Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ДОДАТКОВОГО МОДУЛЯ ДЛЯ CRM «CREATIO» НА БАЗІ «GOOGLE TRANSLATE API» ДЛЯ ПЕРЕКЛАДУ БАЗИ ДАНИХ

DEVELOPMENT OF ADDITIONAL MODULE FOR CRM «CREATIO» BASED ON «GOOGLE TRANSLATE API» FOR DATABASE TRANSLATION

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Марценюк Вікторія Петрівна
_____
(підпис)

Керівник:
Вознюк Анастасія Вадимівна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«02» 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Марценюк Вікторії Петрівні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Розробка додаткового модуля для CRM «Creatio» на базі «Google Translate API» для перекладу бази даних*

Керівник роботи: Вознюк Анастасія Вадимівна
затверджені наказом закладу вищої освіти від «*23*» *грудня 2022* р. № *961-01-02*

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«9» червня 2023 р.*

3. Вихідні дані до роботи: *програмне забезпечення, вимоги до розробки програмного забезпечення, функціональні та не функціональні вимоги до програмного засобу, Технології програмування мікросерверної частини – .NET, C#, JavaScript, API, business process.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів розробки, опис функціонального наповнення об'єкта проектування, розробка дизайну, розробка додатку.

5. Перелік графічного матеріалу: 18 рис; 4 діаграми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Вознюк А. В.*		
Специфікація вимог до розробленої системи	*Вознюк А. В.*		
Розробка об'єкта проектування	*Вознюк А. В.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	56,4%		
Академічна доброчесність	*Ящук А.А.*		

7. Дата видачі завдання *«2» лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітки
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2023 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*10.03.2023 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2023 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2023 р.*	
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*09.06.2023 р.*	

Здобувач вищої освіти _____ Марценюк В. П.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Вознюк А.В.
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: Марценюк Вікторія Петрівна
Спеціальність і група: 121 Інженерія програмного забезпечення, ІПЗс-31
Обсяг кваліфікаційної роботи бакалавра: 62 стор., 22 рис., 20 літературних джерел
Кількість сторінок записки: 62 сторінки

Тема: ***Розробка додаткового модуля для CRM «Creatio» на базі «Google Translate API» для перекладу бази даних***

Коротка характеристика кваліфікаційної роботи та прийнятих рішень:
Кваліфікаційна робота бакалавра складається з трьох розділів, в яких проаналізовані проблематика та поставлені завдання за темою роботи, здійснено теоретичне дослідження та практичну реалізацію додаткового модуля-перекладача для CRM-системи «Creatio».

Самостійні розробки і пропозицій автора: *Кваліфікаційна робота бакалавра присвячена розробці програмного продукту, який буде використовуватись для перекладу даних БД. Користувачу представлений комфортний та функціональний інтерфейс для перекладу даних та затвердження перекладів.*

Недоліки: *Істотних недоліків в роботі не виявлено.*

Загальний висновок: *У кваліфікаційній роботі бакалавра було спроектовано та створено додатковий модуль-перекладач для CRM-системи з інтеграцією «Google Translate API». Робота є результатом самостійного дослідження. Істотних недоліків не виявлено. Кваліфікаційна робота здобувача освіти Марценюк Вікторії рекомендується до захисту екзаменаційній комісії та заслуговує відмінної оцінки.*

Рецензент: Здибіцька Н.В., к.т.н., доцент кафедри КН
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«___» _____ 202__ р.

ЗАТВЕРДЖУЮ
Директор _____
підпис прізвище, ініціали
«02» червня 2023 р.

АКТ
ВПРОВАДЖЕННЯ ПРОПОЗИЦІЙ ТА РОЗРОБОК КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА В ДІЯЛЬНІСТЬ ТОВ «МАСИВ-А.І.»

Замовник **ТОВ «МАСИВ-А.І.»**
(найменування підприємства – об'єкта кваліфікаційної роботи)

Цим актом підтверджується, що результати кваліфікаційної роботи на тему

Розробка додаткового модуля для CRM «Creatio» на базі Google Translate API для перекладу бази даних

(найменування теми, затвердженої наказом № *961/01-02* від *23.12.2022* р.
яку виконано *ст. групи ІПЗс-31 Марценюк Вікторією Петрівною*
(П.І.Б. студента)

на кафедрі *Інженерії програмного забезпечення*

спеціальності *121 «Інженерії програмного забезпечення»*

яка виконувалася з *11. 04. 2023* по *10. 06. 2023* р.

