 Диаграмма классов. 25](#_Toc168876037)

[3.5 Реализация работы Microsoft Translator API. 26](#_Toc168876038)

[3.6 Реализация работы NTextCat. 27](#_Toc168876039)

[3.7 Реализация работы Telegram.Bot. 27](#_Toc168876040)

[4 Тестирование приложения. 29](#_Toc168876041)

[4.1 Перевод с польского языка. 29](#_Toc168876042)

[4.2 Перевод с немецкого языка. 29](#_Toc168876043)

[4.3 Перевод с английского языка. 30](#_Toc168876044)

[Заключение 31](#_Toc168876045)

[Список используемых источников 33](#_Toc168876046)

[Приложение 34](#_Toc168876047)

**Введение**

В современном мире люди все чаще взаимодействуют с информацией через мессенджеры и социальные сети. Коммуникации на разных языках становятся неотъемлемой частью жизни людей, создавая потребность в быстрых и максимально точных текстов. Текущие решения, такие как отдельные приложения и веб-сервисы для перевода, зачастую требуют переключения между платформами, что может быть очень неудобно и тратит много времени.

Автоматический перевод текстов, особенно в мессенджерах, представляет собой важный инструмент для обеспечения коммуникации без языковых барьеров. Он может использоваться в различных сферах деятельности: бизнесе для переговоров с международными партнерами, образовании для перевода учебных материалов, туризме для общения с местными жителями и т.д. Приложения для автоматического перевода позволяют не только экономить время, но и повышают точность переводов, снижая риск недоразумений.

Целью данной работы является разработка телеграм-бота для автоматического перевода текстов с автоматическим распознаванием предоставленного языка и последующим его переводом на русский язык. Для достижения этой цели необходимо решить следующие задачи:

* Провести обзор существующих решений и определить их недостатки;
* Спроектировать архитектуру и интерфейс приложения;
* Реализовать функцию автоматического распознавания языка и перевода текста с использованием Microsoft Translator API;
* Протестировать приложение для обеспечения его стабильной работы;
* Провести тестирование приложения с текстами на различных языках.

Работа состоит из четырех основных разделов: введение, проектирование приложения, реализация приложения и тестирование приложения, а также заключение и приложения. В разделе "Проектирование приложения" описан процесс создания концепции и архитектуры бота, включающий определение основных функций, взаимодействие с пользователем и разработку интерфейса. Раздел "Реализация приложения" посвящен техническим аспектам разработки и интеграции необходимых библиотек и API. В разделе "Тестирование приложения" приведены результаты тестирования и анализа работоспособности бота. Заключение содержит выводы о проделанной работе и возможных направлениях для дальнейшего улучшения приложения.

1. **Анализ предметной области.**
   1. **Глоссарий.**

**Git** – распределенная система контроля версий, используемая для управления исходным кодом и координации работы в команде;

**Microsoft Translator API** – сервис, предоставляемый Microsoft для перевода текста на различные языки. Используется в данном проекте для выполнения перевода текста, отправленного пользователями;

**NTextCat** – библиотека для автоматического определения языка текста. В проекте используется для определения языка текста перед переводом;

**Telegram API –** интерфейс для взаимодействия с мессенджером Telegram, позволяющий создавать ботов, которые могут принимать и отправлять сообщения, обрабатывать команды и взаимодействовать с пользователями;

**TelegramBotClient –** библиотека для взаимодействия с Telegram API, используемая для обработки сообщений, отправленных пользователями, и отправки ответов;

**.NET Core –** кроссплатформенная платформа для разработки приложений, используемая для создания сервера и логики бота на языке C#;

**Bot –** автоматизированная программа, выполняющая предопределенные задачи. В данном проекте бот обрабатывает текстовые сообщения и выполняет их перевод;

**API –** набор инструментов и протоколов для создания приложений. В контексте курсовой работы используется Microsoft Translator API для перевода текста и Telegram API для взаимодействия с пользователями через мессенджер;

**Visual Studio –** комплексная интегрированная среда разработки (IDE), которую можно использовать для написания, редактирования, отладки и сборки кода, а затем для развертывания приложения.

* 1. **Постановка задачи.**

Целью данной курсовой работы является разработка и выпуск автономного бота для платформы Telegram, который способен автоматически определять язык текста, переводить его на русский язык и отправлять пользователю результат. Данный бот использует API сервиса Microsoft Translator для выполнения перевода.

К разрабатываемому приложению выдвинуты следующие требования:

* приложение должно уметь автоматически определять язык введенного пользователем текста;
* поддержка распознавания текста на нескольких языках, включая английский, испанский, польский, немецкий и другие;
* приложение должно использовать API Microsoft Translator для перевода текста на русский язык;
* высокая точность перевода, даже для сложных и специфических терминов;
* приложение должно быть способно получать текстовые сообщения от пользователей Telegram;
* приложение должно отправлять переведенный текст обратно пользователю в чате Telegram;
* приложение должно уведомлять пользователя в случае ошибки перевода или других проблем;
* приложение должно быть совместимо с последними версиями Telegram API;
* приложение должно быть легко разворачиваемым и настраиваемым;
* приложение должно обеспечивать быстрый и надежный перевод текста, минимизируя время ожидания пользователя;
* приложение должно защищать данные пользователя и соблюдаться конфиденциальность переписки;
* использование безопасных методов хранения и передачи API ключей и других конфиденциальных данных.

Для достижения данных целей требуется провести исследование в области обработки естественного языка (NLP), интеграции с внешними API сервисами, а также разработки и тестирования ботов для платформы Telegram.

