НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Автоматика та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Автоматизована система перекладу тексту»

Виконав:
студент IV курсу, групи ІА-з61
Милко Денис Леонідович _____

Керівник:
Старший викладач кафедри АУТС
Яланецький В.А. _____

Рецензент:
Професор кафедри ТК, д.т.н.,
Жураковський Б.Ю. _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Милку Денису Леонідовичу

1. Тема проєкту «Автоматизована система перекладу тексту», керівник проєкту Яланецький Валерій Анатолійович, затверджені наказом по університету від «12» 05 2020 р. №1089-с.

2. Термін подання студентом проєкту 11.06.2020

3. Вихідні дані до проєкту

Java 8, настільний застосунок, миттєвий переклад за «гарячою» клавішою

4. Зміст пояснювальної записки

Аналіз існуючих рішень, аналіз вимог до програмного забезпечення ,

опис програмної реалізації

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Діаграма класів, діаграма використання, діаграма компонентів, діаграма послідовностей

6. Дата видачі завдання _19.12.2019_____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення предметної області	29.12.2019	
2	Порівняльний аналіз існуючих рішень	21.01.2020	
3	Аналіз вимог до програмного забезпечення	14.02.2020	
4	Розробка архітектури програмного забезпечення	27.02.2020	
5	Розробка програмного забезпечення	10.04.2020	
6	Розробка інтерфейсу користувача	16.04.2020	
7	Оформлення пояснювальної записки	15.05.2020	
8	Виконання графічних документів	21.05.2020	
9	Подання ДП на попередній захист	25.05.2020	
10	Подання ДП рецензенту	10.06.2020	
11	Подання ДП на основний захист	11.06.2020	

Студент Денис МИЛКО

Керівник Валерій ЯЛАНЕЦЬКИЙ

АНОТАЦІЯ

Милко Д.Л. Автоматизована система перекладу тексту. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 87 сторінок тексту, 13 рисунків, 1 таблиця, посилання на 32 літературних джерел, додатки та 4 конструкторські документи.

Ключові слова: ISO 639, машинний переклад, розпізнавання мовлення, синтез мовлення, google neural machine translation.

Об’єктом розробки є настільний застосунок з використанням сервісу Google Translate для автоматизації перекладу тексту.

Мета розробки – спрощення отримання перекладу та миттєвий переклад виділеного тексту за «гарячою» клавішею з будь-якого місця – блокноту, документа, книги, чату.

У дипломному проєкті розроблено фрагменти системи автоматизованого перекладу, а саме: переклад тексту з однієї мови на іншу, розпізнання мовлення через мікрофон, синтез мовлення – озвучення тексту.

У першому розділі було описано загальні відомості про Google translate, огляд предметної області та аналіз рішень існуючих застосунків, їх актуальність на сьогоднішній час. У другому розділі було проведено аналіз сценаріїв використання та наведено перелік вимог до програмного забезпечення. У третьому розділі було проведено аналіз засобів розробки та обґрунтування їх вибору – середовище розробки, мова програмування, використані бібліотеки. У четвертому розділі було виконано проєктування архітектури — шаблон та принципи проєктування, архітектура клієнт-сервер. У п’ятому розділі було проведено детальний опис програмної реалізації та моделювання класів.

Значну увагу приділено точному розпізнанню мовлення та виклику перекладу за «гарячою» клавішею.

SUMMARY

Mylko DL Automation of text translation. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 87 pages of text, 13 figures, 1 table, references to 32 literature sources, appendices and 4 design documents.

Keywords: ISO 639, machine translation, speech recognition, speech synthesis, google neural machine translation.

The object of development is a desktop application using the Google Translate service to automate text translation.

The purpose of development is to simplify obtaining a translation and instant translation of selected text by hot key from anywhere — notebook, document, book, chat.

The graduation project developed fragments of the system of automated translation, namely: translation of the text from one language to another, speech recognition through a microphone, speech synthesis — sounding of the text.

The first section described general information about Google translate, an overview of the subject area and analysis of solutions for existing applications, their relevance to date. The second section analyzes usage scenarios and lists software requirements. In the third section, an analysis of the means of development and justification of their choice — the development environment, programming language, use of the library. In the fourth section, the architecture was designed — the template and principles of design, client-server architecture. The fifth section provides a detailed description of software implementation and modeling of classes.

Considerable attention is paid to accurate speech recognition and hot key translation.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА361.020БАК.005 ПЗ	Пояснювальна записка	87		
6	А3	ІА361.020БАК.005 Д1	Автоматизована система	1		
7			перекладу тексту. Діаграма			
8			використання			
9	А3	ІА361.020БАК.005 Д2	Автоматизована система	1		
10			перекладу тексту. Діаграма			
11			класів			
12	А3	ІА361.020БАК.005 Д3	Автоматизована система	1		
13			перекладу тексту. Діаграма			
14			компонентів			
15	А3	ІА361.020БАК.005 Д4	Автоматизована система	1		
16			перекладу тексту. Діаграма			
17			послідовностей процесу			
18			перекладу тексту			
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА361.020БАК.005 ТП		
Розроб.		Милко			Автоматизована система перекладу тексту Відомість технічного проєкту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Яланецький				Т	
						КПІ ім. Ігоря Сікорського, гр. ІА-з61	
Н. Контр.							
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматизована система
перекладу тексту»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

GUI	–	Graphical User Interface
HTTP	–	Hyper Text Transfer Protocol
IDE	–	Integrated development environment
JSON	–	JavaScript Object Notation
JVM	–	Java Virtual Machine
MVC	–	Model View Controller
SOLID	–	Single responsibility, open-closed, Liskov substitution,
	–	interface segregation, dependency inversion
XML	–	eXtensible Markup Language
ДКП	–	Довга короткочасна пам'ять
ЗНМ	–	Згорткова нейронна мережа
МУ	–	Механізм уваги
НМП	–	Нейронний машинний переклад
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
РНМ	–	Рекурентна нейронна мережа
СМП	–	Статистичний машинний переклад

ВСТУП

Актуальність та предмет дослідження. Сьогодні обчислювальна техніка допомагає полегшити життя людини в багатьох сферах його діяльності — оптимізує роботу, знижує фізичні та інтелектуальні витрати. Автоматизований переклад з однієї на іншу мову не став винятком.

Процес перекладу став легким та швидким, так як зникла необхідність наявності та використання паперових словників, довідників по синтаксису та граматиці.

Більш досконалим автоматизованим перекладом є машинний переклад, який максимально зменшує участь людини в перекладі знаходячи потрібні слова та правопис. Послуги машинного перекладу надзвичайно економічно вигідні, дозволяють перекласти набагато більше вмісту за фіксованим бюджетом, ніж це було б можливо, використовуючи лише людських перекладачів. Служба машинного перекладу набагато швидша, ніж людські перекладачі, робить переклад тексту на різні мови швидшим, ніж будь-коли раніше. Машинний переклад є надзвичайно цінним, коли документ потребує перекладу, а швидкість є суттєвою. Швидкість машинного перекладу дозволяє зрозуміти документ на іншій мові за лічені хвилини.

Машинний переклад дозволяє перекладати набагато більше контенту, ніж будь-коли раніше — насправді набагато більше інформації, ніж коли-небудь можна було б перекласти лише людськими перекладачами. Сервіси машинного перекладу можуть навчатись із уже перекладеними текстами та з виправленими публікаціями, виправленнями від людських перекладачів, тобто сервіс покращується з часом. Чим більше користувачі використовують механізми перекладу, тим точнішими будуть їхні переклади, це означає, що з часом ваші витрати зменшаться, оскільки генеруємі переклади будуть кращими та якіснішими, потребуватимуть меншого редагування зі сторони користувача. Це може зробити машинний переклад ідеальною довгостроковою інвестицією для ваших майбутніх потреб.

Найпопулярнішим серед сервісів машинного перекладу є Google translate. Цей сервіс дозволяє безкоштовно, швидко та надійно перекладати тексти великих розмірів. Але весь переклад Google translate здійснює виключно в браузері, і для перекладу текстів потрібно копіювати необхідну для перекладу інформацію на сайт сервісу. Це змушує виконувати зайві рухи, втрачати фокус на важливих задачах, відкривати браузер та перемикати актуальні закладки або створювати нові, що витрачає час.

Актуальність теми підтверджується тим, що проект був розроблений на замовлення сторонньої організації (додаток Г).

Мета і завдання. Спрощення отримання перекладу та організація миттєвого перекладу тексту за гарячою клавішею з будь-якого місця – блокноту, документа, книги, чату. Крім цього потрібно реалізувати основний функціонал, яким володіє Google translate – набір всіх доступних мов сервісу, реалізація перекладу з однієї мови на іншу, розпізнання мовлення через мікрофон, синтез мовлення – прослуховування тексту, тобто перетворення звуку у аудіо формат.

Проблема спрощення отримання перекладу вирішується створенням сервісу в середині ОС, що реагує на виклик за гарячою клавішею та виконує миттєвий переклад виділеного тексту з будь-якого місця, де тільки можливе виділення тексту.

Структура роботи. Дипломний проєкт складається з наступних розділів: вступ, вивчення предметної області, аналіз вимог до ПЗ, засоби розробки, проєктування архітектури ПЗ, опис програмної реалізації, список використаних джерел, додатки – таблиця списку кодів стану HTTP, таблиця мовних кодів ISO 639—1, програмний код класу Controller, копія довідки про впровадження результатів дипломного проєкту в діяльність військової частини.

1 ВИВЧЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний огляд Google Translate

Google Translate — веб-служба компанії Google, що призначена для машинного перекладу тексту на іншу мову [2].

Google використовує самонавчальний алгоритм машинного перекладу — у березні 2017 року компанія повністю перевела механізм перекладу на нейромережі для більш якісного перекладу. Сервіс самостійно пристосовується до нової лексиці — оновлюється і розвивається паралельно з мовою. Тобто, якщо з'являється нове слово або змінюється його лексичне значення, то система розпізнає його і оновить свою базу даних. І, як наслідок, прискорюється «навчання» системи і вдосконалюється якість генерації тексту. На даний момент відомо, що сервер Google Translate володіє більш ніж трильйоном слів.

Основними типами сучасного машинного текстового перекладу є СМП, переклад «за правилами» і НМП. Перекладач Google Translate певний час використовував СМП — заснований на пошуку відповідників мови між перекладним текстом і гігантським масивом сервісу, який складається зі слів, що вносяться користувачем раніше під час їх перекладу. Недолік СМП — цій системі необхідно високопродуктивне апаратне забезпечення. Для вдосконалення програми потрібна величезна кількість обчислень. Також для перекладу СМП характерно те, що якість генерації тексту повністю залежить від кількості даних в корпусі сервісу.

Через характеристики НМП та його перевагу над СМП, нещодавно галузь почала впроваджувати і НМП: у вересні 2016 року команда Google Brain опублікувала блог, де показано, що вони почали використовувати НМП для заміни машинного перекладу на основі фрази (різновид СМП) для китайсько-англійського перекладу Google Translate.

1.1.1 Нейронні мережі Google Voice

Google Voice реалізовано за допомогою тривалої нейромережі з короткою тривалістю пам'яті (ДКП РНМ – рисунок 1.1) [5].

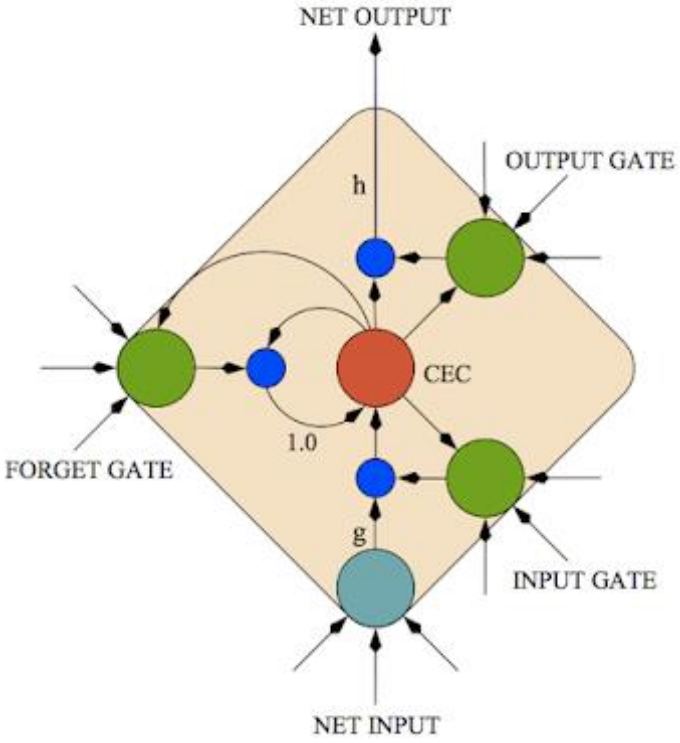

Рисунок 1.1 — Комірка пам'яті ДКП, яка показує механізми решітки, які дозволяють зберігати і передавати інформацію [32]

З моменту запуску в 2009 році сервіс Google Voice використовував акустичні моделі Гауссової суміші розподілу, які до цього вже використовувалась у розпізнаванні мовлення протягом більше ніж 30 років. Складні методи, такі як адаптація моделей до голосу мовця, доповнили цей порівняно простий метод моделювання. Потім приблизно в 2012 році глибокі нейронні мережі здійснили революцію в галузі розпізнавання мовлення. Ці багатошарові мережі відрізняють звуки краще, ніж Гауссова суміш розподілу, використовуючи РМ, диференціюючи фонетичні одиниці, а не моделюючи кожну окремо.

Але все реально швидко покращилося з РНМ, і особливо з ДКП РНМ, вперше запущеними з розпізнавачем мовлення в Android в травні 2012 року. У порівнянні з глибокими нейронними мережами, ДКП РНМ мають додаткові періодичні з'єднання та комірки пам'яті, що дозволяють їм «запам'ятовувати» попередні отримані дані, які поступають впродовж часу, поки користувач продовжує мовлення.

1.1.2 Нейронний машинний переклад

Нейронний машинний переклад — це підтип машинного перекладу, який застосовує велику штучну нейронну мережу для прогнозування ймовірності послідовності слів, часто у вигляді цілих речень [3, 4, 6, 7]. На відміну від статистичного машинного перекладу, який вимагає більше пам’яті та часу, НМП навчає свою базу даних для досягнення максимальної продуктивності. Система не тільки застосовує великий набір даних для тренування своїх алгоритмів, її кінцева конструкція дозволяє системі навчатися з часом і створювати кращі, більш природні переклади.

Як працює НМП? НМП використовує векторні уявлення для слів. Це означає, що слова транслюються у вектор, визначений унікальною величиною та напрямком. Замість окремих компонентів, як мовна модель та модель перекладу, НМП використовує модель єдиної послідовності, яка генерує по одному слову за ітерацію.

НМП використовує двонаправлену періодичну нейронну мережу (рисунок 1.2) , яку також називають кодером для розділення вихідного речення на вектори для другої періодичної нейронної мережі, що називається декодером для прогнозування слів на перекладаємій мові.

Ця модель з’явилась у 2013 році як нова структура кодера-декодера для кінцевого перекладу. Ця модель кодує заданий вихідний текст у безперервний вектор, використовуючи згорнуту нейронну мережу ЗНМ, а

потім використовуючи РНМ в якості декодера для перетворення вектору стану в перекладаєму мову.

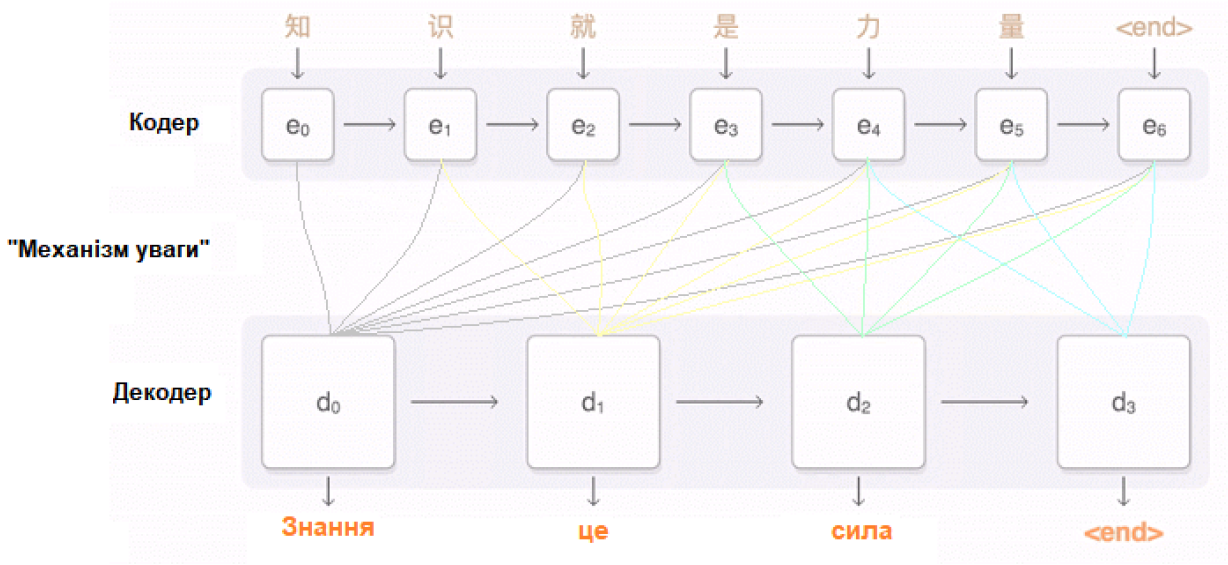

Рисунок 1.2 — Механізм архітектури «кодер-декодерних мереж» архітектури Google НМП [31]

Нелінійне відображення НМП відрізняється від лінійних СМП-моделей і описує семантичну еквівалентність за допомогою векторів стану, які з'єднують кодер і декодер. Крім того, РНМ, як передбачається, здатний фіксувати інформацію за нескінченною тривалістю речень та вирішувати проблему «переупорядкування на великі відстані». Однак проблема зникнення градієнта заважає РНМ впоратися із залежностями на великі відстані; відповідно модель НМП на початку не досягла хороших показників. Для вирішення проблеми залежності на великі відстані та зникнення градієнта було введено ДКП — механізм дозволяє явно видаляти пам'ять та оновлювати ДКП, проблема зникнення градієнта контролюється, щоб модель могла значно краще фіксувати залежність на великі відстані у реченні. Але виникла проблема вектора фіксованої довжини — нейронній мережі необхідно стиснути вихідне речення у вектор фіксованої довжини, що призведе до збільшення складності та невизначеності під час розшифровки, особливо коли сгенероване речення довге.