впроваджені *в діяльність Товариства з обмеженою відповідальністю «Масив-А.І.»*
(найменування підприємства, де здійснювалось впровадження)

1. Вид впроваджених результатів:
Додатковий модуль для CRM „Creatio" на базі Google Translate API для перекладу бази даних.
2. Характеристика масштабу впровадження
розроблено додатковий модуль для CRM
3. Наукова новизна результатів кваліфікаційної роботи:
результати цієї роботи можуть бути корисними для компаній, які використовують CRM систему „Creatio" та потребують автоматичного перекладу БД для ефективної міжнародної комунікації та розширення своєї клієнтської бази.

ВІД ЗВО
Науковий керівник кваліфікаційної роботи
_____ Вознюк А.І.
(підпис) П.І.Б.

ВІД ПІДПРИЄМСТВА
Директор _____
(підпис) П.І.Б.

Україна м. Київ Товариство з обмеженою відповідальністю "МАСИВ-А.І." Ідентифікаційний код 36346731

АНОТАЦІЇ

Марценюк В. П. Розробка додаткового модуля для CRM «Creatio» на базі «Google Translate API» для перекладу бази даних. Кваліфікаційна робота бакалавра. Кваліфікаційна робота за спеціальністю 121 Інженерія програмного забезпечення. Луцький національний технічний університет. Луцьк, 2023. 53 с.

Робота присвячена розробці додаткового модуля для перекладу бази даних в CRM-системі «Creatio» з інтеграцією «Google Translate API». Основна мета цього дослідження полягає в поліпшенні функціональності CRM-системи, забезпечуючи автоматичний переклад вмісту бази даних на різні мови. У процесі розробки додаткового модуля використовуються сучасні методи програмування та інструменти, забезпечуючи ефективну реалізацію функціональності.

Результати цієї роботи можуть бути корисними для компаній, які використовують CRM-систему «Creatio» та потребують автоматичного перекладу вмісту бази даних для ефективної міжнародної комунікації та розширення своєї клієнтської бази.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Ключові слова: *CRM, система, переклад, провайдер, база даних, таблиця, модуль.*

ANNOTATION

Martseniuk V. P. Development of an additional module for CRM «Creatio» based on the «Google Translate API» for database translation. Bachelor's qualifying work**.** Qualification work on specialty 121 Software engineering. Lutsk National Technical University. Lutsk, 2023. 53 p.

The qualification work is devoted to the development of an additional module for database translation in the CRM system «Creatio» with the integration of «Google Translate API». The main goal of this study is to improve the functionality of the CRM system by providing automatic translation of database content into different languages. In the process of developing the additional module, modern programming methods and tools are used, ensuring effective implementation of functionality.

The results of this work can be useful for companies that use the CRM system «Creatio» and need automatic translation of database content for effective international communication and expansion of their client base.

The bachelor's qualification work consists of an introduction, three sections, conclusions, a list of used sources and appendices.

Keywords: CRM, system, translation, provider, database, table, module.

ЗМІСТ

ВСТУП

У сучасному світі, де глобалізація стає все більш актуальною, забезпечення швидкого та точного перекладу стає важливою складовою для багатьох підприємств. Зокрема, компанії, які використовують CRM-систему «Creatio» для керування своїми клієнтами та базою даних, потребують надійного і ефективного інструменту для перекладу цих даних.

У зв’язку з цим, розробка додаткового модуля для CRM «Creatio» на базі «Google Translate API» має великий потенціал для вирішення цього завдання. «Google Translate API» – це могутній інструмент машинного перекладу, розроблений «Google», який забезпечує високоякісні та швидкі переклади між багатьма мовами.

Дана розробка може забезпечити компаніям можливість автоматичного перекладу текстових даних, включаючи клієнтські замовлення, контактну інформацію та інші дані, які зберігаються в системі. Це дозволить компаніям зосередитися на своїх основних функціях та прискорити комунікацію з клієнтами по всьому світу без необхідності вручну перекладати кожен елемент інформації.