* 1. **Предметная область проекта.**

Проект разработки бота для платформы Telegram, который автоматически определяет язык текста, переводит его на русский язык и отправляет пользователю результат, затрагивает несколько ключевых областей информационных технологий и искусственного интеллекта. Данный бот предоставляет пользователям возможность легко и быстро получать переводы текста, используя современные технологии обработки естественного языка (NLP) и взаимодействия с внешними API.

Обработка естественного языка (NLP) представляет собой область информатики, фокусирующуюся на взаимодействии между компьютерами и человеческим языком. Основная задача NLP - научить машины понимать, интерпретировать и генерировать человеческий язык таким образом, чтобы он был полезен и значим для пользователей. В контексте данного проекта, технология NLP используется для автоматического определения языка текста, что является первым шагом перед его переводом.

Основные задачи NLP в проекте включают: Определение языка текста: Используя библиотеку NTextCat, бот определяет язык входящего сообщения. Эта библиотека использует заранее обученные модели, основанные на анализе частотности использования различных символов и слов в тексте. Сегментация текста: Разделение текста на отдельные предложения или слова для дальнейшей обработки. Извлечение и анализ лексических единиц: Определение частей речи, синтаксический анализ и семантическая интерпретация текста.

Интеграция с внешними API является важной частью проекта, поскольку позволяет использовать мощные облачные сервисы для выполнения сложных задач, таких как перевод текста. Microsoft Translator API используется в данном проекте для обеспечения качественного и быстрого перевода.

Процесс интеграции с API включает следующие этапы: Настройка и авторизация: Использование ключей API для доступа к сервисам Microsoft Translator. В проекте используются API-ключ и хост, предоставленные Microsoft, для аутентификации запросов. Формирование запросов: Создание и отправка запросов на сервер Microsoft Translator с указанием исходного текста и целевого языка перевода. Обработка ответов: Обработка полученного от сервера ответа и извлечение переведенного текста для отправки пользователю.

Telegram предоставляет мощный API для разработки ботов, что делает его отличной платформой для реализации проекта. Основные аспекты разработки включают: Обработка сообщений: Использование методов библиотеки Telegram.Bot для приема и обработки текстовых сообщений от пользователей. Ответы на запросы: Формирование ответных сообщений с переведенным текстом и отправка их пользователю через Telegram API. Обработка ошибок: Обработка возможных ошибок, таких как проблемы с подключением к API или некорректные входные данные, и уведомление пользователя о возникших проблемах.

Разработка такого бота имеет значительное прикладное значение. В современном мире потребность в быстром и точном переводе текста возрастает. Пользователи могут столкнуться с текстами на различных языках в повседневной жизни, в профессиональной деятельности или в учебе. Возможность мгновенно получить перевод с любого языка на русский значительно упрощает взаимодействие с информацией и повышает удобство использования.

Проект также демонстрирует возможности интеграции различных технологий для создания комплексных и полезных приложений. Комбинация NLP для определения языка, использование мощных облачных API для перевода и удобство взаимодействия через Telegram делает этот проект ярким примером современного подхода к разработке программного обеспечения.

Проект разработки бота для автоматического определения и перевода текста охватывает несколько важных аспектов информационных технологий, включая обработку естественного языка, интеграцию с внешними API и разработку ботов для мессенджеров. Это позволяет создавать удобные и функциональные инструменты, которые находят широкое применение в различных сферах жизни. Такой бот не только облегчает повседневные задачи пользователей, но и служит примером успешной интеграции современных технологий для решения практических задач.

* 1. **Обзор аналогов.**
     1. **Multitran Bot (@multitran\_bot).**

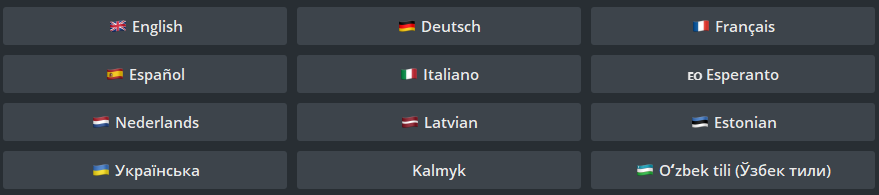


Рисунок 1 – языки Multitran Bot.

**Multitran Bot** использует онлайн-словарь Multitran для перевода слов и фраз. Основное преимущество этого бота в том, что он хорошо подходит для перевода отдельных слов и коротких фраз. Однако бот имеет следующие особенности и ограничения:

* мультиязычность: Multitran Bot поддерживает перевод на несколько языков (Рисунок 1);
* ограничение объема текста: Поддерживает только короткие фразы и отдельные слова, что ограничивает его применение для более длинных текстов;
* качество перевода: Перевод основывается на словаре Multitran, что может не всегда быть точным для сложных текстов.

Преимущества AutoTranslatorRUS\_bot:

* широкая поддержка языков: Бот поддерживает перевод всех текстов на множество языков, доступных через Microsoft Translator API;
* поддержка длинных текстов: Возможность перевода длинных текстов и предложений без ограничений.
  + 1. **Translation Chatbot(@TranslationChatBot).**

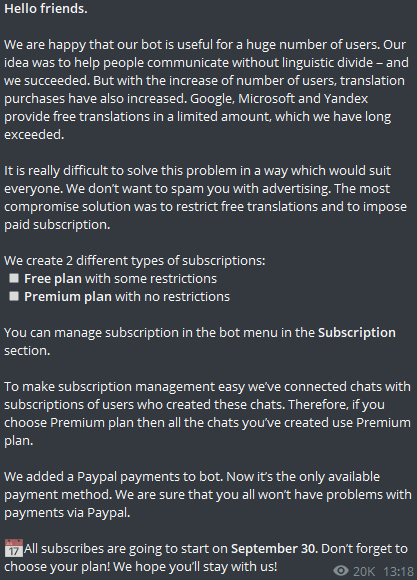


Рисунок 2 – Подписка Translation Chatbot.