Проблема вектора фіксованої довжини почала вирішуватися з того часу, коли був введений МУ до НМП у 2014 році. Продуктивність НМП значно покращилася з того часу, і «уважні кодер-декодер мережі» стали найсучаснішою моделлю в галузі НМП.

З іншого боку, все ще існують проблеми та виклики НМТ, які потрібно вирішити: процес навчання та розшифровки досить повільний; стиль перекладу може бути невідповідним для одного і того ж слова; існує проблема «поза словникового запасу» за результатами перекладу; механізм нейронної мережі «чорний ящик» призводить до поганої інтерпретації.

1.1.3 Нейронний синтез мовлення

Нейронна мережа Tacotron 2 від Google синтезує мовлення безпосередньо з тексту [8, 30]. Вона функціонує на основі поєднання ЗНМ і РНМ.

Tacotron 2 використовує просту структуру декодера-кодера, яка мала великий успіх у моделюванні послідовності до послідовності.

Кодер виготовлений з трьох частин. Спочатку вивчається побудова слова. Потім побудова передається через Prenet — нейронна мережа для регуляризації. Нарешті, результати надходять до РНМ у двох напрямках. Структура кодера та декодера з'єднана через МУ. Перша частина — це кодер, який перетворює послідовність символів у вектор структури слова. Пізніше кодер використовується декодером для прогнозування спектрограм. Модель приймає символи як вхідні дані та видає відповідну спектрограму, яка потім подається в алгоритм відновлення Гриффіна-Ліма для синтезу мови.

Декодер складається з двошарової мережі ДКП — згорткової постмережі та нейромережі Prenet. Під час тренінгу основний кадр подається через Prenet і передається як вхід до шарів ДКП. Декодер являє собою авторегресивну періодичну нейронну мережу, яка прогнозує спектрограму з кодованої вхідної послідовності по одному кадру. Декодер використовує МУ

на виході кодера для створення кадрів мел-спектрограми. У декодері також є попередньо-мережевий шар, за яким слідує одношаровий вентильний рекурентний вузол, вихід якого з'єднується з виходом кодера для отримання контекстного вектора через механізм уваги. Потім цей вихід вентильного рекурентного вузла об'єднується з контекстним вектором, щоб отримати вхід блоку РНМ декодера. Модуль РНМ декодера виробляє кадри мел-спектрограми.

Вихід з з'єднаної МУ мережі ДКП надсилається через два проекційні шари. Перший проєктує інформацію на спектрограму, а другий проєктує її на стоп-токен.

Потім спектрограма направляється через згорткову постмережу для обчислення залишку, який слід додати до сгенерованої спектрограми. Додавання сгенерованої спектрограми та залишкових результатів у кінцевій спектрограмі відбувається з використанням алгоритма Гриффіна-Ліма. Окрім архітектурних відмінностей, важливою відмінністю є те, що Tacotron2 використовує Wavenet замість алгоритма Гриффіна-Ліма, щоб повернути аудіосигнал, який робить дуже реалістичним звучання мови.

1.2 Опис предметного середовища

Предметне середовище — це знання про середовище, в якому працюють установи і організації, і воно охоплює розуміння динаміки галузі, історії, секторів та сегментів, бізнес-моделі, конкурентного простору, цінової політики, клієнтів, бази даних та галузевих стратегій організації. Предметне середовище — це знання певної галузі. Необхідно мати певну компетенцію, пильність та швидкість у розумінні та вирішенні ділової ситуації таким чином, який, ймовірно, приведе до хорошого результату та правильного проєктування.

Набуття ділової проникливості та досвіду в галузі предметного середовища надає можливість приймати бездоганні рішення та робити

швидкі висновки, та надає можливість керівникам вирішувати складні бізнес-ситуації. Досягнення розуміння галузі означає більш продуманий аналіз, чіткішу логіку, що лежить в основі ділових рішень, більший пріоритет до ключових аспектів впровадження та функціонування та більш дисциплінованого управління продуктивністю.

Розуміння програми є важливою частиною всіх заходів з обслуговування та вдосконалення програмного забезпечення. Як зараз практикується, розуміння програми складається в основному з читання коду – цей метод розуміння дуже обмежений, так як намагається пов'язати систему з її призначенням або вимогами, тому виникає велика кількість питань щодо нерідко складної організації програми.

Приналежність до певної сфери суспільства або економіки, якої стосується програма — це предметне середовище цієї програми. Модель предметного середовища програми може слугувати доповненням до методів та інструментів аналізу та розробки програмного забезпечення. Доменна модель містить знання про межі предметного середовища, термінологію та можливі архітектури. Ці знання можуть допомогти аналітику визначити вимоги щодо призначення програми. Крім того, модель домену може надавати інформацію про те, як пов’язані між собою поняття домену. Представлення результатів розуміння програм на основі домену також важливо, і обговорюються різноманітні методи представлення.

Виділимо базові сутності даної предметної області, які утворюють структуру проєктованої програми:

- ISO 639 (таблиця Б.1) — набір стандартів Міжнародної організації зі стандартизації (ISO), пов’язаний зі стандартизацією назв мов і мовних груп;
- prenet — це нейронна мережа Google для регуляризації синтезу мовлення;
- WaveNet — це глибока нейронна мережа для генерації необробленого звуку;

– автоматизований переклад — це переклад, в процесі якого людина користується допомогою комп'ютерного програмного забезпечення – перевірка правопису, граматики, використання словників;
- алгоритм Гриффіна-Ліма — прогнозує викинуту фазову інформацію короткочасного перетворення Фур’є при перетворенні на спектрограмму. Ітераційно намагається знайти форму хвилі, величина якої при короткочасному перетворенні Фур’є ближче до генерованої спектрограми;
- вентильні рекурентні вузли — це вентильний механізм у рекурентних нейронних мережах, представлений 2014 року. Вони подібні до довгої короткочасної пам'яті з вентилем забування, але мають менше параметрів, оскільки не мають вихідного вентиля.
– вихідна мова — мова оригіналу, мова з якої робиться переклад.
– запит — процес звернення користувача до серверу з метою введення, отримання або зміни інформації в базі даних;
– згорткова нейронна мережа — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень;
– зникнення градієнта — це ускладнення, яке виникає в тренуванні штучних нейронних мереж методами навчання на основі градієнту та зворотного поширення. В таких методах кожен з вагових коефіцієнтів нейронної мережі отримує уточнення пропорційно до частинної похідної функції похибки по відношенню до поточної ваги на кожній ітерації тренування;
– інтерредагування — редагування тексту відбувається безпосередньо в процесі перекладу;
– клієнт — активне і окреме від сервера програмне забезпечення, що використовує дані, що поставляються сервером шляхом передачі клієнтських запитів серверу;
– код мови — короткі алфавітні або цифрові коди, розроблені для подання мов в обробці даних і комунікаціях;

– мел — психофізична одиниця висоти звуку, застосовується головним чином в музичній акустиці. Назва походить від слова «мелодія».
– мел-спектрограмма — це звичайна спектрограмма, де частота виражена не в Гц, а в мелах.
- машинний переклад — це автоматизований переклад у вузькому значенні — переклад з мінімальною участю людини (лише на етапах пост—, перед—, інтерредагування та його змішаних варіантах) ;
– машинний переклад на основі правил (rule-based machine translation) — системи машинного перекладу, засновані на лінгвістичній інформації про вихідні та цільові мови, в основному отримані (одномовні, двомовні або багатомовні) словники та граматики, що охоплюють основні семантичні, морфологічні та синтаксичні закономірності кожної мови відповідно;
- машинний переклад на основі прикладів (example-based machine translation) — Машинний переклад на основі прикладів — це метод машинного перекладу, який часто характеризується використанням двомовного корпусу з паралельними текстами як його основної бази знань під час виконання. Це, по суті, переклад за аналогією, і його можна розглядати як реалізацію підходу до обґрунтування машинного навчання на основі конкретного випадку.
- механізм уваги (англ. Attention mechanism, attention model) — техніка використовувана в рекурентних нейронних мережах (РНМ) і згортальних нейронних мережах (ЗНМ) для пошуку взаємозв'язків між різними частинами вхідних і вихідних даних;
- нейронний машинний переклад — це система нейронного машинного перекладу, розроблена компанією Google і представлена в листопаді 2016 року, яка використовує штучну нейронну мережу для підвищення швидкості і якості перекладу в Google Translate
– передредагування — попереднє редагування тексту людиною перед введенням для машинного перекладу;

- переклад — діяльність з інтерпретації сенсу тексту на вихідній мові(ВМ) і створення нового еквівалентного йому тексту перекладаємою мовою;
- перекладаєма мова — мова, на яку робиться переклад, мова перекладу;
- постредагування — це процес, в результаті якого людина виправляє машинний переклад і доводить його до бажаного рівня якості;
- рекурентна нейронна мережа — це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють граф орієнтований у часі;
- розпізнавання мовлення — процес перетворення мовленнєвого сигналу в текстовий потік;
- розрізнювальна модель — клас керованого машинного навчання, що використовується для класифікації або регресії. Вони відрізняють межі прийняття рішень шляхом виведення знань із спостережуваних даних;
- сервер — програмний компонент обчислювальної системи, що виконує сервісні (обслуговуючі) функції по запиту клієнта, надаючи йому доступ до певних ресурсів або послуг. Пасивна сторона системи клієнт-сервер;
- синтез мовлення — формування мовного сигналу з друкованого тексту;
- спектрограма — зображення, що показує залежність спектральної щільності потужності сигналу від часу. Спектрограми застосовуються для ідентифікації мови, аналізу звуків, обробки мови;
- статистичний машинний переклад (statistical machine translation) — парадигма машинного перекладу, де переклади створюються на основі статистичних моделей, параметри яких отримані з аналізу двомовних текстових корпусів;

1.3 Порівняльний аналіз існуючих рішень

1.3.1 Настільний застосунок Google Translate Client

Google Translate Client (рисунок 1.3) — це умовно безкоштовний додаток, яке використовує сервіс Google Translate [9].

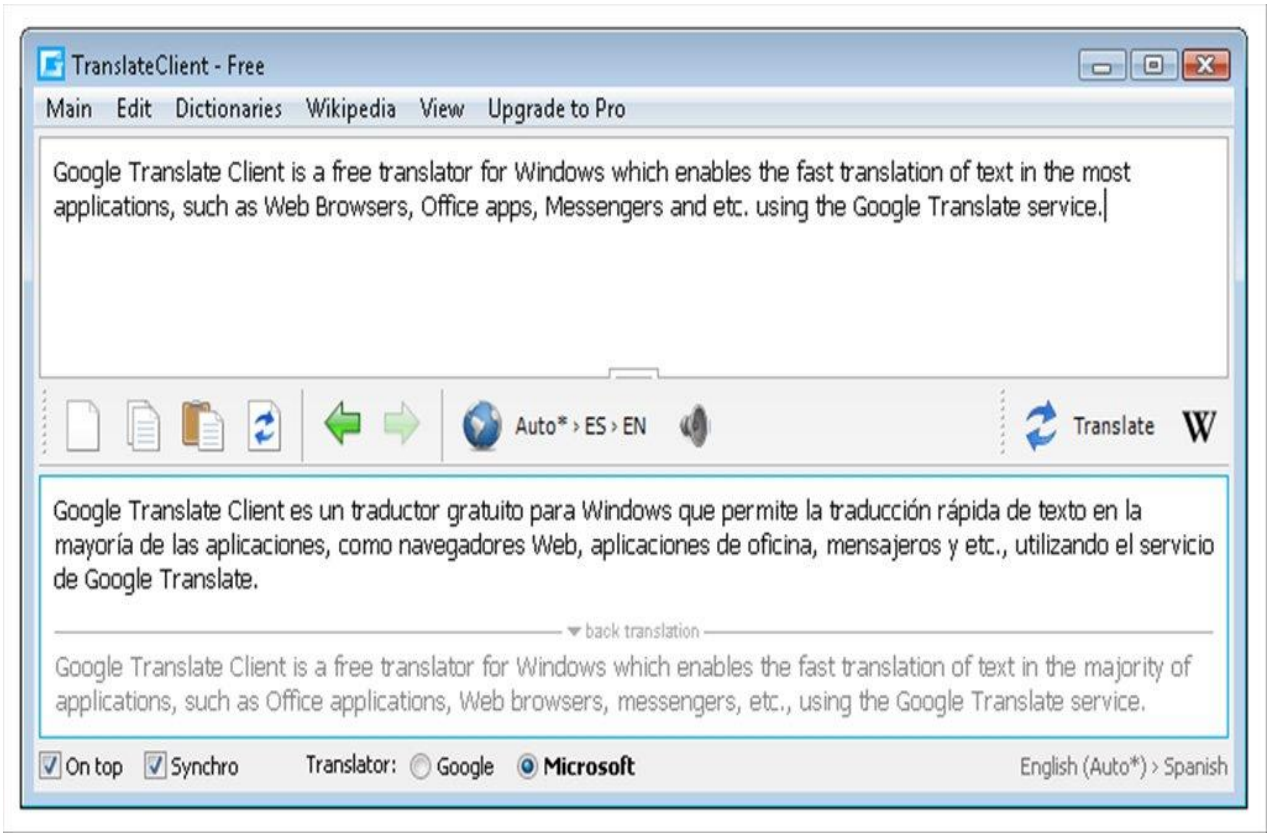

Рисунок 1.3 – Програма Google Translate Client [9]

Програма працює під управлінням операційної системи Windows, займає досить мало місця, і практично не нагружає систему. Цей перекладач працює тільки при наявності інтернет з'єднання, але при цьому споживає мінімальну кількість трафіку, і тільки під час самого перекладу. Головною особливістю програми є те що для перекладу будь-якого тексту, потрібно зробити мінімум дій.

Google Translate Client існує в двох версіях, звичайна, яка доступна для безкоштовного завантаження, і версія Pro, за яку потрібно платити кошти. В даному контексті я буду розглядати безкоштовну версію.

Є можливість виклику програми за допомогою гарячих клавіш. Якщо отриманий переклад Вас не задовольняє, є можливість відправити свій більш кращий переклад. Як плюс можна відзначити автоматичну перевірку оновлень.

Ця програма зручна в першу чергу тим що можна перевести як текст з сайту, але і текст інших ресурсів, наприклад, від якого-небудь англомовного ресурсу, наприклад для перекладу пошти.

Установка проста і не вимагає великих зусиль. Portable версія Google Translate Client не вимагає установки, Ви можете завантажити її, запустити і відразу працювати з програмою, як варіант можна зберігати цю програму на флешці і переносити її з комп'ютера на комп'ютер.

Пункт «перекласти з» — тут зі списку можна вибрати мову з якогось потрібно буде здійснити переказ, пункт «улюблені мови» — якщо поставити тут прапорець, то при визначенні мов будуть в першу чергу використовуватися обрані вами мови, і якщо натиснути на посилання то, можна буде шляхом вибору зі списку вибрати передбачувані мови для автоматичного визначення, пункт «перекласти на» — в цьому розділі можна буде вибрати зі списку на яку мову програма повинна перекладати.

Пункт «налаштування» — цей пункт меню дає змогу настройки програми, всі налаштування поділяються на кілька розділів, а саме: пункт «дія для перекладу» — тут є можливість вибору з трьох варіантів, іншими словами для того що б перевести текст в якійсь програмі потрібно буде зробити наступне — пункт «виділити текст і натиснути на значок G», або «виділити текст і натиснути на ліву кнопку миші», або «просто вибрати текст». На мій погляд найзручніший — це перший варіант.

Пункт «гарячі» клавіші – «використовувати «гарячі» клавіші» тут можна вибрати власні поєднання клавіш, при якому буде відкриватися головне вікно програми. Щоб вибрати потрібне поєднання потрібно поставити курсор в поле і натиснути на потрібне поєднання клавіш відповідно , а що б прибрати можливість запуску після натискання гарячих клавіш потрібно зняти прапорець поруч з написом use hot keys.