Переваги такого додаткового модуля очевидні. Він значно знижує витрати часу та зусиль на переклад, забезпечує високу точність перекладу та дозволяє компаніям ефективніше працювати з клієнтами з різних країн і культур. Крім того, інтеграція з «Google Translate API» забезпечує надійний та безпечний механізм перекладу, що важливо для збереження конфіденційності даних.

Визначити актуальність розробки додаткового модуля для CRM «Creatio» на базі «Google Translate API» та чи стане цей модуль цінним доповненням для компаній, які прагнуть розширити свою глобальну присутність і поліпшити свої комунікаційні процеси.

Метою дослідження є розробка додаткового модуля для CRM-системи «Creatio» для перекладу баз даних. Основним завданням є створення зрозумілого зручного модуля з інтеграцією «Google Translate API».

Об’єктом дослідження є розроблений додатковий модуль, який дозволяє швидко та зручно здійснювати переклад даних в БД. Дослідження включає розробку ПЗ, застосування сучасних технологій, CRM-систем, мов програмування, API провайдерів для досягнення поставленого завдання.

Предметом дослідження є програмна реалізація модуля-перекладача на базі «Creatio» з використанням API та таких технологій як: бізнес-процеси, бібліотеки .NET. Основними сторонами дослідження є проєктування, розробка та тестування модуля.

Завданнями дослідження є:

– виконати аналіз предметної області використання програмного продукту та виділити основні вимоги до його функціональності;

– на основі майбутньої функціональності додатку здійснити вибір програмних технологій та засобів реалізації;

– дослідити можливості інтеграції «Google Translate API» з CRM системою «Creatio»;

– виконати побудову структурної та функціональної схеми програмного модулю;

– розробити модуль для CRM системи «Creatio», керуючись попередньо побудованими структурною та функціональною схемами, який буде забезпечувати переклад даних в базі даних за допомогою «Google Translate API». Налаштувати модуль для використання різних мов;

– забезпечити безпеку та конфіденційність даних, які підлягають перекладу;

– протестувати розроблений модуль для виявлення та виправлення можливих помилок та недоліків;

– виконати опис розробленого програмного забезпечення, розробити інструкцію користувача з експлуатації. Підготувати звіт про результати роботи та представити його для захисту кваліфікаційної роботи бакалавра.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

На даний момент є досить багато підприємств, які попри труднощі планують збільшувати своє виробництво. За даними Європейської Бізнес Асоціації (ЄБА) 3 з 4 представників малого та середнього бізнесу в Україні (МСБ) планують розширення діяльності у 2023 році.

За планами, сімдесят шість відсотків компаній планують розширювати свій бізнес у 2023 році. З цих компаній двадцять вісім відсотків планують розширювати свою географічну присутність, двадцять шість відсотків мають намір відкривати нові напрямки, двадцять п'ять відсотків бажають вийти на зовнішні ринки, а двадцять один відсоток планують зміну кількості своїх співробітників, як повідомили у ЄБА [16].

Що стосується більш широких перспектив, минулого року Україна отримала довгоочікуваний статус кандидата на вступ до Європейського союзу і прикладає максимум зусиль, щоб прискорити процес інтеграції. Це стане ключовим каталізатором змін для вітчизняного бізнесу, оскільки Україна стане повноправним учасником європейського ринку.

Звичайно, приєднання до ЄС призведе до зростання конкуренції для українського бізнесу, що змусить його прагнути підвищити свою конкурентоспроможність. Однак це також відкриє нові можливості для розширення ринків збуту та притоку додаткових інвестицій в українську економіку.

Щоб успішно конкурувати на міжнародному ринку потрібно вміти швидко пристосовуватись до нових умов; реорганізовувати взаємодію зі споживачами, Тому що багато підприємств вже сьогодні стикаються з проблемою автоматизації управління відносинами зі клієнтами. Що включає збір, зберігання й аналіз даних про, постачальників, партнерів, клієнтів та їх взаємовідносини.

Крім того постає питання швидкої транслітерації вже наявних даних адже переклад даних в БД на різні мови може бути корисним для додатків або сайтів, які мають багатомовну аудиторію. Основна плюсом є те, що це дозволяє користувачам з різних країн та мовних середовищ працювати з додатком або сайтом на рідній для них мові, що підвищує їхню зручність та задоволеність використанням продукту.