Translation Chatbot переводит все сообщения в чате на выбранные языки. Этот бот поддерживает 50 языков и может работать как в личных чатах, так и в группах.

* мультиязычность: Translation Chatbot поддерживает множество языков, что делает его удобным для пользователей из разных стран;
* ограничение объема текста: Поддерживает перевод любых объемов текста в чате, что делает его универсальным инструментом. Однако объем сильно ограничен без платной подписки (рисунок 2);
* качество перевода: Среднее, подходит для общего общения, но может быть недостаточным для специализированных текстов.

Преимущества AutoTranslatorRUS\_bot:

* более широкая языковая поддержка: Бот поддерживает больше языков благодаря интеграции с Microsoft Translator API;
* улучшенное качество перевода: Более высокая точность перевода с использованием современных алгоритмов NLP;
* Цена: Бесплатен и не имеет платных подписок.

1. **Проектирование приложения.**
   1. **Требования к приложению.**

Функциональные требования - это вид требований, описывающий поведение системы. В этом виде требований должен отражаться предоставляемый пользователю функционал.

Функциональные требования:

* приложение должно автоматически определять язык текста, введенного пользователем с использованием библиотеки NTextCat для определения языка текста;
* приложение должно переводить текст на выбранный язык пользователя, используя Microsoft Translator API для обеспечения точного перевода, с поддержкой множества языков, включая английский, испанский, китайский и другие;
* приложение должно принимать текстовые сообщения от пользователей через Telegram, интегрируясь с Telegram API для получения и отправки сообщений, обрабатывая текстовые сообщения от пользователей и отправляя переведенный текст обратно.
* приложение должно уведомлять пользователя в случае ошибки перевода или других проблем, реализуя систему уведомлений для информирования пользователей о статусе перевода и отправляя сообщения об ошибках в случае проблем с переводом или соединением;
* приложение должно сохранять историю переводов для дальнейшего использования пользователями, предоставляя возможность сохранения истории переводов для доступа к предыдущим переводам и функцию экспорта истории переводов в удобный формат;
* приложение должно быть интуитивно понятным и простым в использовании, обеспечивая дружелюбный и интуитивно понятный интерфейс для удобства пользователей и простые и понятные команды для взаимодействия с ботом;
* приложение должно быть надежным и безопасным, защищая данные пользователя и обеспечивая конфиденциальность переписки, используя безопасные методы хранения и передачи API ключей и других конфиденциальных данных;
* приложение должно поддерживать одновременную работу с большим количеством пользователей, обеспечивая масштабируемость для обработки большого объема запросов без ухудшения производительности и поддержку многопользовательского режима для одновременной работы с несколькими пользователями.

Нефункциональные требования:

* приложение должно работать на всех платформах, таких как Windows, Linux и Mac OS, iOS и Android;
* приложение должно быть производительным, обеспечивая высокую скорость обработки запросов для минимального времени ожидания пользователей и оптимизацию кода для обеспечения быстрой работы приложения;
* приложение должно быть адаптируемым, позволяя добавление новых языков и функций без необходимости значительных изменений в коде и обеспечивая гибкость в настройках для удовлетворения потребностей различных пользователей;
* приложение должно быть масштабируемым, поддерживая увеличение нагрузки без снижения производительности и возможность работы в облачных средах для обеспечения высокой доступности и отказоустойчивости.
  1. **Диаграмма вариантов использования.**

В разработанном приложение отсутствует виды пользователей. Пользователь может использовать функционал только при наличии профиля Telegram.

Диаграмма представлена на рисунке 3.

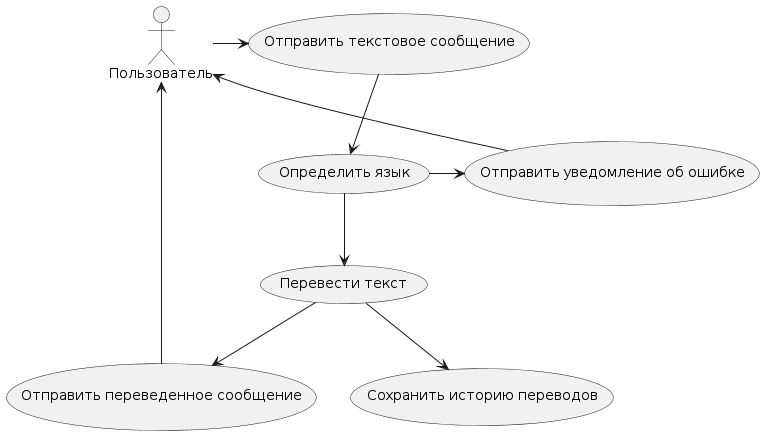


Рисунок 3 – Диаграмма вариантов использования.