Пункт «запуск» — тут можна встановити наступне: запуск Google Translate Client разом із запуском Windows, timeout запиту, — тут можна

виставити час очікування запиту, «повторити з'єднання через» — далі потрібно вказати кількість часу в секундах.

Пункт «новини та оновлення» — з заголовка розділу ясно що тут можна налаштувати отримання новин та оновлень: «отримувати сповіщення про новини», «автоматична перевірка оновлень», «автоматичне завантаження і установка оновлень».

«Спливаюче вікно» — тут присутній ряд налаштувань, за допомогою яких можна налаштувати спливаюче вікно Google Translate Client.

«Інші» — тут знаходяться всі інші настройки, які не підходять не під один розділ, а саме: «використовувати «запропонувати переклад краще» — тут можна запропонувати свій переклад, якщо поточний переклад Вас не влаштує, «використання вікіпедії», «розумний пошук в вікіпедії», «Використовувати пошук» — тут можна вибрати зі списку одну з пошукових систем, і при цьому можна нижче в рядку вказати як програма повинна звертатися до пошукача.

У самому низу програми розташовано три checkbox:
Поверх всіх вікон — відображати головне вікно програми по верх всіх вікон. Синхронна — переклад по ходу друку, і подальше копіювання в буфер обміну. Трансліт — перевести весь перекладений текст в трансліт (латинськими літерами).

Після того як програма встановиться вона буде прописана в автозавантаження, і основна робота з програмою полягає не з головним вікном програми, а з її іконкою в системному треї.

Іконка в треї відображається в двох кольорах — синій і помаранчева. Якщо іконка синя то Google Translate Client неактивний, якщо помаранчева то клієнт активний. За допомогою лівого одинарного кліка мишкою по іконці можна включити або виключити програму для запущеного в даний момент програми. Подвійний клік лівою кнопкою миші викличе головне вікно

програми. Правий клік покаже контекстне меню, яке трохи повторює вище описане головне меню програми.

1.3.2 Настільний застосунок Dictionary .NET

Dictionary.NET (рисунок 1.4) використовує технологію Google Translate для миттєвого перекладу різними мовами у всіх можливих комбінаціях [10].

Рисунок 1.4 – Програма Dictionary.NET [10]

Dictionary.NET не потребує установки та працює непомітно у фоновому режимі. Ви можете відкрити її за допомогою комбінації клавіш, комбінації миші або з іконки програми у системному треї, а потім ввести будь-яке слово чи текст, які ви хочете перекласти. Інший варіант — можливо, простіше — це вибрати задане слово чи текст, а потім використовувати заздалегідь задану комбінацію клавіш або комбінацію миші, щоб вивести Dictionary.NET з уже завантаженим перекладом.

Dictionary.NET — це програма, за допомогою якої можна шукати переклади і визначення слів. Він працює від Google Translate.

Це портативна програма, тому встановлювати Dictionary.NET не потрібно. Отже, ви можете розмістити програму на USB-флешці та запустити її на будь-якому комп'ютері. Що ще важливіше, ваші записи реєстру Windows залишаться недоторканими.

Інтерфейс інструменту чистий та інтуїтивно зрозумілий. Ви можете перемикатися між двома вкладками, щоб вибрати мову вводу та виводу. Ви можете робити закладки на словах, користуватися функцією «текст у мову», а також текстовим редактором та віртуальною клавіатурою. Якщо ви відкриєте контекстне меню, ви можете скопіювати, вставити та зберегти текст у вигляді HTML або TXT-файлу, а також завантажити OCR.

У меню «Параметри» можна включити Dictionary.NET під час запуску ОС, мінімізувати запуск та перевірити наявність оновлень. Але ви також можете вибрати рівень непрозорості та команду активації миші для перекладу, змінити шпалери, видалити словники з інтерфейсу тощо.

Загалом, словник .NET — це дуже хороша альтернатива ручному пошуку визначень слів і перекладу в Інтернеті (оскільки це дуже трудомісткий процес), рекомендується до використання всім користувачам.

1.3.3 Настільний застосунок QuestSoft QTranslate

Qtranslate (рисунок 1.5) — це невеликих розмірів безкоштовна програма перекладач, за допомогою якої можна швидко здійснити переклад різних слів і текстів будь-якої складності на різні мови світу [11]. Переклад дана програма здійснює шляхом використання можливостей безкоштовних онлайн сервісів або за допомогою локальних словників, які можна вільно скачати з офіційного сайту. Незважаючи на свої малі розміри, Qtranslate володіє гарною функціональністю і поширюється абсолютно безкоштовно без будь-яких обмежень, завдяки чому може стати відмінним помічником у навчанні, роботі або повсякденному житті.

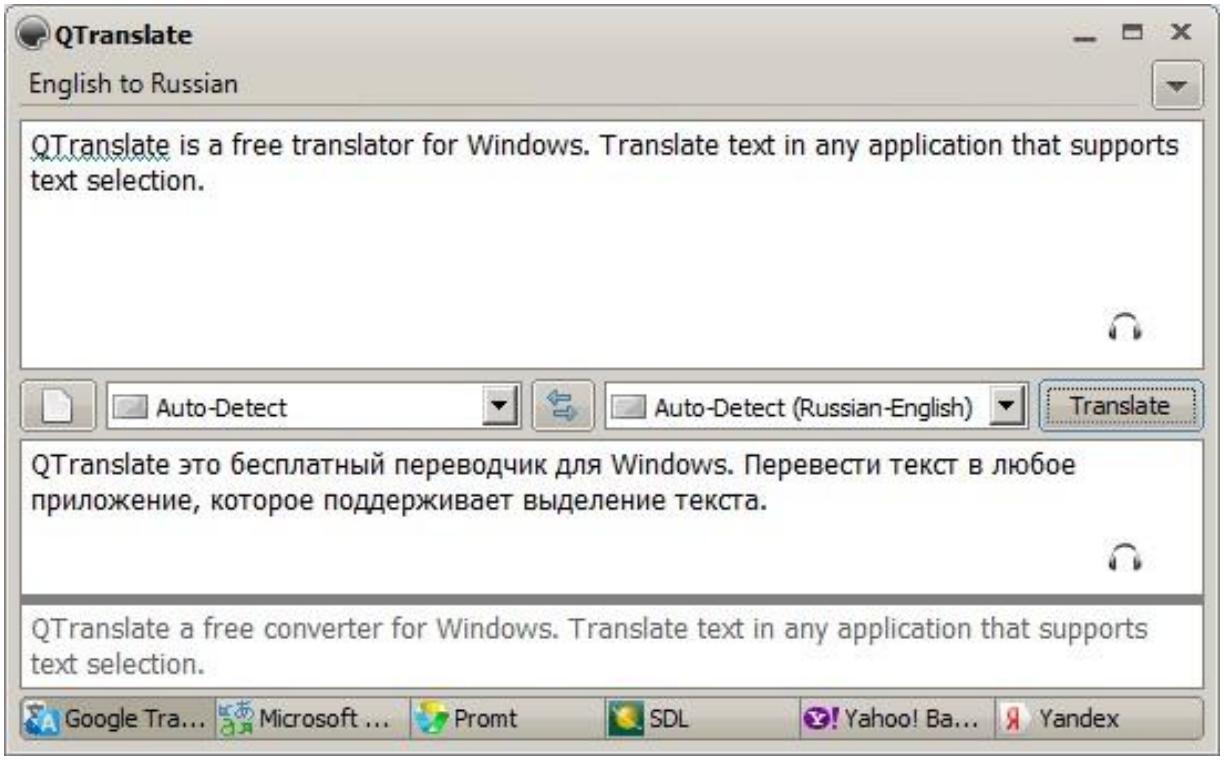

Рисунок 1.5 – Програма QuestSoft QTranslate [11]

Онлайн перекладач Qtranslate надає можливість користувачеві здійснювати переклад тексту на українську або інші мови як за допомогою контекстного перекладу в різних програмах, що підтримують виділення тексту, так і шляхом введення тексту безпосередньо в головному вікні програми. При використанні контекстного перекладу досить виділити слово, словосполучення або цілу фразу в будь-якому додатку, будь-то браузер google chrome, microsoft office, sublime text editor, telegram і ін. В результаті чого користувач при включеній даної функції миттєво отримає переклад в контекстному вікні в форматі обраному в настройках «Режим миші». При здійсненні перекладу шляхом використання основного вікна програми перекладач Qtranslate підтримує кілька методів введення тексту, що перекладається: безпосередньо прямого набору тексту, з використанням буфера обміну — шляхом копіювання і вставки, а також за допомогою функції «голосове введення».

Також для збільшення швидкості роботи користувача з даною програмою в Qtranslate реалізована можливість використання комбінацій гарячих клавіш, які можна налаштовувати. Крім цього, для того, щоб онлайн переклад тексту, наприклад переклад з англійської на українську, був більш коректним, в програмі є можливість вибору онлайн сервісу (google, baidu, babylon, deepL, microsoft, yandex, promt та ін.), за допомогою якого він буде здійснюватися.

Основними можливостями програми Qtranslate є переклад тексту в будь-якому додатку, яке підтримує виділення тексту, вбудована функція розпізнавання мови для здійснення голосового введення тексту, підтримка великої кількості сервісів онлайн перекладу: google, baidu, babylon, deepL, microsoft, yandex, promt та ін.

Крім того, можливість розпізнавання тексту з фотографії, можливість відтворення як тексту, що перекладається, так і перекладеного тексту за допомогою синтезу мовлення, можливість пошуку в онлайн словниках, вбудована перевірка орфографії, функція автоматичного визначення мови тексту, що перекладається, можливість збереження перекладів в історії, віртуальна клавіатура.

1.3.4 Настільний застосунок Dicter

Dicter (рисунок 1.6) — це невеликих розмірів безкоштовна онлайн програма перекладач, за допомогою якої можна швидко здійснити переклад різних слів і текстів будь-якої складності на різні мови світу [12].

Якщо доводиться часто робити переклад слів, виразів або тексту, програма Dicter дозволяє набагато зручніше використовувати можливості перекладача Google Translate.

Принцип роботи програми Dicter дуже простий — досить виділити незнайомий текст і натиснути на клавіатурі клавіші CTRL + ALT — в результаті отримуємо переклад виділеного тексту в красивому спливаючому

вікні, в якому можна копіювати перекладений текст, змінювати напрямок перекладу і прослухати виділений текст і переклад.

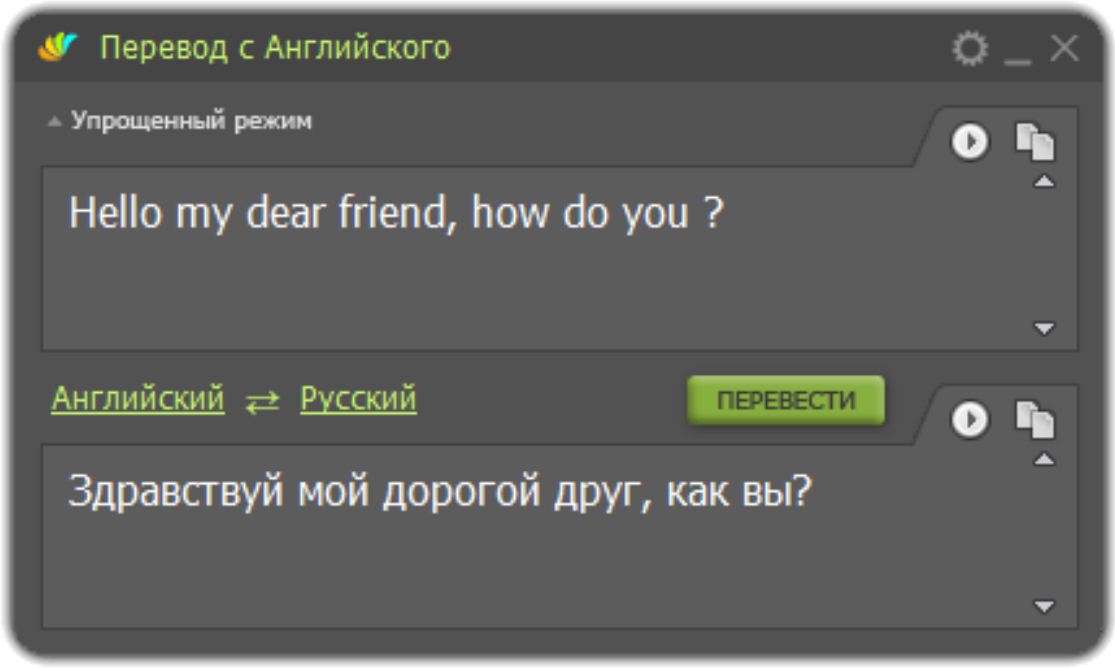

Рисунок 1.6 – Програма Dicter [12]

На відміну від інших перекладачів, Dicter не встановлює на комп'ютер сотні мегабайт словникових баз, які можуть ніколи не знадобитися. Єдина особливість — для здійснення переказу потрібно бути підключеним до Інтернет.

Висновки до розділу 1

У даному розділі розглянуто Google Translate та його предметну область. Google використовує самонавчальний алгоритм машинного перекладу — у березні 2017 року компанія повністю перевела механізм перекладу на нейромережі для більш якісного перекладу. Сервіс самостійно пристосовується до нової лексиці — оновлюється і розвивається паралельно з мовою. Тобто, якщо з'являється нове слово або змінюється його лексичне значення, то система розпізнає його і оновлює свою базу даних. І, як наслідок, прискорюється «навчання» системи і вдосконалюється якість

генерації тексту. На даний момент відомо, що «корпус» Google Translate володіє більш ніж трильйоном слів.

Виділено базові сутності предметної області, які утворюють структуру проєктованої програми.

До порівняльного аналізу було обрані кращі настільні застосунки, що використовують сервіс Google Translate. До уваги взяті наступні характеристики, які впливають на розробку власної реалізації – багатоплатформність, привабливий та простий дизайн, миттєвий переклад за гарячою клавішею, підтримка розробником, відкрите програмне забезпечення, актуальні мови Google translate, унікальні можливості.

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Діаграма використання

Діаграма використання — це потужна методика для вивчення потреб користувачів [13, 14]. Вони корисні для презентацій для керівників та зацікавлених сторін проєкту, але діаграма використання має значно більшу цінність, оскільки вона спрощує описання заплутаних та не зовсім зрозумілих актуальних вимог до програмного забезпечення.

Для складання діаграми використання спочатку потрібно ретельно проаналізувати всю систему. Ви повинні з'ясувати кожну функцію, яка надавати розроблювальна система. Після виявлення всіх функціональних можливостей системи, ці функції перетворюються на різні випадки використання, які будуть використані у діаграмі використання. Організувавши випадки використання, ми повинні виявити різних акторів, які збираються взаємодіяти із системою.

Після виявлення акторів та випадків використання, вам доведеться вивчити взаємозв'язок конкретного актора з випадком використання або системою. Потрібно визначити загальну кількість способів взаємодії актора з системою. Один актор може взаємодіяти з кількома випадками використання одночасно, або він може взаємодіяти з численними випадками використання одночасно.

Під час малювання діаграм використання потрібно використовувати елементи зазначені на рисунку 2.1.

- актор — це людина, організація або зовнішня система, яка відіграє роль в одній або декількох взаємодіях з вашою системою. Актори малюються як людиноподібні силуети;
- прецедент — випадок використання, дія. Описує послідовність дій, які надають актору щось вимірне значення і малюється як овал;

– граничні межі системи — охоплюють усі випадки використання у системі, позначається прямокутником. Граничні рамки системи використовуються рідко.

Рисунок 2.1 — Елементи діаграми використання

Елементи взаємодії діаграми використання:

– використовує — користувач виконує дію;
– включає — один прецедент використовує інший;
– розширює — представлення дочірніх прецедентів;
– вимагає — наступний прецедент вимагає виконання попереднього;
– схожий — прецеденти подібні, але описують різну функціональність;
– рівнозначний — подібна функціональність, але користувач сприймає, як різну.

2.2 Перелік вимог

Програмні вимоги — це опис особливостей та функціональних можливостей для застосунку, що розроблюється. Вимоги передають

очікування користувачів від програмного продукту. Визначено наступні вимоги до застосунку:

– реалізація мовою програмування Java 8 відповідно до потреб організації (додаток Г);

– швидкий переклад — переклад тексту у маленькому спливаючому повідомленні, виділеного мишкою в будь-якій програмі – веб-браузер, провідник, текстовий редактор та ін..;

– багатоплатформність – можливість використання на операційних системах Windows, Linux, Mac;

– реалізувати гарячі клавіші;

– спливаюче повідомлення повинно з’являтися біля курсора миші;

– спливаюче повідомлення повинно мати функцію синтезу мовлення;

– налаштування швидкості синтезу;

– синтез перекладаємого або перекладеного тексту;

– автоматичне розпізнання мови перекладаємого тексту;

– голосове ведення тексту для перекладу;

– відміна тривалих операції – синтез тексту, голосове ведення;

– меню-слайдер з налаштуваннями в стилі «андроїд» замість окремого вікна з налаштування;

– настільний застосунок;

– зберігати останні налаштування користувача – мови, історію та ін.;

– GUI повинен мати світлу та темну кольорову схеми, які зручно використовувати в залежності від освітленості в приміщенні;

– відібрані мови — можливість швидкого вибору потрібних мов для перекладу;

– додати всі доступні мови сервісу google translate.

2.3 Сценарії використання

2.3.1 Успішний сценарій перекладу тексту

Описання головного успішного сценарію та його розширення:

а) Користувач вибирає мови.

1) Відсутня потрібна мова у списку швидкого вибору.

— Користувач відкриває список усіх доступних мов у меню та обирає потрібні.

2) Вихідна мова оригінального тексту невідома.

— Користувач включає режим автовизначення мови.

б) Користувач вводить текст для перекладу.

1) Текстове поле містить текст від попереднього запуску.

— Користувач натискає кнопку очищення текстових полів.

в) Система перевіряє рекомендований об’єм тексту.

г) Система відправляє текст на сервер.

1) На переклад поданий текст з помилкою.

— Користувач відміняє переклад.

д) Система повертає переклад з серверу.

2.3.2 Успішний сценарій синтезу мовлення

Описання головного успішного сценарію та його розширення:

а) Користувач вводить текст для синтезу;

1) Текстове поле містить текст від попереднього запуску.

— Користувач натискає кнопку очищення текстових полів.

б) Система перевіряє рекомендований об’єм тексту;

в) Система відправляє текст на сервер;

г) Система повертає аудіофайл з серверу;

д) Система запускає на прослуховування аудіофайл.

1) Програвання аудіофайлу триває довго.

— Користувач відміняє прослуховування.

2) Програвання аудіофайлу має незадовільну швидкість

— Користувач відміняє прослуховування.

— Користувач налаштовує потрібну швидкість та висоту тону звучання слів у меню.

2.3.3 Успішний сценарій розпізнання мовлення

Описання головного успішного сценарію та його розширення:

а) Користувач вмикає мікрофон.

1) Текстове поле містить текст від попереднього запуску.

— Користувач натискає кнопку очищення текстових полів.

2) Весь текст промовлений.

— Користувач відміняє подальше розпізнання.

б) Користувач промовляє текст для синтезу.

в) Система відправляє аудіофайл на сервер.

г) Система повертає розпізнаний текст з серверу.

2.3.4 Успішний сценарій швидкого перекладу тексту

Описання головного успішного сценарію та його розширення:

а) Користувач виділяє текст для перекладу.

б) Користувач натискає комбінацію клавіш для перекладу.

в) Система отримує виділений текст через буфер обміну.

г) Система відправляє текст на сервер.

д) Система повертає переклад з серверу.

е) Система генерує спливаюче повідомлення з перекладом.

На основі сценаріїв створено діаграму використання ІАЗ61.020БАК.005 Д1. Вона корисна для презентацій для керівників та зацікавлених сторін проєкту, але діаграма використання має значно більшу цінність, оскільки вона спрощує описання заплутаних актуальних вимог до програмного забезпечення.

Висновки до розділу 2

У даному розділі виконано аналіз вимог до програмного забезпечення – вимоги до синтезу мовлення, перекладу тексту, розпізнання мовлення, багато потоковості, GUI, мов.

Діаграма використання — це потужна методика для вивчення потреб користувачів. Вони корисні для презентацій для керівників та/або зацікавлених сторін проєкту, але діаграма використання має значно більшу цінність, оскільки вона спрощує описання заплутаних та не зовсім зрозумілих актуальних вимог до програмного забезпечення.

Після виявлення всіх функціональних можливостей системи, ці функції перетворюються на різні випадки використання, які будуть використані у діаграмі використання. Організувавши випадки використання, ми повинні виявити різних акторів, які збираються взаємодіяти із системою.

Сценарії використання — це письмовий опис того, як користувачі будуть користуватися функціями програми. З точки зору користувача, він визначає поведінку системи під час відповіді на запит. Кожен випадок використання представлений у вигляді послідовності простих кроків, починаючи з цілі користувача і закінчуючи, коли ця мета виконується.

Після виявлення акторів та випадків використання, вам доведеться вивчити взаємозв'язок конкретного актора з випадком використання або системою. Потрібно визначити загальну кількість способів взаємодії актора з системою. Один актор може взаємодіяти з кількома випадками використання одночасно, або він може взаємодіяти з численними випадками використання одночасно.

3 ЗАСОБИ РОЗРОБКИ

3.1 Середовище розробки Eclipse

Середовищем розробки обрана Eclipse IDE [15].

Розроблена за допомогою Java, платформа Eclipse може використовуватися для розробки Rich internet application клієнтських додатків, інтегрованих середовищ розробки та інших інструментів. Eclipse може використовуватися як IDE для будь-якої мови програмування, для якої доступний плагін. Її часто вибирають програмісти, яким подобається настроювати середовище розробки під себе, підключаючи додаткові плагіни для створення IDE мрії. Платформа Eclipse, яка забезпечує основу для Eclipse IDE, складається з плагінів і розроблена таким чином, щоб розширюватися за допомогою додаткових плагінів. Також в даному середовищі розробки присутня відмінна візуалізація коду для більш легкого сприйняття.

В Eclipse можна відкрити кілька проєктів в одному workspace, що значною мірою знижує з модульними проєктами або частковим перенесенням коду з одного проєкту в інший.

Eclipse дозволяє автоматично додавати імпортування бібліотек, більше не має необхідності додавати їх вручну — це заощадить багато часу при використанні Eclipse для створення коду Java.

Eclipse — один з найкращих прикладів створення чудового та безкоштовного застосунку з відкритим кодом. Нам не доведеться платити за його використання.

3.2 Середовище розробки JavaFX Scene Builder

JavaFX Scene Builder — це візуальний інструмент проєктування, який дозволяє користувачам швидко проєктувати користувальницькі інтерфейси додатків JavaFX без написання коду [16]. Користувачі можуть перетягувати компоненти інтерфейсу користувача до робочої області, змінювати їх

властивості, застосовувати таблиці стилів, а FXML-код для створеного ними макета автоматично генерується у фоновому режимі. Результат — файл FXML, який потім можна комбінувати з проєктом Java, прив'язавши інтерфейс користувача до логіки програми.

Scene Builder дозволяє легко компонувати елементи управління, діаграми, форми та контейнери JavaFX, що дозволяє швидко прототипувати інтерфейси користувачів. Анімації та ефекти можна легко застосувати для більш досконалих інтерфейсів користувача.

Scene Builder генерує FXML — мову розмітки на основі XML, яка дозволяє користувачам розробляти користувальницький інтерфейс програми, окремо від логіки програми. Ви також можете відкривати та редагувати наявні файли FXML, авторами яких є інші користувачі.

Scene Builder може використовуватися в поєднанні з будь-яким IDE для програмування на Java. Ви можете прив'язати інтерфейс користувача до вихідного коду, який буде обробляти події та дії, зроблені на кожному елементі за допомогою простого процесу, запустити додаток у Eclipse, а будь-які зміни FXML у Eclipse також відзначаться у вашому проєкті Scene Builder.

У будь-який час під час створення вашого проєкту ви можете попередньо переглянути, як буде виглядати користувальницький інтерфейс при розгортанні, меню та палітрі інструменту.

Scene Builder написано як додаток JavaFX, підтримується в Windows, Mac OS X та Linux. Це ідеальний приклад повноцінної настільної програми JavaFX. Scene Builder упаковується як автономний додаток, а це означає, що він постачається в комплекті з влаштованою копією JRE.

Є можливість застосувати обраний зовнішній вигляд до макета GUI, використовуючи таблиці стилів. Це так просто, як вибрати компонент GUI та вказати на обраний вами CSS файл із панелі властивостей. Аналізатор CSS

дозволяє зрозуміти, як конкретні правила CSS можуть впливати на аспекти компонента JavaFX.

3.3 Мова програмування Java 8

Проект був розроблений на замовлення сторонньої організації, на вимоги якої необхідне використання мови програмування Java 8 (Додаток Г). Це пояснюється тим, що організація вже успішно використовує застосунки написані на Java 8 і не планує розширювати вдалий стек використаних технологій в середині організації. Крім того Java 8 має великі переваги [17].

Величезна перевага Java полягає в тому, що на цій мові можна створювати програми, здатні працювати на різних платформах.

Відмінною особливістю Java в порівнянні з іншими мовами програмування загального призначення є забезпечення високої продуктивності програмування, ніж продуктивність роботи програми або ефективність використання ним пам'яті. Java демонструє міць об'єктно-орієнтованої розробки програм, поєднуючи простий і знайомий синтаксис з надійним і зручним в роботі середовищем розробки.

Java 8 поставляється з досить об'ємною бібліотекою розміром 4240 класів, що надає програмісту великий вибір об'єктів для абстрагування багатьох системних функцій, використовуваних при роботі з вікнами, мережею, введення-виведення та ін.

Java — це відкрите програмне забезпечення. Одна із найпопулярніших мов і понині не дивлячись на майже 30 річне існування. Тому на Java програмують багато людей, за цей час накопичилось багато бібліотек і проєктів, які можна досліджувати, використовувати та розширяти своїми напрацюваннями.

Java — найпопулярніша мова в світі, який сьогодні працює на мільярді пристроїв, що працюють на платформі Java, Java торкнулася будь-якої галузі розробки програмного забезпечення, особливості Java роблять її

найпотужнішою мовою на сучасному ринку розробки програмного забезпечення.

Це мова високого рівня. Це поняття передбачає, що мова програмування більше нагадує людську мову, а не машинну. Отже, писати, читати та підтримувати його слід легко і просто.

Стійкість. Кажуть, що рішення, створені за допомогою Java, є стабільними. Частково це відбувається так, оскільки щодня випускається нова версія Java з новими функціями з розширеними функціями.

Об'єкти-орієнтовна мова. Оскільки Java належить до об'єктно-орієнтованого програмування, вона дозволяє розробнику писати типові програми та повторно використовувати код. Отже, можна розробляти класи, генерувати об'єкти класів, працювати та підтримувати взаємодію між двома об’єктами.

Дешеве обслуговування. Характер роботи програми Java не покладається на якусь унікальну апаратну інфраструктуру, тому запустити сервер можна на будь-якій машині. Результат: це недороге обслуговування.

Безпека. Java — перша технологія, яка забезпечила безпеку невід'ємною частиною дизайну. JVM має спеціальний ідентифікатор, який визначає байт-код і перевіряє перед запуском.

Багатопотоковість. Внутрішня програма Java може виконувати кілька

Портативність. Портативність означає, що розробник повинен один раз написати лише один код і програму можна запустити на будь-якій платформі. Єдина умова — ця платформа повинна підтримувати JVM.

Надійність. Java є найбільш надійною та потужною мовою. Його компіляторам вдається виявити кожен тип помилок у вашому коді. Крім того, у Java є такі чудові функції, як обробка винятків та «збирання сміття», які також доводять надійність Java.

3.4 Використані бібліотеки

JavaFX — це програмна платформа для створення та розгортання настільних додатків, а також Rich internet application, які можуть працювати на широкому спектрі пристроїв [18, 19]. JavaFX призначений замінити Swing як стандартну бібліотеку графічного інтерфейсу для Java SE, але обидва будуть включені в осяжному майбутньому. JavaFX має підтримку настільних комп'ютерів та веб-браузерів у Microsoft Windows, Linux та macOS.

ControlsFX — це проєкт з відкритим кодом для JavaFX, який спрямований на забезпечення високоякісних елементів управління інтерфейсом та інших інструментів для доповнення основної дистрибуції JavaFX [20].

Java flac encoder — забезпечує кодування FLAC, реалізоване в Java. Він розроблений для забезпечення легкого додавання кодування FLAC в Java застосунки.

JUnit – бібліотека для модульного тестування для мови програмування Java.

Висновки до розділу 3

У даному розділі було виконано моделювання архітектури, інтерфейсу, вибір допоміжних бібліотек, обґрунтування вибору мови, детальна реалізація програмного забезпечення.

Середовищем розробки обрана Eclipse IDE. Платформа Eclipse, яка забезпечує основу для Eclipse IDE, складається з плагінів і розроблена таким чином, щоб розширюватися за допомогою додаткових плагінів.

JavaFX Scene Builder — це візуальний інструмент проєктування, який дозволяє користувачам швидко проєктувати користувальницькі інтерфейси додатків JavaFX без написання коду. Користувачі можуть перетягувати компоненти інтерфейсу користувача до робочої області, змінювати їх властивості, застосовувати таблиці стилів, а FXML-код для створеного ними

макета автоматично генерується у фоновому режимі. Результат — файл FXML, який потім можна комбінувати з проєктом Java, прив’язавши інтерфейс користувача до логіки програми.

Java — найпопулярніша мова в світі, який сьогодні працює на мільярді пристроїв, що працюють на платформі Java, Java торкнулася будь-якої галузі розробки програмного забезпечення, особливості Java роблять її найпотужнішою мовою на сучасному ринку розробки програмного забезпечення.

Використані додаткові бібліотеки, такі як JavaFX, ControlsFX, Java flac encoder , JUnit .

4 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Шаблон проєктування MVC

Архітектура реалізована на шаблоні MVC [21]. Дані програми, які призначені для користувацького інтерфейсу і керуючої логіки поділено на три окремих компоненти: модель (model), представлення (view) і контролер (controller) таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан. Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі. Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Шаблон MVC є звичним явищем у розробці. MVC — це схема розробки програмного забезпечення, що складається з трьох основних об'єктів:

- модель — це місце, де зберігаються дані. Містить в собі характеристику станів процесів, об'єкти моделей, парсер, мережевий код;
- представлення – «обличчя» програми;
- контролер є посередником між представленням та моделю за допомогою шаблону делегування. В кращому випадку об’єкт контролера «не знатиме», з чим він має справу.

Схема з’єднання шаблону виглядає приблизно так (рисунок 4.1).

Кожен з цих об'єктів відокремлений один від одного, і кожен виконує певну роль. Як і у багатьох випадках у розробці програмного забезпечення, нічого не є ідеальним, і MVC також не є.

Модель . Рівень моделі охоплює дані застосунку. Інші класи та об'єкти також включені до цього шару. Шари абстрагування — типові об’єкти-

менеджери, які часто виступають мостом між іншими класами. Це також можуть бути обгортки навколо нижчого рівня, більш надійні API. Джерела даних та делегати – розділення обов'язків класів, які будуть джерелом даних або делегатом інших компонентів, таких як таблиць або колекції. Константи — застосовування файлів із константами, таким чином, є можливість повторного використовувати формати дат, кольорів тощо. Розширення – легке додавання методів розширення можливостей рядків, колекцій тощо.

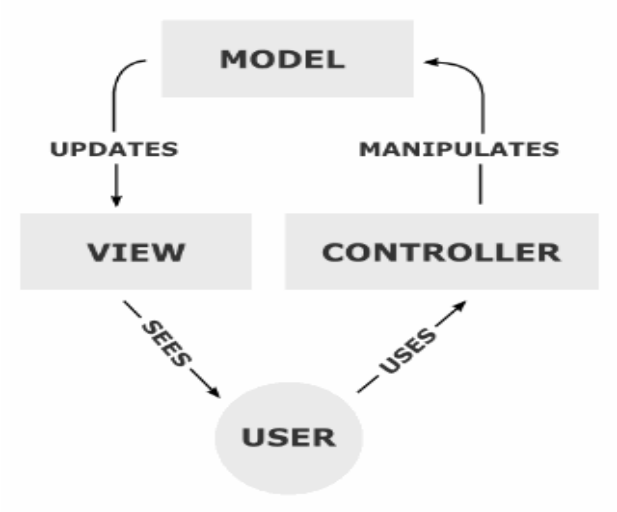

Рисунок 4.1 — Патерн MVC [21]

Представлення. Коли користувач взаємодіє зі застосунком, він взаємодіє з шаром представлення. Він не повинен містити жодної бізнес—логіки. Під час перегляду коду ви зазвичай бачите підкласи UI, основну анімацію та графіку. Мережеві виклики, бізнес логіка, маніпулювання моделями не стосується інтерфейсу користувача.

Контролер. Рівень контролера є частиною застосунку з небагаторазовим використанням у подальших розробках інших застосунків, оскільки включає правила, пов'язані з предметним середовищем. Потім контролер буде використовувати всі елементи у вашому шарі моделі для визначення потоку інформації у вашій програмі. Рівень контролера як мозок або двигун програми — він вирішує, що буде далі.

Жодна архітектура, стара чи нова, не є «срібною кулею», і завжди слід спершу орієнтуватися на хороші інженерні принципи, такі як SOLID.

4.2 Принципи проєктування SOLID

Принципи SOLID — це п'ять пунктів проєктування класів та управління залежностями для об'єктно-орієнтованого програмування та проєктування [22]. Абревіатура SOLID була введена Робертом Мартіном, відомим серед розробників як «дядько Боб». Кожна буква абревіатури є назвою принципу. Під час роботи з програмним забезпеченням, в якому керування залежністю виконується погано, код може стати «жорстким» , негнучким та важко розширюваним. Жорсткий код — це код, що важко змінити, змінити існуючий функціонал або додати нові функції. Ненадійний код чутливий до введення помилок, особливо тих, які з’являються в модулі, коли змінюється інша область коду. Якщо дотримуватися принципів SOLID, ви можете створити більш гнучкий та надійний код, який має більш високу можливість повторного використання.