Описание вариантов использования:

* пользователь отправляет текстовое сообщение боту в Telegram. Это является отправной точкой для всех дальнейших процессов;
* бот автоматически определяет язык текста, используя библиотеку NTextCat. Этот процесс необходим для правильного выбора языка перевода;
* бот использует Microsoft Translator API для перевода текста на выбранный язык. Это основной процесс, который обеспечивает предоставление услуги перевода пользователю;
* после перевода текста бот отправляет переведенное сообщение обратно пользователю в Telegram. Пользователь получает результат своей заявки;
* переведенный текст сохраняется в истории переводов для дальнейшего использования пользователями. Пользователь может просматривать и использовать предыдущие переводы;
* В случае возникновения ошибки бот отправляет пользователю уведомление об ошибке. Это обеспечивает пользователя информацией о статусе его запроса и позволяет принять необходимые действия в случае проблем.
  1. **Интерфейс бота.**

Разработка интерфейса бота “AutoTranslatorRUS\_bot”.

Чат с ботом представлен на рисунке 4.

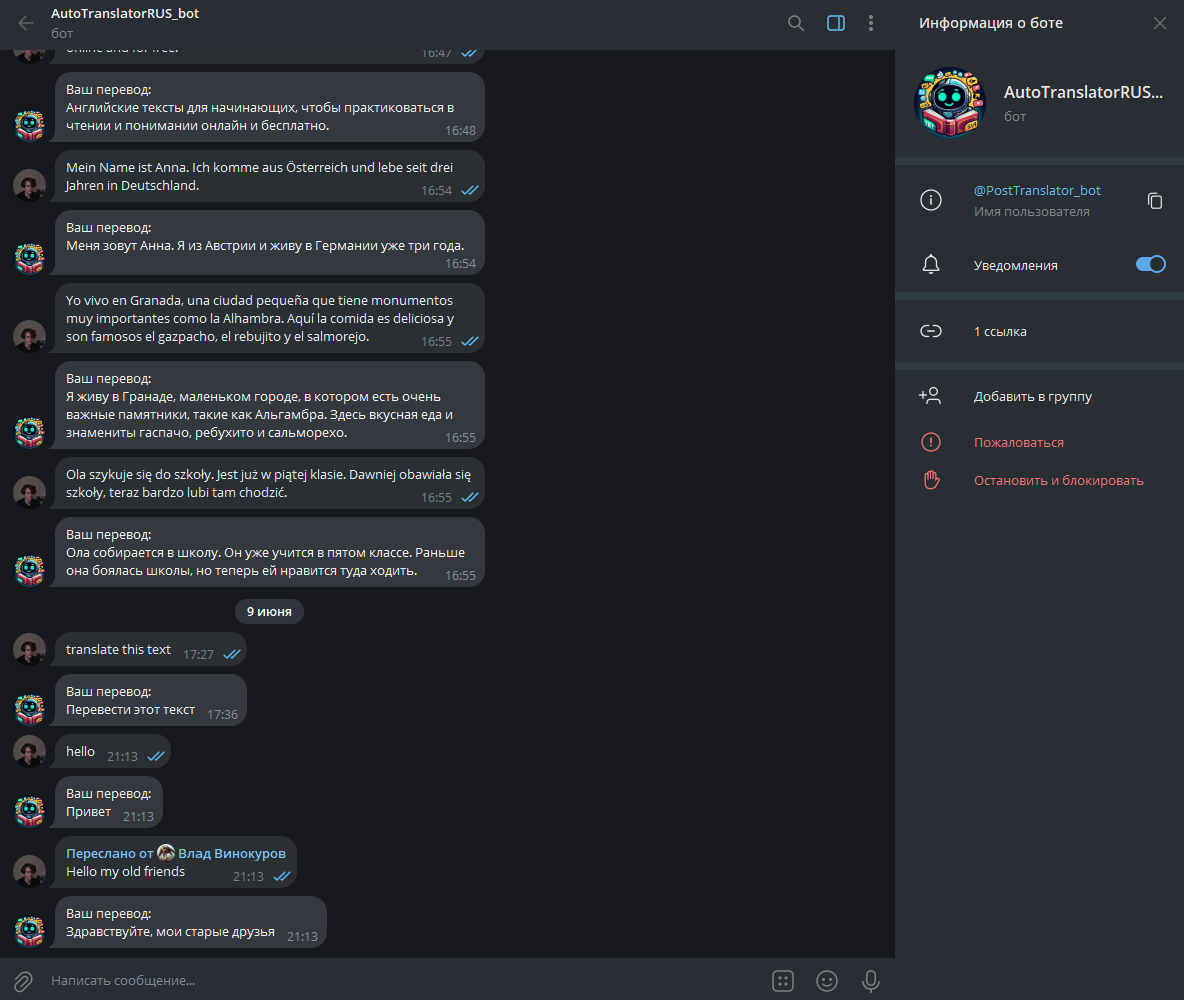


Рисунок 4 – Главная страница бота.

Форма <<главная страница>> - экран приложения, отвечающий главным функционалом приложения. Здесь пользователь будет взаимодействовать с ботом, отправлять текстовые сообщения для перевода и получать ответы. Бот AutoTranslatorRUS\_bot отображает оригинальные сообщения и их переводы, что обеспечивает удобство использования и прозрачность процесса перевода.

На рисунке 5 представлено то, как выглядит бот в списке чатов.



Рисунок 5 – Бот в списке чатов.

Форма <<бот в списке чатов>> показывает, как бот выглядит в списке чатов в Telegram.

На рисунке 6 представлено то, как выглядит переписка с ботом.



Рисунок 6 – Запросы и ответы от бота.

Форма <<запросы и ответы от бота>> демонстрирует процесс отправки сообщений для перевода и получения переводов от бота. Пользователь отправляет текстовое сообщение, бот определяет язык, переводит текст и отправляет переведенное сообщение обратно пользователю. Также показаны случаи, когда бот уведомляет пользователя об ошибках в процессе перевода.

На рисунке 7 представлен перевод пересланного сообщения от другого пользователя.

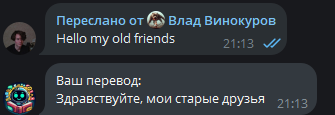


Рисунок 7 – Перевод пересланного сообщения.

Форма <<перевод пересланного сообщения>> отображает функциональность перевода пересланных сообщений. Пользователь может переслать сообщение из другого чата боту для перевода. Бот переводит пересланное сообщение и отправляет перевод обратно пользователю. Это удобно для перевода сообщений, полученных от других пользователей.

1. **Реализация приложения.**
   1. **Анализ существующих решений для реализации на рынке.**

На рынке существует множество решений для автоматического перевода текста, которые могут быть использованы для создания Telegram бота. В этом разделе будут рассмотрены основные преимущества и недостатки существующих решений, которые могут повлиять на выбор технологии для реализации бота.

Google Translate API является одним из наиболее популярных и широко используемых сервисов для перевода текста. Он поддерживает более 100 языков и обеспечивает высокую точность перевода.

Некоторые из основных достоинств Google Translate API:

* высокая точность перевода: Google Translate API способен обеспечивать высокую точность перевода, особенно для распространенных языков и фраз;
* широкий спектр поддерживаемых языков: Поддержка более 100 языков делает этот API универсальным решением для пользователей со всего мира;
* интеграция с другими сервисами Google: Возможность интеграции с другими сервисами Google, такими как Google Docs и Google Sheets, что делает его удобным для различных бизнес-приложений.

Некоторые из возможных недостатков Google Translate API:

* стоимость: Использование Google Translate API платное, что может стать значительным расходом при большом объеме переводов;
* ограничения на использование: Существуют ограничения на количество запросов в сутки, что может ограничить использование сервиса для крупных проектов.

Microsoft Translator API является еще одним популярным сервисом для перевода текста, предоставляемым компанией Microsoft. Этот сервис поддерживает множество языков и интегрируется с различными платформами Microsoft.

Некоторые из основных достоинств Microsoft Translator API:

* высокая точность перевода: Microsoft Translator API обеспечивает высокую точность перевода, особенно для делового и технического контента;
* интеграция с продуктами Microsoft: Возможность интеграции с продуктами Microsoft, такими как Azure, Office 365 и другими корпоративными сервисами;
* поддержка множества языков: Поддержка более 60 языков делает этот сервис подходящим для международных проектов.

Некоторые из возможных недостатков Microsoft Translator API:

* стоимость: Использование Microsoft Translator API также платное, и расходы могут увеличиваться с ростом объема переводов;
* требования к интеграции: Интеграция с некоторыми платформами может потребовать дополнительных усилий и знаний в области разработки.

Yandex.Translate API является популярным решением в России и странах СНГ. Этот сервис поддерживает множество языков и предоставляет бесплатный доступ с ограничениями на количество запросов.

Некоторые из основных достоинств Yandex.Translate API:

* бесплатный доступ: Возможность бесплатного использования с ограничениями на количество запросов, что делает его доступным для небольших проектов и тестирования;
* поддержка распространенных языков: Поддержка основных языков, используемых в России и странах СНГ;
* высокая скорость перевода: Быстрая обработка запросов и высокая производительность.

Некоторые из возможных недостатков Yandex.Translate API:

* ограничения на количество запросов: Бесплатный доступ ограничен количеством запросов, что может быть недостаточным для крупных проектов;
* ограниченная точность перевода: Точность перевода может быть ниже по сравнению с Google Translate или Microsoft Translator для некоторых языков и специализированных терминов.

IBM Watson Language Translator является одним из передовых решений на рынке, предлагающим широкий спектр возможностей для перевода текста и анализа языка.

Некоторые из основных достоинств IBM Watson Language Translator:

* интеграция с Watson: Возможность интеграции с другими сервисами IBM Watson для создания комплексных решений;
* поддержка различных форматов: Поддержка множества языков и возможность работы с различными форматами данных;
* анализ контекста: Способность анализировать контекст и улучшать точность перевода с учетом специфики текста.

Некоторые из возможных недостатков IBM Watson Language Translator:

* стоимость: Высокая стоимость использования, что может быть значительным барьером для небольших компаний и проектов;
* сложность интеграции: Интеграция с другими сервисами может требовать дополнительных усилий и технических знаний.

DeepL Translator — это один из новых и быстро набирающих популярность сервисов для перевода текста, известный своей высокой точностью и естественностью переводов.

Некоторые из основных достоинств DeepL Translator:

* высокая точность и естественность перевода: DeepL Translator обеспечивает переводы, которые звучат более естественно по сравнению с другими сервисами;
* интуитивно понятный интерфейс: Простой и удобный интерфейс для использования как в веб-версии, так и в интеграциях;
* поддержка популярных языков: DeepL поддерживает основные европейские языки и расширяет список поддерживаемых языков.

Некоторые из возможных недостатков DeepL Translator:

* ограниченная поддержка языков: В настоящее время DeepL поддерживает меньшее количество языков по сравнению с Google Translate или Microsoft Translator;
* стоимость: Коммерческое использование DeepL Translator платное, что может увеличить затраты при большом объеме переводов.