Принцип єдиної відповідальності (SRP — The Single Responsibility Principle) означає, що ніколи не повинно бути більше однієї причини для зміни класу. Це означає, що необхідно розробити класи так, щоб кожен мав єдину мету. Це не означає, що кожен клас повинен мати лише один метод, але всі члени в класі пов'язані з основною функцією класу. Якщо клас має декілька обов'язків, їх слід розділити на нові класи. Коли клас має кілька обов'язків, збільшується ймовірність того, що його потрібно буде змінити. Щоразу, коли клас змінюється, зростає ризик введення помилок. Зосереджуючись на одній відповідальності, цей ризик обмежується.

Принцип відкритості / закритості (OCP — The Open Closed Principle) означає, що програмні об'єкти (класи, модулі, функції тощо) повинні бути відкритими для розширення, але закритими для модифікації. У «закритій» частині правила зазначено, що після того, як модуль був розроблений і

протестований, код слід коригувати лише для виправлення помилок. «Відкрита» частина говорить про те, що ви повинні мати можливість розширити існуючий код, щоб запровадити нові функції. Як і у принципі єдиної відповідальності, цей принцип зменшує ризик введення нових помилок шляхом обмеження змін до існуючого коду.

Принцип заміщення Лісков (LSP — The Liskov Substitution Principle) означає, що «функції, що використовують посилання на базові класи, повинні мати можливість використовувати об'єкти похідних класів, не знаючи цього». «Під час роботи застосунка код, який використовує базовий клас, повинен мати можливість замінити підклас, не знаючи цього». Принцип названий на честь Барбари Лісков. Якщо ви створюєте клас із залежністю від даного типу, ви повинні мати можливість надати об’єкт цього типу або будь-який його підклас, не вводячи несподіваних результатів, і без залежного класу знати фактичний тип наданої залежності. Якщо тип залежності необхідно перевірити, щоб поведінку можна було змінити відповідно до типу, або якщо підтипи породжували несподівані правила чи побічні ефекти, код може стати більш складним, жорстким та крихким.

Принцип поділу інтерфейсу (ISP — The Interface Segregation Principle) означає, що код клієнта не слід змушувати залежати від інтерфейсів, які вони не використовують. Це правило означає, що коли один клас залежить від іншого, кількість членів в інтерфейсі, який видно залежному класу, слід мінімізувати. Часто, коли ви створюєте клас із великою кількістю методів та властивостей, клас використовується іншими типами, які потребують доступу лише до одного або двох членів. Класи щільніше поєднуються, оскільки кількість членів, про які вони знають, зростає. Коли ви стежите за ISP, великі класи реалізують кілька менших інтерфейсів, які групують функції відповідно до їх використання. Класи пов'язані з цим принципом для більш слабкого зв'язку, підвищення надійності, гнучкості та можливості повторного використання.

Принцип інверсії залежності (DIP — The Dependency Inversion Principle) — останнє з п'яти правил. DIP має два правила. Перше полягає в тому, що модулі високого рівня не повинні залежати від модулів низького рівня. Обидва повинні залежати від абстракцій. Друге правило полягає в тому, що абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій. DIP в першу чергу стосується концепції шарування в межах додатків, де модулі нижчого рівня займаються дуже детальними функціями, а модулі вищого рівня використовують класи нижчого рівня для досягнення більш великих завдань. Принцип визначає, що там, де існують залежності між класами, їх слід визначати за допомогою абстракцій, таких як інтерфейси, а не шляхом прямого посилання на класи. Це зменшує крихкість, викликану змінами модулів низького рівня, що вводять помилки у вищі шари. DIP часто зустрічається із застосуванням ін'єкції залежності.

4.3 Мережева архітектура «Клієнт – сервер»

Архітектура клієнт-сервер (рисунок 4.2) ділить програму на дві частини: «клієнт» та «сервер» [23]. Така програма реалізована в мережі і підключає клієнта до сервера. Серверна частина цієї архітектури забезпечує центральну функціональність: тобто будь-яка кількість клієнтів може підключитися до сервера і робити запит на виконання завдання. Сервер приймає ці запити, виконує необхідне завдання та повертає результати клієнту, якщо це доречно.

Якщо програмне забезпечення є одним монолітним елементом, то кожного разу, коли щось змінюється чи оновлюється, всю програму потрібно перерозподіляти заново. Поліпшенням може бути розділення програми на дві частини. Одна частина, клієнт, може надавати інтерфейс для користувачів і поширюватись на них. Інша частина може зберігатися і працювати на власному серверному апараті компанії.

Рисунок 4.2 – Схема архітектури клієнт-сервер

Клієнтська програма може відображати інформацію та використовуватись для передачі інформації на сервер для перекладу тексту.

Клієнтський застосунок або програмне забезпечення досить часто називають рівнем «презентації».

У моделі клієнт-сервер багато клієнтів можуть підключитися до серверної програми та запитувати інформацію, наприклад, про переклад тексту. Сервер повинен обробляти ці запити та надсилати відповідь клієнту, який створив запит, а не іншому клієнту. Поки мережа працює добре і сервер може не відставати від відповідей на всі запити, які вона отримує, такий «розділений» додаток надаватиме такий же рівень обслуговування, як і монолітний варіант. Цю просту архітектуру клієнт-сервер також називають «дворівневою архітектурою».

Загальний клієнт здійснює доступ до серверу за допомогою HTTP.

Клієнт, який поширюється на користувачів, може змінюватися, в той час як серверна частина може бути централізованим компонентом, який підтримує динамічні, глобальні дані послідовно та безпечно для організації та доступу користувачів та їх використання.

В завдання входить створення клієнту, який здійснює доступ до готового працюючого серверу Google за допомогою HTTP.

Висновки до розділу 4

У даному розділі було виконано моделювання архітектури за допомогою MVC, SOLID, розглянуто архітектуру клієнт-сервер.

Архітектура реалізована на шаблоні MVC. Дані програми, які призначені для користувацького інтерфейсу і керуючої логіки поділено на три окремих компоненти: модель, представлення і контролер таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Принципи SOLID — це п'ять пунктів проєктування класів та управління залежностями для об'єктно-орієнтованого програмування та проєктування. Якщо дотримуватися принципів SOLID, ви можете створити більш гнучкий та надійний код, який має більш високу можливість повторного використання.

Архітектура клієнт-сервер ділить програму на дві частини: «клієнт» та «сервер». Така програма реалізована в комп'ютерній мережі, яка підключає клієнта до сервера. Серверна частина цієї архітектури забезпечує центральну функціональність: тобто будь-яка кількість клієнтів може підключитися до сервера і вимагати, щоб він виконував завдання. Сервер приймає ці запити, виконує необхідне завдання та повертає будь-які результати клієнту, якщо це доречно.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

5.1 Переклад тексту

Перед відправкою на сервер генерується запит у вигляді HTTP URL, який складається з більш ніж 10 параметрів — вихідної та перекладаємої мов у форматі кода ISO—639—1, вихідного тексту, кодування символів, параметрів безпеки та інших параметрів.

В процесі перекладу сервер відправляє відповідь у форматі json, який необхідно правильно обробити за допомогою синтаксичного аналізу.

JSON або JavaScript Object Notation — мінімальний, читабельний формат для структуризації даних. Найбільш широко використовуваний формат даних для обміну даними в Інтернеті. Цей обмін даними може відбуватися між двома комп'ютерними програмами в різних географічних місцях або працювати в межах однієї апаратної машини. Він використовується головним чином для передачі даних між сервером та застосунком, як альтернатива XML. Перевага JSON — це легкочитаємий формат як для людини, так і для машини.

Для обробки json використовуємо регулярні вирази з класу java.util.regex.Pattern. Це Компільоване представлення регулярного виразу. Регулярний вираз, вказаний як рядок, спочатку повинен бути скомпільований в екземпляр цього класу. Отриманий Pattern потім може бути використаний для створення об'єкта Matcher, який може відповідати довільним послідовностям символів регулярного виразу. Matcher – клас java.util.regex.Matcher. Цей об’єкт виконує операції зі збігом послідовностей символів, визначений Pattern. Matcher створюється з шаблону шляхом виклику методу відповідності шаблону. Методи повертають булеве значення із зазначенням успіху чи невдачі. Matcher знаходить збіги у підмножині його входу, що називається регіоном. За замовчуванням область містить весь вхід відповідника.

«Вирізаємо» до першої коми строку, де міститься переклад. Далі очищуємо її від «сміття» — квадратних дужок, лапок та символів переносу. Переклад поміщається до масиву, де кожен його елемент – окрема строка, яку потрібно вивести з переносом з нової строки.

Рекомендований об’єм перекладаємого тексту – 2000 символів. Максимальний рекомендований об’єм – 3900. Google translate дозволяє розширити запит понад максимальний обєм навіть у декілька разів(!), але гарантії успішного перекладу відсутні. У випадку невдачі ми отримаємо помилку HTTP 413 (Request Entity Too Large Error — таблиця А.1), що означає, що клієнт відправив занадто великий запит на сервер.

В режимі автоперекладу під час набору тексту здійснюється відправка запиту після кожного натиснення клавіші. Якщо виконано більш ніж 100 запитів за короткий час (менш ніж за годину), сервер google translate заблокує ip-адресу! У відповідь на запит отримаємо помилку HTTP 429 (Too Many Requests — таблиця А.1), що означає, що клієнт відправив занадто багато запитів на сервер. Термін блокування ip-адреси – 12 годин.

Кнопки перекладу, розпізнання та синтезу реалізовані як toggle button – кнопки, що мають два стани – увімкнутий та вимкнутий. Така реалізація необхідна для можливості вимкнення запущених процесів, які вони запускають. Це особливо стосується процесу розпізнання мови, який не має обмежень в часі.

Усі toggle button запускають окремий потік – під час виконання будь-яких процесів програма не має підвисань, так як потік main() продовжує працювати з графічним інтерфейсом в подальшому.

5.2 Спливаюче вікно

При натисненні гарячих клавіш викликається переклад у маленькому віконці (рисунок 5.1) – переклад тексту виділеному мишкою у будь-якій програмі [25]. У створеному вікну є можливість синтезу мовлення.

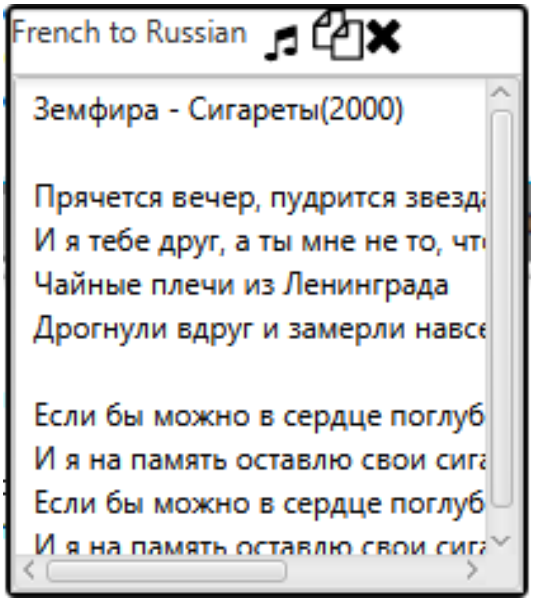

Рисунок 5.1 – Спливаюче вікно з перекладом

Нам потрібен клас java.awt.MouseInfo. Клас MouseInfo надає методи отримання інформації про мишу, такі як розташування вказівника миші та активності кнопок миші. Цей клас надає поточні координати знаходження миші — точка (java.awt.Point), що представляє розташування в x, у координатному просторі, які ми використовуємо при створенні екземпляра спливаючого вікна, розміщуючи його біля курсору.

Так як нам розміри перекладаємого тексту можуть бути різними, а розмір вікна незмінний, нам необхідний клас javafx.scene.control.ScrollPane. Цей елемент забезпечує вікно прокручення обрізаного контенту. Це дозволяє користувачеві прокручувати вміст безпосередньо або за допомогою смуг прокрутки.

Кожен екземпляр вспливаючого вікна містить текстове поле з класу javafx.scene.control.TextArea. На відмуну від звичайного текстового поля дане поле не матиме функції редагування тексту.

Вікно повинно бути компактним. Нам необхідне группування елементів, яке можливе завдяки класам javafx.scene.layout.HBox та javafx.scene.layout.VBox. HBox та VBox розміщує дочірні елементи в один горизонтальний та вертикальний ряд один за одним.

5.3 Маніпуляція з буфером обміну

Для реалізації перекладів до popup-повідомлень виникла необхідність маніпуляцій з буфером обміну – перед початком перекладу програма зберігає попередній стан буферу обміну, робить копіювання тексту до буферу та його переклад до popup-повідомлення. Після чого відбувається відновлення буферу обміну до попереднього стану.

Використаний клас javafx.scene.input.Clipboard представляє буфер обміну операційної системи, в який можуть бути розміщені дані, наприклад, під час операцій вирізання, копіювання та вставки.

У додатку є лише один екземпляр системного буфера обміну, тому цілком прийнятно зберігати посилання на нього десь зручно, якщо ви захочете.

Буфер обміну працює над концепцією наявності в буфер обміну одного концептуального елемента в будь-який час — хоча він може бути розміщений у буфері обміну в різних форматах. Вміст визначається у буфері обміну за допомогою методу setContent (java.util.Map). Щоразу, коли викликається setContent, будь-які попередні дані в буфері обміну очищаються та замінюються цим новим вмістом.

5.4 Клавіатурний перехоплювач

Так як виклик popup-повідомлення відбувається зі зміщенням фокусу до вікна з виділеним текстом, недостатньо реалізувати програмні гарячі клавіші. Гарячі клавіші працюють лише при фокусі на вікні розробленої програми. Виникла необхідність розробити клавіатурний перехоплювач, який відслідковує усі натисненні клавіші та виконує виклик подій при спрацюванні зазначених клавіш. Перехоплювач запускається разом з програмою у мінімізованому режимі, яка знаходиться в область повідомлень (system tray).

Коли ви набираєте або натискаєте будь-яку клавішу, натискається перемикач, який замикає ланцюг і дозволяє текти невеликій кількості струму. Процесор аналізує положення натиснутих клавіш і передає цю інформацію на комп'ютер, де вона надсилається на те, що називається «контролер клавіатури».

Контролер клавіатури — це пристрій, який відповідає за обробку даних отриманих від клавіатури комп'ютера. Працює контролер за таким принципом: при натисканні або відпусканні клавіші створюється байт — скан-код. В цей байт записується порядковий номер клавіші розміром 7 біт, а значення останнього біта, вказує на те, була клавіша натиснута або відпущена. Цей скан-код читає за допомогою порту 60h.

Цей контролер обробляє інформацію, що надсилається процесором клавіатури, і, у свою чергу, передає її в ОС. Потім ОС перевіряє ці дані, щоб проаналізувати, чи містять вони будь-які команди системного рівня, наприклад Ctrl + C, що є комбінацією для копіювання.

Якщо є такі команди системного рівня, комп'ютер виконує їх — якщо ні, то він пересилає інформацію до поточної програми. Потім програма перевіряє, чи натискання клавіш стосується команд у програмі, наприклад Ctrl + SPACE, що є комбінацією для виклику перекладу.

5.5 Значок в області оповіщень

Системний трей — це спеціалізована область робочого столу, де користувач може отримати доступ до даного поточно запущеного застосунку. В цьому є велика необхідності для реалізації швидкого перекладу, так як нам необхідна служба, яка буде постійно відслідковувати натискання клавіш для виклику перекладу.

Клас java.awt.SystemTray, представлений у Java 6 і вище, представляє системний трей для робочого столу. Доступ до системного трею можна викликати через статичний метод SystemTray.getSystemTray(). Але с початку

виконується перевірка методом isSupported(), щоб перевірити, чи підтримується системний трей в даній ОС. Якщо системний трей не підтримується на цій платформі, метод isSupported () повертає значення false.

Додаток не може створити інший екземпляр класу SystemTray – реалізовано шаблон «синглтон». Може існувати лише один екземпляр, створений у цьому класі, і цей екземпляр можна отримати за допомогою методу getSystemTray().

5.6 Синтез мовлення

Синтез мовлення потребує відправки тексту на сервер. Перед відправкою на сервер генерується запит у вигляді HTTP URL за допомогою класу java.net.URL. URL-адреса класу являє собою уніфікований локатор ресурсів, вказівник на «ресурс» — базу даних Google. Перед відправкою запиту перевіряється чи всі символи є дозволеними, відбувається автовизначення мови, передаються текст і параметри швидкості та висоту тону синтезу (рисунок 5.2), якщо текст занадто великий, то він розділяється на маленькі речення, кожне речення асинхронно відправляється на сервер за допомогою класа java.util.concurrent.ExecutorService [27]. Executor надає методи управління припиненням та методи, які можуть створити об’єкт Future для відстеження ходу одного або декількох асинхронних завдань.