Анализируя существующие решения, можно сделать вывод, что каждое из них имеет свои сильные и слабые стороны. Для разработки Telegram бота, обеспечивающего высокое качество перевода и удобство использования, рекомендуется использовать Microsoft Translator API, который предоставляет оптимальное соотношение точности перевода и возможностей интеграции.

* 1. **Средства реализации приложения.**

Для реализации данного приложения были выбраны следующие средства:

* операционная система Windows 11;
* язык программирования C#: Используется для создания логики бота и его взаимодействия с API Telegram и Microsoft Translator;
* Telegram.Bot библиотека: Обеспечивает интеграцию с API Telegram, позволяя принимать и отправлять сообщения пользователям.
* NTextCat библиотека: Используется для определения языка текста, введенного пользователем;
* Microsoft Translator API: Используется для перевода текста на различные языки, обеспечивая высокую точность перевода;
* .NET Core SDK: Набор инструментов для разработки и запуска приложений на платформе .NET;
* Visual Studio: Среда разработки, используемая для написания, тестирования и отладки кода приложения;
* Rider: Среда разработки, используемая для написания, тестирования и отладки кода приложения;
* Git: Система контроля версий для управления исходным кодом и совместной работы над проектом.

* 1. **Диаграмма последовательности.**

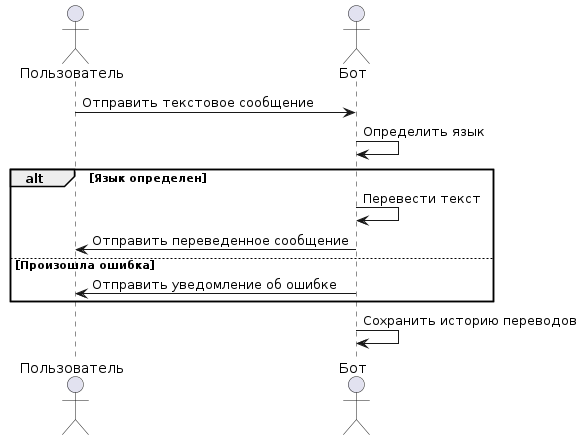


Рисунок 8 – Схема диаграммы последовательности.

Диаграмма последовательности иллюстрирует процесс взаимодействия пользователя с ботом.

Пользователь отправляет текстовое сообщение:

* gользователь инициирует процесс, отправляя текстовое сообщение боту в Telegram. Это сообщение содержит текст, который пользователь хочет перевести;

Бот определяет язык:

* бот получает сообщение и определяет язык текста с использованием библиотеки NTextCat. Это необходимо для правильного выбора языка перевода.

Язык определен:

* если язык текста успешно определен, бот переходит к следующему шагу перевода текста.

Бот переводит текст:

* бот использует Microsoft Translator API для перевода текста на выбранный язык. Этот процесс включает в себя отправку текста на сервер и получение переведенного текста.

Бот отправляет переведенное сообщение:

* после успешного перевода бот отправляет переведенное сообщение пользователю в чат Telegram. Пользователь получает текст на целевом языке.

Произошла ошибка:

* если на каком-либо этапе процесса произошла ошибка (например, не удалось определить язык или выполнить перевод), бот отправляет пользователю уведомление об ошибке. Это информирует пользователя о проблеме и позволяет предпринять необходимые действия.

Бот сохраняет историю переводов:

* переведенный текст сохраняется в чате с ботом. Это позволяет пользователю просматривать и использовать предыдущие переводы в будущем.
  1. **Диаграмма классов.**

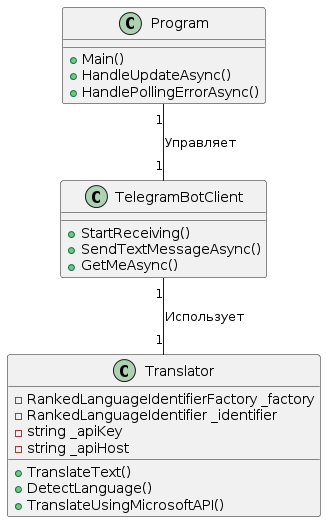


Рисунок 9 – Диаграмма классов.

Диаграмма классов для Telegram бота, выполняющего перевод текста, представлена на рисунке 9 и включает в себя три основных класса: Program, TelegramBotClient и Translator. Каждый из этих классов имеет свои методы и поля, необходимые для выполнения задач бота.

Класс Program:

Методы:

* Main(): Главный метод, запускающий приложение;
* HandleUpdateAsync(): Метод для обработки входящих обновлений от Telegram;
* HandlePollingErrorAsync(): Метод для обработки ошибок, возникающих при опросе сервера Telegram.

Связи:

* Управляет объектом класса TelegramBotClient.

Класс TelegramBotClient:

Методы:

* StartReceiving(): Метод для запуска процесса получения сообщений от Telegram;
* SendTextMessageAsync(): Метод для отправки текстового сообщения пользователю;
* GetMeAsync(): Метод для получения информации о боте.

Связи:

* Использует объект класса Translator.

Класс Translator:

Поля:

* RankedLanguageIdentifierFactory \_factory: Фабрика для создания идентификаторов языка;
* RankedLanguageIdentifier \_identifier: Идентификатор языка;
* string \_apiKey: API ключ для доступа к Microsoft Translator API;
* string \_apiHost: Хост API для Microsoft Translator.

Методы:

* TranslateText(): Метод для перевода текста;
* DetectLanguage(): Метод для определения языка текста;
* TranslateUsingMicrosoftAPI(): Метод для перевода текста с использованием Microsoft Translator API.
  1. **Реализация работы Microsoft Translator API.**

Для реализации работы с Microsoft Translator API в нашем Telegram боте используются ключевые переменные и функции, обеспечивающие взаимодействие с этим сервисом.

Основные этапы реализации:

* инициализация API ключа и хоста: Необходимо инициализировать переменные для хранения API ключа и хоста, которые используются для аутентификации запросов;
* метод перевода текста: Основной метод, отвечающий за отправку текста на сервер Microsoft Translator и получение переведенного текста, использует HTTP-запросы для взаимодействия с API;
* обработка перевода: Метод обработки текста TranslateText вызывает основной метод перевода и возвращает результат перевода пользователю.
  1. **Реализация работы NTextCat.**

NTextCat используется для автоматического определения языка текста, что является важным этапом перед переводом.

Основные этапы реализации:

* инициализация NTextCat: Включает инициализацию необходимых фабрик и идентификаторов для обработки текстов;
* метод определения языка: Основной метод DetectLanguage анализирует входящий текст и определяет его язык на основе вероятностных моделей, предоставляемых библиотекой NTextCat;
* интеграция с ботом: Метод определения языка вызывается при получении нового сообщения от пользователя, обеспечивая правильное направление текста на перевод.
  1. **Реализация работы Telegram.Bot.**

Библиотека Telegram.Bot используется для взаимодействия с API Telegram, обеспечивая возможность приема и отправки сообщений.

Основные этапы реализации:

* инициализация клиента TelegramBotClient: Инициализация клиента включает в себя установку токена бота и начало процесса приема сообщений;
* обработка обновлений: Метод HandleUpdateAsync отвечает за обработку входящих обновлений, таких как новые сообщения, команды и другие виды взаимодействий от пользователей;
* отправка сообщений: Метод SendTextMessageAsync используется для отправки сообщений пользователям, включая переведенные тексты и уведомления об ошибках;
* обработка ошибок: Метод HandlePollingErrorAsync обрабатывает ошибки, возникающие при взаимодействии с API Telegram, обеспечивая стабильность работы бота.

1. **Тестирование приложения.**

В процессе разработки и реализации Telegram бота для перевода текста было проведено тестирование его функциональности. Тестирование включало проверку перевода с различных языков, обработку пересланных сообщений и оценку точности переводов.

* 1. **Перевод с польского языка.**

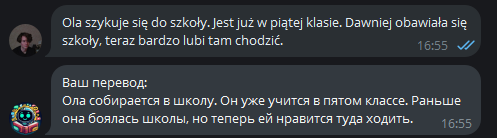


Рисунок 10 – Перевод с польского.

На скриншоте 10 представлен пример перевода с польского языка. Пользователь отправил текст на польском языке:

«Ola szykuje się do szkoły. Jest już w piątej klasie. Dawniej obawiała się szkoły, teraz bardzo lubi tam chodzić»

Бот успешно определил язык и перевел текст на русский язык:

«Ола собирается в школу. Он уже учится в пятом классе. Раньше она боялась школы, но теперь ей нравится туда ходить.»

Перевод точный и соответствует смыслу исходного текста.

* 1. **Перевод с немецкого языка.**

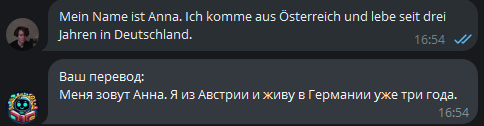


Рисунок 11 – Перевод с немецкого.

На рисунке 11 представлен пример перевода с немецкого языка. Пользователь отправил текст на немецком языке:

«Mein Name ist Anna. Ich komme aus Österreich und lebe seit drei Jahren in Deutschland. »

Бот успешно определил язык и перевел текст на русский язык:

«Меня зовут Анна. Я из Австрии и живу в Германии уже три года. »

Перевод выполнен корректно, и смысл оригинального текста полностью сохранен.

* 1. **Перевод с английского языка.**

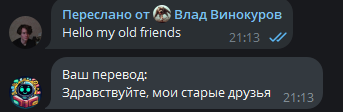


Рисунок 12 – Перевод с английского.

На третьем скриншоте представлен пример перевода пересланного сообщения на английском языке. Пользователь переслал сообщение:

«*Hello my old friends*»

Бот распознал пересланное сообщение и перевел его на русский язык:

«*Здравствуйте, мои старые друзья*»

Перевод выполнен корректно и точно передает смысл исходного сообщения.

**Заключение**

В данной курсовой работе был разработан и реализован Telegram бот для автоматического перевода текстов. Целью проекта было создание автономного приложения, способного определять язык текста и переводить его на различные языки с использованием Microsoft Translator API. В ходе работы были рассмотрены и проанализированы существующие решения на рынке, такие как Google Translate API, Yandex.Translate API и IBM Watson Language Translator, что позволило выбрать оптимальное решение для реализации проекта.

В процессе разработки были использованы следующие средства и технологии:

* **язык программирования C#** и платформа .NET Core для разработки логики приложения;
* **библиотека Telegram.Bot** для интеграции с API Telegram и обработки сообщений пользователей;
* **Microsoft Translator API** для выполнения переводов текста;
* **NTextCat** для автоматического определения языка текста;
* **инструменты Visual Studio** и Rider для разработки, тестирования и развертывания приложения.