Потім після повернення відповіді сервера кожен аудіозапис додається в загальний аудіострім за допомогою класу java.util.concurrent.Callable — потік, який повертає результат і може повернути виняток [26]. Інтерфейс Callable схожий на Runnable, оскільки обидва розроблені для класів, екземпляри яких потенційно виконуються іншим потоком. Однак Runnable не повертає результату і не може повернути очікуваний виняток. Всі дані додаються у інтерфейс java.util.Set – це колекція, яка не містить дублікатів елементів, яка реалізовує клас java.util.LinkedHashSet — таблиця хешування та реалізація лінкованого списку інтерфейсу Set із передбачуваним порядком

ітерації. Ця реалізація відрізняється від HashSet тим, що вона підтримує подвійно лінкований список, що проходить через усі його записи. Цей лінкований список визначає порядок ітерації, який є порядком, в якому елементи були вставлені в набір (порядок вставки). Порядок вставки не впливає, якщо елемент повторно вставлений у набір.

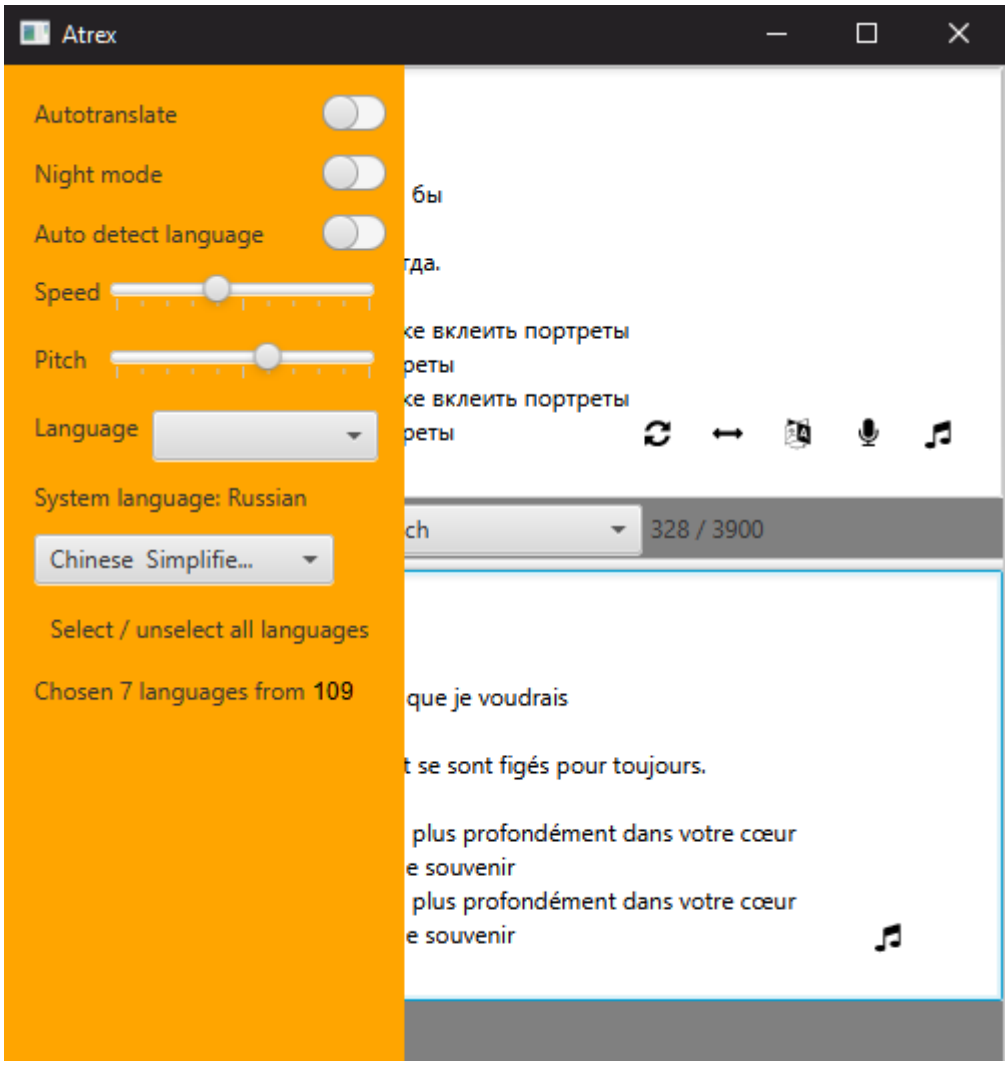

Рисунок 5.2 – Головне вікно застунку з відкритим слайд–меню з різними налаштування, в тому числі параметри швидкості та висоту тону синтезу

Результат виконання Callable є тип класу java.io.InputStream, який призначений для зчитування потоків необроблених байтів аудіоданих. Всі

InputStream потрапляють до типу java.util.concurrent.Future. Future представляє результат асинхронного обчислення.

Після запуску процесу синтезу кнопка фарбується в оранжевий колір (рисунок 5.3), а після вимкнення чи завершення — в попередній колір в залежності від вибраної кольорової гами.

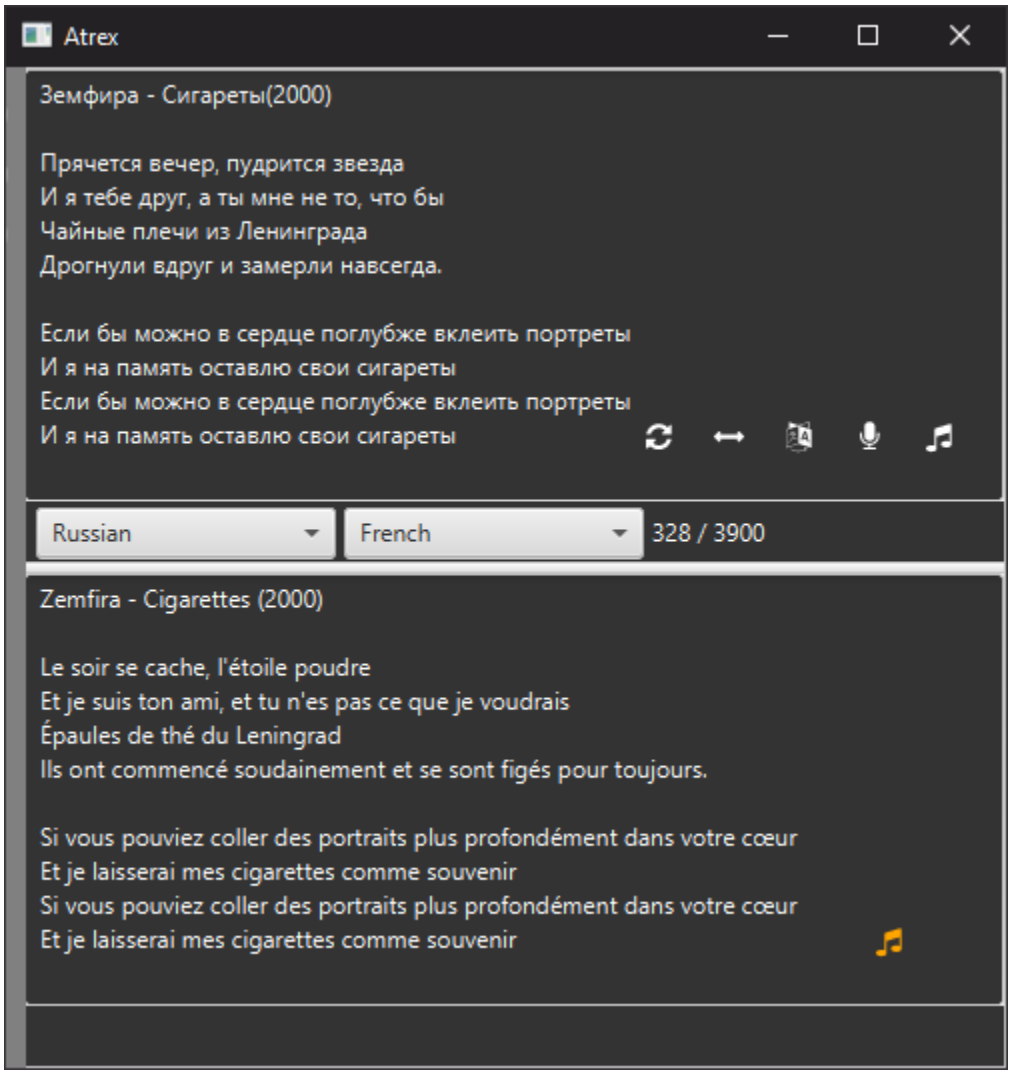

Рисунок 5.3 – Головне вікно застунку з запущеним процесом синтезом французької мови, кнопка якого пофарбована в помаранчевий

Для синтезу мовлення створюється та запускається екземпляр аудіо програвача javazoom, а сам програвач отримує параметр у вигляді аудіостріму.

5.7 Розпізнання мовлення

За допомогою патерна «спостерігача» виставляється прослушка на цільову лінію даних – це клас javax.sound.sampled.TargetDataLine. Цільова лінія даних — це тип DataLine, з якого можна читати аудіодані. У нашому випадку — лінія даних, яка отримує свої дані від пристрою захоплення аудіо — мікрофону. Інтерфейс TargetDataLine забезпечує метод зчитування захоплених даних із буфера цільової лінії даних. Застосунок повинен зчитувати дані з цільової лінії даних досить швидко, щоб запобігти переповненню буфера, так як це може спричинити розриви захоплених даних. Необхідно визначати кількость даних, що стоїть у черзі в буфері рядка даних. Якщо буфер переповнюється, найдавніші дані в черзі відкидаються та замінюються новими даними.

За допомогою URL HTTP дані відправляются на сервер типу javax.sound.sampled.AudioInputStream. Це аудіо вхідний потік — це вхідний потік із заданим звуковим форматом та довжиною. Довжина виражається у зразках кадрів, а не в байтах. Для зчитування певної кількості байтів із потоку передбачено кілька методів або не визначене число байтів. Потік аудіо входу відстежує останній прочитаний байт.

Створюються два потоки. Один потік відправляє дані мікрофона типу flac – клас net.sourceforge.javaflacencoder.FLACFileWriter. Ця бібліотека дозволяє ввімкнути flac-кодування, не вдаючись до використання JNI або конвертації файлів.

Інший потік повертає розпізнаний текст, який аналізується класом java.util.Scanner. Простий сканер тексту, який може робити граматичний аналіз примітивних типів та рядків, використовуючи регулярні вирази. Сканер розбиває свій вхід на токени за допомогою шаблону роздільника, який за замовчуванням відповідає пробілу. Потім отримані лексеми можуть бути перетворені у значення різних типів за допомогою різних методів.

5.8 Моделювання класів

У таблиці 5.1 наведено опис основних класів системи, у яких інкапсулюються методи, необхідні для коректної роботи застосунка. Діаграма класів зображена на ІА361.020БАК.005 Д2

Таблиця 5.1. – Опис головних класів системи

Назва класу	Опис
AtrextApp	Містить точку входу
ClipboardManager	Відповідає за обробку буферу обміну
Controller (додаток В)	Відповідає за управління взаємодією з користувачем, роботи з моделлю і вибору представлення для відображення
GoogleTranslationService	Відповідає за переклад, обробку тексту
KeyLogger	Відповідає за глобальні «гарячі» клавіші
Languages	Містить мови
MicrophoneAnalyzer	Відповідає за роботу з мікрофоном, розпізнання мовлення
PlayerService	Відповідає за процес синтез тексту
PopupText	Відповідає за миттєвий переклад у маленькому popup вікні
PropertiesXML	Відповідає за збереження користувацьких налаштувань
RecognizeService	Відповідає за процес розпізнання мовлення
SliderMenu	Відповідає за роботу зі слайд меню
SynthesisConverter	Відповідає за синтез тексту
TranslateService	Відповідає за процес перекладу
TrayView	Відповідає за значок в області оповіщень, запуск глобальних гарячих клавіш

Пакети а Java необхідні для групування класів, що взаємопов'язані між собою (рисунок 5.1). Аналогічне призначення мають директорії ОС з файлами в файловій системі.

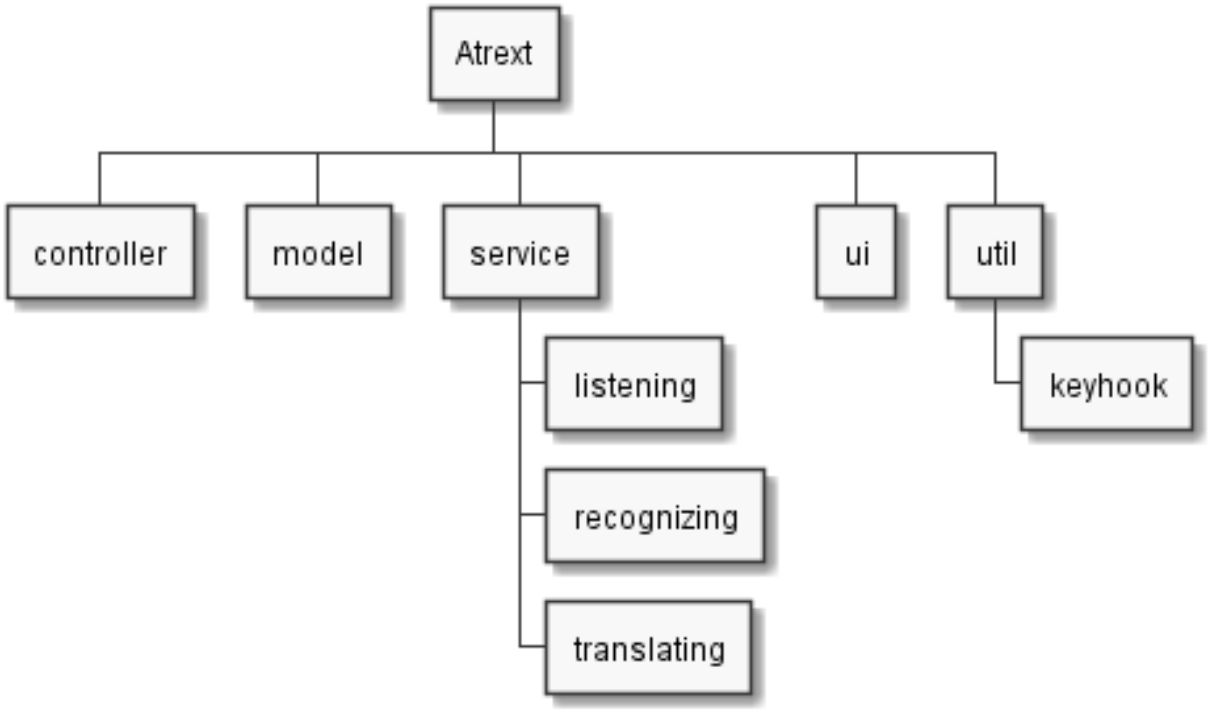

Рисунок 5.1 – Ієрархія пакетів проєкту

Пакет — це механізм інкапсуляції групи класів, підпакетів та інтерфейсів. Пакети використовуються для:

- запобігання конфліктування імен;
- полегшення пошуку, локалізації та використання класів, інтерфейсів, перерахувань та приміток;
- надання контрольованого доступу: protected, default мають доступ на рівні пакету. Захищений член доступний класам в одному пакеті та його підкласах. Член за замовчуванням (без будь-якого специфікатора доступу) доступний лише класам в одному пакеті;
- пакети розглядаються як інкапсулювання даних (або приховування даних).

Все, що потрібно нам зробити, — це помістити класи, що пов'язані, в пакети. Після цього ми пишемо класи імпортів зі створених пакетів і використовуємо їх в нашій програмі. Пакет — це контейнер групи пов'язаних класів, де деякі класи доступні, а інші зберігаються для внутрішніх цілей.

Ми можемо повторно використовувати наявні класи з пакетів стільки часу, скільки нам потрібно в нашій програмі.

Існує офіційна конвенція про іменування пакетів — пакети називаються у зворотному порядку іменен існуючих доменів або іншим способом, який надає унікальність класу з використанням імен організації, установи, відділу, назви застосунку, повного імені розробника. В нашому випадку ua.mylkodenys.kpi.fict.atrext.

Висновки до розділу 5

У даному розділі було виконано моделювання та детальна реалізація програмного забезпечення.

Переклад формується у вигляді запиту, який генерується у вигляді HTTP URL та складається з більш ніж 10 параметрів — вихідної та перекладаємої мов у форматі коду ISO 639—1, вихідного тексту, кодування символів, параметрів безпеки та інших параметрів. В процесі перекладу сервер відправляє відповідь у форматі json, який необхідно правильно обробити за допомогою синтаксичного аналізу.

При натисненні гарячих клавіш викликається переклад у маленькому віконці – переклад тексту виділеному мишкою у будь-якій програмі. У створеному вікну є можливість синтезу мовлення.

Для реалізації перекладів до popup-повідомлень виникла необхідність маніпуляцій з буфером обміну – перед початком перекладу програма зберігає попередній стан буферу обміну, робить копіювання тексту до буферу та його переклад до popup-повідомлення. Після чого відбувається відновлення буферу обміну до попереднього стану.

Так як виклик popup-повідомлення відбувається зі зміщенням фокусу до вікна з виділеним текстом, недостатньо реалізувати програмні гарячі клавіші. Гарячі клавіші працюють лише при фокусі на вікні розробленої програми. Виникла необхідність розробити клавіатурний перехоплювач, який відслідковує усі натисненні клавіші та виконує виклик подій при спрацюванні зазначених клавіш. Перехоплювач запускається разом з програмою у мінімізованому режимі, яка знаходиться в область повідомлень (system tray).