В работе были подробно рассмотрены и описаны основные функции и методы классов, участвующих в работе бота:

* **Program: Управляет процессом обработки обновлений и ошибок**;
* **TelegramBotClient: Обрабатывает входящие сообщения и отправляет ответы пользователям**;
* **Translator: Определяет язык текста и выполняет перевод с использованием Microsoft Translator API.**

Тестирование бота показало, что он успешно справляется с определением языка и переводом текстов с различных языков, включая польский, немецкий и английский. Были приведены примеры работы бота с различными типами сообщений, что подтвердило его функциональность и точность переводов.

Таким образом, разработанный Telegram бот демонстрирует высокую производительность и удобство использования, обеспечивая пользователям качественный и быстрый перевод текстов. Проект показал, что использование современных API и библиотек позволяет создавать эффективные и полезные инструменты для автоматизации и улучшения коммуникации в современных мессенджерах.

**Список используемых источников**

1. Microsoft Translator API Documentation [сайт] – URL:

<https://learn.microsoft.com/en-us/azure/ai-services/translator/>

(дата обращения 14.04.24)

1. Telegram.Bot Library Documentation [сайт] – URL: <https://github.com/TelegramBots/telegram.bot>

(дата обращения 14.04.24)

1. Telegram.Bot Library Documentation [сайт] – URL: <https://github.com/ivanakcheurov/ntextcat>

(дата обращения 14.04.24)

1. .NET Core Documentation [сайт] – URL:

<https://learn.microsoft.com/en-us/dotnet/fundamentals/>

(дата обращения 14.04.24)

**Приложение**

using System;

using System.IO;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Telegram.Bot;

using Telegram.Bot.Extensions.Polling;

using Telegram.Bot.Types;

using Telegram.Bot.Types.Enums;

using NTextCat;

namespace AutoTranslatorBot

{

class Program

{

public static async Task Main(string[] args)

{

var host = CreateHostBuilder(args).Build();

await RunBotAsync(host.Services);

}

static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureAppConfiguration((context, config) =>

{

config.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true);

})

.ConfigureServices((context, services) =>

{

services.AddHttpClient();

services.AddSingleton<Translator>();

services.AddSingleton<ITelegramBotClient>(new TelegramBotClient(context.Configuration["TelegramBotToken"]));

});

static async Task RunBotAsync(IServiceProvider services)

{

var botClient = services.GetRequiredService<ITelegramBotClient>();

var translator = services.GetRequiredService<Translator>();

using var cts = new CancellationTokenSource();

botClient.StartReceiving(

HandleUpdateAsync,

HandlePollingErrorAsync,

new ReceiverOptions { AllowedUpdates = { } },

cts.Token

);

var me = await botClient.GetMeAsync();

Console.WriteLine($"Start listening for @{me.Username}");

Console.ReadLine();

cts.Cancel();

}

private static async Task HandleUpdateAsync(ITelegramBotClient botClient, Update update, CancellationToken cancellationToken)

{

if (update.Type == UpdateType.Message && update.Message.Type == MessageType.Text)

{

var chatId = update.Message.Chat.Id;

var messageText = update.Message.Text;

var detectedLanguage = await translator.DetectLanguageAsync(messageText);

var translatedText = await translator.TranslateTextAsync(messageText, detectedLanguage, "ru");

await botClient.SendTextMessageAsync(chatId, $"Ваш перевод: {translatedText}", cancellationToken: cancellationToken);

}

}

private static Task HandlePollingErrorAsync(ITelegramBotClient botClient, Exception exception, CancellationToken cancellationToken)

{

Console.WriteLine(exception);

return Task.CompletedTask;

}

}

public class Translator

{

private readonly IHttpClientFactory \_httpClientFactory;

private readonly string \_apiKey;

private readonly string \_apiHost;

private readonly RankedLanguageIdentifierFactory \_factory;

private readonly RankedLanguageIdentifier \_identifier;

public Translator(IConfiguration configuration, IHttpClientFactory httpClientFactory)

{

\_httpClientFactory = httpClientFactory;

\_apiKey = configuration["TranslatorApiKey"];

\_apiHost = "https://api.cognitive.microsofttranslator.com";

\_factory = new RankedLanguageIdentifierFactory();

\_identifier = \_factory.Load("Core14.profile.xml");

}

public async Task<string> DetectLanguageAsync(string text)

{

var languages = \_identifier.Identify(text);

var mostCertainLanguage = languages.FirstOrDefault();

return mostCertainLanguage != null ? mostCertainLanguage.Item1.Iso639\_3 : "en";

}

public async Task<string> TranslateTextAsync(string text, string fromLanguage, string toLanguage)

{

var route = $"/translate?api-version=3.0&from={fromLanguage}&to={toLanguage}";

var uri = \_apiHost + route;

var requestBody = new[] { new { Text = text } };

var content = new StringContent(JsonConvert.SerializeObject(requestBody), Encoding.UTF8, "application/json");

using var client = \_httpClientFactory.CreateClient();

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", \_apiKey);

client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Region", "YOUR\_REGION");

var response = await client.PostAsync(uri, content);

var jsonResponse = await response.Content.ReadAsStringAsync();

var translationResult = JsonConvert.DeserializeObject<List<TranslationResult>>(jsonResponse);

return translationResult?.FirstOrDefault()?.Translations?.FirstOrDefault()?.Text;

}

}

public class TranslationResult

{

public List<Translation> Translations { get; set; }

}

public class Translation

{

public string Text { get; set; }

public string To { get; set; }

}

}