Синтез мовлення потребує тексту відправки на сервер. Перед відправкою запиту перевіряється чи всі символи є дозволеними, відбувається автовизначення мови, передаються текст і параметри швидкості та висоту тону синтезу, якщо текст занадто великий, то він розділяється на маленькі речення, кожне речення асинхронно відправляється на сервер за допомогою класа java.util.concurrent.ExecutorService.

За допомогою патерна «спостерігача» виставляється прослушка на цільову лінію даних – це тип DataLine, з якого можна читати аудіодані. У нашому випадку — лінія даних, яка отримує свої дані від пристрою захоплення аудіо — мікрофону.

Системний трей — це спеціалізована область робочого столу, де користувач може отримати доступ до даного поточно запущеного застосунку. В цьому є велика необхідності для реалізації швидкого перекладу, так як нам необхідна служба, яка буде постійно відслідковувати натискання клавіш для виклику перекладу.

ВИСНОВКИ

Метою даного проєкту було розроблення системи автоматизованого перекладу тексту. Даний застосунок дозволяє виконувати не тільки переклад, а також синтез мовлення, розпізнання мовлення через мікрофон та швидкий переклад виділеного тексту при натисненні «гарячої» клавіші — спрощення отримання перекладу та організація миттєвого перекладу тексту за гарячою клавішею з будь-якого місця – блокноту, документа, книги, чату.

При огляді технологій Google в поєднанні з оглядом застосунків зі схожою тематикою, які використовують сервіс Google були сформовані чіткі та основні вимоги до інтерфейсу, архітектури, можливостей застосунку розробленого у даному проєкті.

Актуальність теми підтверджується тим, що проект був розроблений на замовлення сторонньої організації (додаток Г). Усі функції застосунку є актуальними, та такими, що необхідні у використанні користувачами.

Користувацький інтерфейс був розроблений виходячи з того, що він повинен бути простим та містити нейтральні кольори, які не втомлюють користувача та забезпечують приємне враження від користування застосунком.

Архітектура застосунку реалізована на шаблоні MVC та архітектурі клієнт-сервер. Така програма реалізована в комп'ютерній мережі, яка підключає клієнта до сервера. Дані програми поділено на три окремих компоненти: модель, представлення і контролер.

Даний проєкт можна розширювати у майбутньому, а саме додати можливість розпізнання тексту з фото, додати більш широкий переклад – синоніми та приклади використання перекладених слів, створити мобільний додаток під Android та IOS.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дорогой Я.Ю., Дорошенко К.С., Репнікова Н.Б., Юрчук Л.Ю. Дипломний проєкт бакалавра. Розробка, оформлення, захист. Київ: НТТУ КПІ, 2020. 77 с.
2. Как работает переводчик языка Google? URL : https://nnnn.su/24/02/2020/11135/kak-rabotaet-perevodchik-yazyka-google.html (дата звернення: 20.12.2019).
3. Как работает нейросеть Google Translate. URL : https://www.cossa.ru/trends/196086/ (дата звернення: 20.12.2019).
4. Google преодолевает барьер между человеческим и машинным переводом. URL : https://www.reg.ru/blog/google-preodolevaet-barer-mezhdu-chelovecheskim-i-mashinnym-perevodom/ (дата звернення: 20.12.2019).
5. How does Google Voice work, and should you use it? URL : https://www.tomsguide.com/news/how-does-google-voice-work (дата звернення: 20.12.2019).
6. Google Translate Client. URL : https://habr.com/ru/post/61608/ (дата звернення: 30.12.2019).
7. Переводчик Google Translate подключили к нейросети. URL : https://habr.com/ru/post/397959/ (дата звернення: 20.12.2019).
8. Google открывает API для распознавания речи на 80 языках. URL : https://habr.com/ru/news/t/392205/ (дата звернення: 20.12.2019).
9. Client for Google Translate 6.2.620 Десктопный клиент гугл переводчика для windows. URL : https://apps24.org/windows/ofis/perevodchiki/client-for-google-translate (дата звернення: 30.12.2019).
10. Dictionary.NET 9.6.7009 Бесплатный мультиязычный язычный переводчик для компьютера. URL : https://apps24.org/windows/ofis/perevodchiki/dictionary-net (дата звернення: 30.12.2019).

11. QTranslate Бесплатная программа для перевода текстов с различных языков. URL : https://apps24.org/windows/ofis/perevodchiki/qtranslate (дата звернення: 30.12.2019).
12. Dicter 3.83 Мощный переводчик с поддержкой более 100 языков. URL : https://apps24.org/windows/ofis/perevodchiki/dicter (дата звернення: 30.12.2019).
13. Диаграмма прецедентов (вариантов использования) UML. URL : https://planerka.info/item/diagramma-precedentov-variantov-ispolzovaniya-uml/ (дата звернення: 22.01.2020).
14. Призначення і склад діаграми варіантів використання. URL : https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/tema12_2 (дата звернення: 22.01.2020).
15. Eclipse IDE. URL : https://ru.bmstu.wiki/Eclipse_IDE (дата звернення: 15.03.2020).
16. Using Scene Builder with NetBeans IDE. URL : https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm (дата звернення: 15.03.2020).
17. Новое в Java 8. URL : https://habr.com/ru/post/216431/ (дата звернення: 15.03.2020).
18. Учебник по JavaFX: FXML и SceneBuilder. URL : https://habr.com/ru/post/474982/ (дата звернення: 17.03.2020).
19. Введение в Java FX. URL : https://javarush.ru/groups/posts/2560-vvedenie-v-java-fx (дата звернення: 17.03.2020).
20. ControlsFX Features. URL : http://fxexperience.com/controlsfx/features/ (дата звернення: 17.03.2020).
21. Что такое MVC: рассказываем простыми словами. URL : https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami (дата звернення: 20.02.2020).

22. Принципы SOLID, о которых должен знать каждый разработчик. URL: https://medium.com/webbdev/solid-4ffc018077da (дата звернення: 20.02.2020).
23. Клиент-серверная архитектура в картинках. URL : https://habr.com/ru/post/495698/ (дата звернення: 20.02.2020).
24. Часть 7: Развёртывание | Учебник по JavaFX. URL : https://code.makery.ch/ru/library/javafx-tutorial/part7/ (дата звернення: 17.03.2020).
25. How to Create a Pop Up Window in JavaFX. URL : http://www.learningaboutelectronics.com/Articles/How-to-create-a-pop-up-window-in-JavaFX.php (дата звернення: 17.03.2020).
26. Concurrent интерфейсы Callable, Future. URL : http://java-online.ru/concurrent-callable.xhtml (дата звернення: 17.03.2020).
27. Java ExecutorService. URL : http://tutorials.jenkov.com/java-util-concurrent/executorservice.html (дата звернення: 17.03.2020).
28. Коды ответа HTTP. URL : https://developer.mozilla.org/ru/docs/Web/HTTP/Status (дата звернення: 30.03.2020).
29. Codes for the Representation of Names of Languages. URL : http://www.loc.gov/standards/iso639-2/php/English_list.php (дата звернення: 30.03.2020).
30. Нейросетевой синтез речи с помощью архитектуры Tacotron 2, или «Get alignment or die tryin'«. URL : https://habr.com/ru/company/nix/blog/436312/ (дата звернення: 20.12.2019).
31. Переводчик Google Translate подключили к нейросети. URL : https://www.pvsm.ru/ii/193397 (дата звернення: 20.12.2019).
32. The Long Short-Term Memory cell. URL : https://www.researchgate.net/figure/The-Long-Short-Term-Memory-cell-The-figure-shows-an-LSTM-cell-with-a-net-input-a_fig2_220245121 (дата звернення: 20.12.2019).

ДОДАТОК А

Таблиця А.1 — Список кодів стану HTTP сервера google translate для вказівки помилок з боку клієнта [28]

Код	Описання відповіді сервера
408 Request Timeout	час очікування сервером передачі від клієнта минув. Через деякий час сервер може закрити з'єднання зі свого боку, щоб дати можливість іншим клієнтам зробити запит.
409 Conflict	запит не може бути виконаний через конфліктного звернення до ресурсу. Таке можливо, наприклад, коли два клієнта намагаються змінити ресурс
410 Gone	ресурс раніше був за вказаною URL, але був вилучений і тепер недоступний.
413 Payload Too Large	запит занадто великого розміру. Сервер може закрити з'єднання, щоб припинити подальшу передачу запиту.
414 URI Too Long	URI занадто довгий. Таку помилку можна спровокувати, наприклад, коли клієнт намагається передати довгі параметри
415 Unsupported Media Type	з якихось причин сервер відмовляється працювати з вказаним типом даних при цьому методі.
421 Misdirected Request	запит був перенаправлений на сервер, не здатний дати відповідь.
422 Unprocessable Entity	сервер успішно прийняв запит, може працювати із зазначеним видом даних (наприклад, в тілі запиту знаходиться XML-документ, який має вірний синтаксис), проте є якась логічна помилка, через яку неможливо провести операцію над ресурсом.

Продовження таблиці А.1

426 Upgrade Required	сервер вказує клієнту на необхідність оновити протокол.
429 Too Many Requests	клієнт спробував відправити занадто багато запитів за короткий час, що може вказувати, наприклад, на спробу DDoS-атаки.
431 Request Header Fields Too Large	Перевищено допустима довжина заголовків.
434 Requested host unavailable	запитується адреса недоступна
449 Retry With	повертається сервером, якщо для обробки запиту від клієнта надійшло недостатньо інформації.
451 Unavailable For Legal Reasons	доступ до ресурсу закритий з юридичних причин, наприклад, на вимогу органів державної влади або на вимогу правовласника в разі порушення авторських прав.

ДОДАТОК Б

Таблиця Б.1 — Список мовних кодів ISO 639—1 [29]

№з/п	Код	Назва мови	Англійська назва	підтримується Google Translate
1	ab	Абхазька	Abkhazian	
2	av	Аварська	Afar	
3	ae	Авестійська	Avestan	
4	az	Азербайджанська	Azerbaijani	так
5	ay	Аймара	Aymara	
6	ak	Акан	Akan	
7	sq	Албанська	Albanian	так
8	am	Амхара	Amhara	так
9	en	Англійська	English	так
10	ar	Арабська	Arabic	так
11	an	Арагонська	Aragonese	
12	as	Ассамська	Assamese	
13	aa	Афарська	Afar	
14	af	Африкаанс	Afrikaans	так
15	bm	Бамбара	Bambara	
16	eu	Баскська	Basque	так
17	ba	Башкирська	Bashkir	
18	bn	Бенгальська	Bengali	так
19	be	Білоруська	Belarusian	так
20	my	Бірманська	Burmese	так
21	bi	Біслама	Bislama	
22	bh	Біхарі	Bihari	

Продовження таблиці Б.1

23	bg	Болгарська	Bulgarian	так
24	bs	Боснійська	Bosnian	так
25	br	Бретонська	Breton	
26	nb	Букмол	Bokmal	
27	cy	Валлійська	Welsh	так
28	wa	Валлонська	Walloon	
29	ve	Венда	Venda	
30	vi	В'єтнамська	Vietnamese	так
31	hy	Вірменська	Armenian	так
32	vo	Волапюк	Volapyuk	
33	wo	Волоф	Wolof	
34	haw	Гавайська	Hawaiian	так
35	ht	Гаїтянська	Haitian	так
36	gl	Галісійська	Galician	так
37	hz	Гереро	Guerrero	
38	hi	Гінді	Hindi	так
39	ho	Гірі-моту	Giri-motu	
40	kl	Гренландська	Greenlandic	
41	el	Грецька	Greek	так
42	ka	Грузинська	Georgian	так
43	gn	Гуарані	Guarani	
44	gu	Гуджараті	Gujarati	так
45	da	Данська	Danish	так
46	dz	Дзонг-ке	Dzong-ke	
47	dv	Дівехі	Divehi	
48	ee	Еве	Here	
49	eo	Есперанто	Esperanto	так

Продовження таблиці Б.1

50	et	Естонська	Estonian	так
51	fy	Західно-фризька	West Frisian	так
52	zu	Зулу	Zulu	так
53	he	Іврит	Hebrew	так
54	ig	Ігбо	Igbo	так
55	io	Ідо	Ido	
56	id	Індонезійська	Indonesian	так
57	ia	Інтерлінгва	Interlingua	
58	iu	Інуктітут	Inuktitut	
59	ik	Інупіак	Inupiac	
60	ga	Ірландська	Irish	так
61	is	Ісландська	Icelandic	так
62	es	Іспанська	Spanish	так
63	it	Італійська	Italian	так
64	yi	Їдиш	Yiddish	так
65	yo	Йоруба	Yoruba	так
66	kk	Казахська	Kazakh	так
67	kn	Каннада	Canada	так
68	kr	Канурі	Kanuri	
69	ca	Каталонська	Catalan	так
70	ks	Кашмір	Kashmir	
71	qu	Кечуа	Quechua	
72	ky	Киргизька	Kyrgyz	так
73	zh	Китайська	Chinese	так (спрощена та традиційна)
74	ki	Кікуйю	Kikuyu	

Продовження таблиці Б.1

75	rw	Кінаруанда (Руандійська)	Kinaruanda	Так
76	rn	Кірундійська	Kirundian	
77	kv	Комі	Komi	
78	kg	Конголезька	Congolese	
79	ko	Корейська	Korean	так
80	kw	Корнська	Cornish	
81	co	Корсиканська	Corsican	так
82	xh	Коса	Коса	так
83	kj	Кунама	Kunama	
84	ku	Курдська	Kurdish	так
85	km	Кхмерська	Khmer	так
86	lo	Лаоська	Lao	так
87	la	Латинська	Latin	так
88	lv	Латиська	Latvian	так
89	lt	Литовська	Lithuanian	так
90	li	Лімбурзька	Limburg	
91	ln	Лінґала	Lingala	
92	lu	Луба-катанга	Luba-katanga	
93	lg	Луганда	Luganda	
94	lb	Люксембурзька	Luxembourgish	так
95	mk	Македонська	Macedonian	так
96	mg	Малагасійська	Malagasy	так
97	ms	Малайська	Malay	так
98	ml	Малаялам	Malayalam	так
99	mt	Мальтійська	Maltese	так
100	mi	Маорі	Maori	так

Продовження таблиці Б.1

101	mr	Маратхі	Marathi	так
102	mh	Маршальська	Marshal	
103	gv	Менська	Minsk	
104	cr	Мова Крі	Cree language	
105	mn	Монгольська	Mongolian	так
106	nv	Навахо	Navajo	
107	na	Науру	Nauru	
108	ng	Ндонга	Ndonga	
109	ne	Непальська	Nepali	так
110	nl	Нідерландська	Dutch	так
111	de	Німецька	German	так
112	no	Норвезька	Norwegian	так
113	ny	Ньянджа	Nyanja	так
114	nn	Нюношк	Нюношк	
115	oj	Оджибве	Ojibwe	
116	oc	Окситанська	Occitan	
117	ie	Окциденталь	Occidental	
118	or	Орія	Oriya	так
119	om	Орома	Oroma	
120	os	Осетинська	Ossetian	
121	pi	Палі	Fields	
122	pa	Пенджабі	Punjabi	так
123	fa	Перська	Persian	так
124	nr	Південна ндебеле	Southern Ndebele	
125	nd	Північна ндебеле	Northern Ndebele	
126	se	Північно-саамська	North Sami	
127	pl	Польська	Polish	так

Продовження таблиці Б.1

128	pt	Португальська	Portuguese	так
129	ps	Пушту	Pashto	так
130	rm	Ретороманська	Romansh	
131	ru	Російська	Russian	так
132	ro	Румунська; Молдовська	Romanian; Moldovan	так
133	ry	Русинська	Rusyn	
134	sm	Самоанська	Samoan	так
135	sg	Санго	Sango	
136	sa	Санскрит	Sanskrit	
137	sc	Сардинська	Sardinian	
138	ss	Свазі	Swaziland	
139	tn	Свана	Swan	
140	sw	Суахілі	Swahili	так
141	ceb	Себуанська	Cebuano	так
142	sh	Сербо-хорватська	Serbo-Croatian	
143	sr	Сербська	Serbian	так
144	st	Сесото	Sesotho	так
145	si	Сингальська	Sinhala	так
146	ii	Сичуань	Sichuan	
147	sd	Сіндгі	Sindgi	так
148	sk	Словацька	Slovak	так
149	sl	Словенська	Slovenian	так
150	so	Сомалі	Somalia	так
151	su	Сунданська	Sundanese	так
152	tl	Тагалог	Tagalog	так
153	tg	Таджицька	Tajik	так

Продовження таблиці Б.1

154	ty	Таїтянська	Tahitian	
155	th	Тайська	Thai	так
156	ta	Тамільська	Tamil	так
157	tt	Татарська	Tatar	так
158	te	Телугу	Telugu	так
159	bo	Тибетська	Tibetan	
160	ti	Тигрінья	Tigrinya	
161	to	Тонганська	Tongan	
162	ts	Тсонґа	Tsonga	
163	tr	Турецька	Turkish	так
164	tk	Туркменська	Turkmen	так
165	hu	Угорська	Hungarian	так
166	uz	Узбецька	Uzbek	так
167	ug	Уйгурська	Uighur	так
168	uk	Українська	Ukrainian	так
169	ur	Урду	Urdu	так
170	fo	Фарерська	Faroese	
171	fj	Фіджі	Fiji	
172	fi	Фінська	Finnish	так
173	fr	Французька	French	так
174	ff	Фула	Fula	
175	ha	Хауса	House	так
176	hmn	Хмонг	Hmong	так
177	hr	Хорватська	Croatian	так
178	cu	Церковнослов’янська	Church Slavonic	
179	ch	Чаморро	Chamorro	
180	tw	Чві	Chvi	

Продовження таблиці Б.1

181	cs	Чеська	Czech	так
182	ce	Чеченська	Chechen	
183	za	Чжуан	Zhuang	
184	cv	Чуваська	Chuvash	
185	sv	Шведська	Swedish	так
186	sn	Шишона	Shishona	так
187	gd	Шотландська	Scottish	так
188	jv	Яванська	Javanese	так
189	ja	Японська	Japanese	так

ДОДАТОК В

Автоматизована система перекладу тексту

Основні компоненти системи

Текст програми

Аркушів 14

Київ – 2020 року

Клас Controller

```
package ua.mylkodenys.kpi.fict.atrext;

import org.controlsfx.control.CheckComboBox;
import org.controlsfx.control.ToggleSwitch;

import de.jensd.fx.glyphs.fontawesome.FontAwesomeIconView;
import javafx.application.Platform;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ListChangeListener;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.control.SplitPane;
import javafx.scene.control.TextArea;
import javafx.scene.control.ToggleButton;
import javafx.scene.control.ToolBar;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import ua.mylkodenys.kpi.fict.atrext.model.Languages;
import ua.mylkodenys.kpi.fict.atrext.service.listening.PlayerService;
import ua.mylkodenys.kpi.fict.atrext.service.recognizing.RecognizeService;
import ua.mylkodenys.kpi.fict.atrext.service.translating.TranslateService;
import ua.mylkodenys.kpi.fict.atrext.ui.SliderMenu;
import ua.mylkodenys.kpi.fict.atrext.ui.TrayView;
import ua.mylkodenys.kpi.fict.atrext.util.KeyboardKeys;
import ua.mylkodenys.kpi.fict.atrext.util.PropertiesXML;

public class Controller {

    @FXML
    private ComboBox<String> inputLanguageList;
```

```
@FXML
private TextArea inputTextArea;

@FXML
private ToggleSwitch autoDetectLanguageSwitch;

@FXML
private ComboBox<String> outputLanguageList;

@FXML
private TextArea outputTextArea;

@FXML
private ToggleSwitch autotranslateSwitch;

@FXML
private FontAwesomeIconView recognizeVoiceIcon;

@FXML
private ToggleButton recognizeVoiceButton;

@FXML
private FontAwesomeIconView listenInputTextIcon;

@FXML
private ToggleButton listenInputTextButton;

@FXML
private FontAwesomeIconView listenOutputTextIcon;

@FXML
private ToggleButton listenOutputTextButton;

@FXML
private FontAwesomeIconView translateIcon;

@FXML
private ToggleButton translateButton;

@FXML
private Button swapLanguagesButton;

@FXML
```

```
private SplitPane splitPane;

@FXML
private AnchorPane paneRoot;

@FXML
private Pane sliderMenu;

@FXML
private ToggleSwitch nightModeSwitch;

@FXML
private Slider speedSlider;

@FXML
private Slider pitchSlider;

@FXML
private ToolBar toolbar1;

@FXML
private ToolBar toolbar2;

@FXML
private Button cleanTextButton;

@FXML
private Label systemLanguageLabel;

@FXML
public CheckComboBox<String> chosenLanguagesCheckComboBox;

@FXML
public Button clearTextButton;

@FXML
public Label characterLimitInputText;

@FXML
public Label countChosenLanguagesLabel;

@FXML
public ToggleButton selectLanguagesButton;
```

```
@SuppressWarnings(«unused»)
private AtrextApp mainStage;

private TranslateService translateService = new TranslateService();

public PlayerService playerService = new PlayerService();

private RecognizeService recognizeService = new RecognizeService();

private Languages languages = new Languages();
private Stage stage;

private PropertiesXML prop = new PropertiesXML();

private KeyboardKeys keyb = new KeyboardKeys();

private TrayView trayv = new TrayView();

private SliderMenu sliderm = new SliderMenu();

public void clearTextInTextAreas() {
        inputTextArea.clear();
        outputTextArea.clear();
}

public boolean getAutoDetectLanguageSwitch() {
        return autoDetectLanguageSwitch.isSelected();
}

public boolean getAutotranslateSwitch() {
        return autotranslateSwitch.isSelected();
}

public String getInputLanguage() {
        if (getAutoDetectLanguageSwitch())
                return «Autodetect»;
        else
                return getInputLanguageWithoutAutoDetect();
}

public String getInputLanguageList() {
```

```
            return
inputLanguageList.getSelectionModel().selectedItemProperty().getValue();
        }

        public String getInputLanguageWithoutAutoDetect() {
            return getInputLanguageList();
        }

        public String getInputTextArea() {
            return inputTextArea.getText();
        }

        public ToggleButton getListenInputTextButton() {
            return listenInputTextButton;
        }

        public Stage getMainScene() {
            Stage stage = (Stage) paneRoot.getScene().getWindow();
            return stage;
        }

        public boolean getNightModeSwitch() {
            return nightModeSwitch.isSelected();
        }

        public String getOutputLanguage() {
            return getOutputLanguageList();
        }

        public String getOutputLanguageList() {
            return
outputLanguageList.getSelectionModel().selectedItemProperty().getValue();
        }

        public String getOutputTextArea() {
            return outputTextArea.getText();
        }

        public AnchorPane getPaneRoot() {
            return paneRoot;
        }

        public double getPitch() {
```

```
        return getPitchSlider();
    }

    public double getPitchSlider() {
        return pitchSlider.getValue();
    }

    public Pane getSliderMenu() {
        return sliderMenu;
    }

    public double getSpeed() {
        return getSpeedSlider();
    }

    public double getSpeedSlider() {
        return speedSlider.getValue();
    }

    public Stage getStage() {
        return stage;
    }

    public Button getSwapLanguagesButton() {
        return swapLanguagesButton;
    }

    public Label getSystemLanguageLabel() {
        return systemLanguageLabel;
    }

    public String getTextToTranslate() {
        return getInputTextArea();
    }

    @FXML
    public void initialize() {
        initialListeners();
        initialLanguagesList();
        sliderm.initialSliderMenu(this);
        trayv.initialSavePropertiesOnClose(prop, this);
        prop.initialProperties(this);
        setViewColor();
```

```
            loadChosenLanguages();
            trayv.initialTray(this);
            initialFocusListen();
        }

        public void runRecognizeVoice() {
            if (!recognizeVoiceButton.isSelected())
                recognizeService.stopSpeechRecognition();
            else
                (new Thread(() —> {
                    recognizeVoiceIcon.setStyle(«—fx—fill: orange;»);
                    recognizeVoiceButton.setSelected(true);
                    StringProperty stringProperty = new
SimpleStringProperty();
                    try {
                        stringProperty.addListener((ObservableValue<?
extends String> observable, String oldValue,
                                                    String newValue) —>
Platform.runLater(() —> setTextToTranslate(stringProperty.get())));
                        recognizeService.startService(stringProperty);
                    } catch (Exception e) {

        outputTextArea.appendText(System.getProperty(«line.separator»));

        outputTextArea.appendText(System.getProperty(«line.separator»));
                        outputTextArea.appendText(«ERROR! This
language doesn't support recognize!»);
                    } finally {
                        recognizeVoiceIcon.setStyle(«—fx—fill: « +
restoreButtonColor() + «;»);
                        recognizeVoiceButton.setSelected(false);
                    }
                })).start();

        }

        public void runTranslate() {
            Platform.runLater(() —> {
                translateButton.setSelected(true);
                translateIcon.setStyle(«—fx—fill: orange;»);
                try {
                    setTranslatedText(
```

```
        translateService.startService(getInputLanguage(), getOutputLanguage(),
getTextToTranslate()));
                        } catch (Exception e) {

        outputTextArea.appendText(System.getProperty(«line.separator»));

        outputTextArea.appendText(System.getProperty(«line.separator»));
                                outputTextArea.appendText(«ERROR of translating!»);
                        } finally {
                                translateIcon.setStyle(«—fx—fill: « +
restoreButtonColor() + «;»);
                                translateButton.setSelected(false);
                        }
                });

        }

        String [] stringPopupTranslate = null;

        public String [] runPopupTranslate(String text) {
                try {
                        stringPopupTranslate =
translateService.startService(«Autodetect», getOutputLanguage(), text);
                } catch (Exception e) {
                        stringPopupTranslate [0] = «ERROR of translating!»;
                }
                return stringPopupTranslate;

        }

        public void setAutoDetectLanguageSwitch(boolean b) {
                autoDetectLanguageSwitch.setSelected(b);
        }

        public void setAutotranslateSwitch(boolean b) {
                autotranslateSwitch.setSelected(b);
        }

        public void setChosenLanguages(int... is) {

        chosenLanguagesCheckComboBox.getCheckModel().checkIndices(is);
                System.out.println(is);
```

```
    }

    public void setInputLanguage(String s) {
        inputLanguageList.getSelectionModel().select(s);
    }

    public void setInputText(String s) {
        inputTextArea.setText(s);
    }

    public void setMainStage(AtrextApp mainStage) {
        this.mainStage = mainStage;
    }

    public void setNightModeSwitch(boolean b) {
        nightModeSwitch.setSelected(b);
    }

    public void setOutputLanguage(String s) {
        outputLanguageList.getSelectionModel().select(s);
    }

    public void setOutputText(String s) {
        outputTextArea.setText(s);
    }

    public void setPitchSlider(double d) {
        pitchSlider.setValue(d);
    }

    public void setSpeedSlider(double d) {
        speedSlider.setValue(d);
    }

    public void setStage(Stage s) {
        this.stage = s;
    }

    public void setSystemLanguageLabel(Label systemLanguageLabel) {
        this.systemLanguageLabel = systemLanguageLabel;
    }

    public void setTranslatedText(String [] ss) {
```

```
            outputTextArea.setText(«««);
            for (String str : ss) {
                outputTextArea.appendText(str);

        outputTextArea.appendText(System.getProperty(«line.separator»));
            }
        }

        public void showStage() {
            if (stage == null)
                return;
            stage.show();
            stage.toFront();
        }

        public void hideStage() {
            if (stage == null)
                return;
            stage.hide();
        }

        private void initialLanguagesList() {
            chosenLanguagesCheckComboBox.getItems()

        .addAll(FXCollections.observableArrayList(languages.getLanguagesList().k
eySet()));
        }

        private void initialListeners() {

        translateButton.disableProperty().bind(autotranslateSwitch.selectedProperty(
));
            clearTextButton.setOnAction(a —> clearTextInTextAreas());
            swapLanguagesButton.setOnAction(a —>
swapLanguagesInComboBoxes());
            translateButton.setOnAction(a —> runTranslate());
            splitPane.setOnKeyReleased((KeyEvent event) —>
keyb.getShortcuts(this, event));
            recognizeVoiceButton.setOnAction(a —> runRecognizeVoice());
            selectLanguagesButton.setOnAction(a —> selectLanguages());
            inputTextArea.textProperty().addListener((observable, oldValue,
newValue) —> runAutoTranslate());
```

```
            inputLanguageList.getSelectionModel().selectedItemProperty()
                    .addListener((observable, oldValue, newValue) —>
runAutoTranslate());
            outputLanguageList.getSelectionModel().selectedItemProperty()
                    .addListener((observable, oldValue, newValue) —>
runAutoTranslate());
            listenInputTextButton
                    .setOnMouseClicked(a —>
listenText(getInputTextArea(), listenInputTextIcon, listenInputTextButton));
            listenOutputTextButton
                    .setOnMouseClicked(a —>
listenText(getOutputTextArea(), listenOutputTextIcon, listenOutputTextButton));

            nightModeSwitch.selectedProperty().addListener(new
ChangeListener<Boolean>() {
                @Override
                public void changed(final ObservableValue<? extends
Boolean> ov, final Boolean t1, final Boolean t2) {
                    setViewColor();
                }
            });
            inputTextArea.textProperty().addListener(new
ChangeListener<String>() {
                @Override
                public void changed(final ObservableValue<? extends String>
s, final String oldValue,
                                    final String newValue) {

        characterLimitInputText.setText(String.valueOf(inputTextArea.getLength())
+ « / 3900»);
                }
            });


        chosenLanguagesCheckComboBox.getCheckModel().getCheckedItems().ad
dListener(new ListChangeListener<String>() {
                public void onChanged(ListChangeListener.Change<? extends
String> s) {
                    countChosenLanguagesLabel.setText(«Chosen «
                            + String.valueOf(

        (chosenLanguagesCheckComboBox.getCheckModel().getCheckedItems().to
Array().length — 1))
```

```
                                        + « languages from « +
(chosenLanguagesCheckComboBox.getItems().size() — 1));
                                loadChosenLanguages();
                                chooseDefaultLanguages();

                        }

                        private void chooseDefaultLanguages() {
                                if
(!chosenLanguagesCheckComboBox.getCheckModel().isChecked(«English»))

        chosenLanguagesCheckComboBox.getCheckModel().check(«English»);
                                if
(!chosenLanguagesCheckComboBox.getCheckModel().isChecked(«Russian»))

        chosenLanguagesCheckComboBox.getCheckModel().check(«Russian»);
                                }
                        });
                }

                public void initialFocusListen() {
                        startGlobalKeyHook();
                        Platform.runLater(() —> {
                                getMainScene().focusedProperty().addListener(new
ChangeListener<Boolean>() {
                                @Override
                                public void changed(ObservableValue<? extends
Boolean> observable, Boolean oldValue, Boolean newValue) {
                                        if (getMainScene().isFocused()) {
                                                System.out.println(«FOCUS»);
                                                trayv.global.stopGlobalHook();
                                        } else {
                                                startGlobalKeyHook();
                                        }
                                }
                        });
                });
        }

        private void startGlobalKeyHook() {
                (new Thread(() —> {
                        System.out.println(«NOT FOCUS»);
                        trayv.global.startService();
```

```
        })).start();
    }

    private void listenText(String Text, FontAwesomeIconView listenIcon,
ToggleButton listenButton) {
        if (!listenButton.isSelected())
            playerService.stopService();
        else
            (new Thread(() —> {
                listenIcon.setStyle(«—fx—fill: orange;»);
                listenButton.setSelected(true);
                try {
                    playerService.startService(Text, getSpeed(),
getPitch());
                } catch (Exception e) {

    outputTextArea.appendText(System.getProperty(«line.separator»));

    outputTextArea.appendText(System.getProperty(«line.separator»));
                    outputTextArea.appendText(«ERROR! This
language doesn't support synthesis!»);
                } finally {
                    listenIcon.setStyle(«—fx—fill: « +
restoreButtonColor() + «;»);
                    listenButton.setSelected(false);
                }
            })).start();
    }

    private void loadChosenLanguages() {
        inputLanguageList.setItems(

    FXCollections.observableArrayList(chosenLanguagesCheckComboBox.get
CheckModel().getCheckedItems()));
        outputLanguageList.setItems(

    FXCollections.observableArrayList(chosenLanguagesCheckComboBox.get
CheckModel().getCheckedItems()));
    }

    public String restoreButtonColor() {
        return getNightModeSwitch() ? «white» : «black»;
    }
```

```
    private void runAutoTranslate() {
        if (getAutotranslateSwitch())
            runTranslate();
    }

    private void selectLanguages() {
        if (selectLanguagesButton.isSelected())

    chosenLanguagesCheckComboBox.getCheckModel().checkAll();
        else

    chosenLanguagesCheckComboBox.getCheckModel().clearChecks();
    }

    private void setTextToTranslate(String s) {
        inputTextArea.setText(s);
    }

    private void setViewColor() {
        if (getNightModeSwitch()) {
            paneRoot.getStylesheets().clear();
            paneRoot.getStylesheets().add(«/css/night.css»);

        } else {
            paneRoot.getStylesheets().clear();
            paneRoot.getStylesheets().add(«/css/day.css»);
        }
    }

    private void swapLanguagesInComboBoxes() {
        String temp = getInputLanguageWithoutAutoDetect();
        setInputLanguage(getOutputLanguage());
        setOutputLanguage(temp);
    }
}
```

ДОДАТОК Г

Копія довідки про впровадження результатів дипломного проєкту в діяльність військової частини

ДОВІДКА

про впровадження результатів бакалаврської роботи

в діяльність військової частини

У роботі студента групи ІА-з61 факультету інформатики та обчислювальної техніки НТУУ «КПІ» Милка Дениса Леонідовича на тему: «Автоматизована система перекладу тексту» запропоновано оптимізацію та підвищення ефективності перекладу текстів.

Практичну цінність має миттєвий переклад виділеного тексту, матеріали дипломного проєкту та реалізація застосунку мовою Java 8 за вимогою військової частини.

Проведений аналіз та розроблені на його базі пропозиції щодо спрощення перекладу текстів, подальшої підтримки та розширення застосунку будуть використані в подальшій практичній діяльності військової частини.

«11» 06 2020 р.

Командир військової частини А0354-В

капітан Г.ПОГОРІЛИЙ

Міністерство оборони України Військова частина А0354 "В" Код 08283256