Ці вище перераховані проблеми можна вирішити з допомогою використання CRM-систем.

Використання CRM систем дозволяє збільшити продуктивність та результативність бізнесу, зменшуючи час, необхідний для виконання рутинних завдань, наприклад, таких як ведення обліку документації. Однією з найбільш очевидних переваг використання CRM систем є можливість зберігати всі дані про клієнтів в одному місці. Це дозволяє працівникам компанії швидко знаходити всю необроблену інформацію та нові замовлення від клієнтів [20].

CRM (Customer Relationship Management) – це програмні системи, які допомагають компаніям керувати взаємовідносинами зі своїми клієнтами. CRM системи зазвичай зберігають інформацію про клієнтів, їх історію покупок, контактну інформацію, деталі взаємодії з компанією, інтереси та потреби [7].

Ці системи дозволяють спілкуватися з клієнтами на більш ефективному рівні, особливо завдяки автоматизації процесів продажів, маркетингу та обслуговування клієнтів. Вони також надають зручні інструменти для аналізу даних і виявлення тенденцій, що допомагає компаніям приймати стратегічні рішення щодо залучення та утримання клієнтів.

Деякі з основних функцій CRM систем включають:

– збір та збереження інформації про клієнтів: CRM системи дозволяють збирати, зберігати та оновлювати інформацію про клієнтів, включаючи контактну інформацію, історію взаємодії з компанією та інші деталі;

– аналіз даних. Ці системи дозволяють аналізувати дані про клієнтів, щоб зрозуміти їхні потреби, попередній досвід та поведінку. Це дозволяє компанії підходити до кожного клієнта індивідуально та персоналізовано;

– управління продажами. Надають змогу управляти процесом продажу, включаючи ведення списків контактів, створення пропозицій та планування зустрічей;

– управління маркетингом: CRM системи допомагають планувати та виконувати маркетингові кампанії, включаючи розсилку електронних листів, та інші акції;

– управління сервісом підтримки. Дають можливість керувати процесом підтримки клієнтів, включаючи створення та відстеження тікетів підтримки, взаємодію з клієнтами та вирішення проблем;

– звітність та аналітика: CRM системи дозволяють створювати звіти та аналітику про різні аспекти діяльності компанії, включаючи продажі, маркетинг, підтримку та інші. CRM системи можуть бути розроблені для різних галузей, таких як роздрібна торгівля, фінансові послуги, телекомунікації, фармацевтика та інші.

CRM системи можна розділити на категорії, використовуючи різні критерії, серед яких основними є:

– розміщення. CRM системи можуть бути хмарними (SaaS), що означає, що дані зберігаються в хмарі та користувачі мають доступ до системи через Інтернет, або локальними, де дані зберігаються на локальних серверах компанії;

– фокус. CRM системи можуть бути спрямовані на продажі, маркетинг, підтримку клієнтів або керування відносинами з клієнтами загалом;

– масштаб. CRM системи можуть бути призначені для маленьких, середніх або великих компаній залежно від їхньої масштабності та потреб;

– функціональність. Системи з управління відносинами з клієнтами можуть бути розширеними або базовими в залежності від їхніх функціональних можливостей;

– відкритість. Деякі з системи є відкритими, тобто їх можна легко інтегрувати з іншими системами, такими як ERP, маркетингові та фінансові системи;

– ціна. CRM системи можуть бути безкоштовними або платними, в залежності від їхньої функціональності та масштабу.

Відносини з клієнтами є ключовим елементом будь-якої успішної компанії, і CRM системи можуть допомогти забезпечити ефективне управління цими відносинами. Зокрема, вони дозволяють компаніям отримувати більш детальну інформацію про своїх клієнтів, забезпечувати більш персоналізований підхід до обслуговування клієнтів та підвищувати рівень задоволення клієнтів, що може веде до збільшення продажів та збереження клієнтів на довгі терміни.

Вони можуть бути розроблені для великих корпорацій або малих бізнесів, що дозволяє компаніям будь-якого розміру використовувати ці системи для кращого управління своїми клієнтами.

На сьогодні існує досить багато систем для управління відносин з клієнтами, нижче наведено перелік найбільших систем:

– SAP. Основна продуктова лінійка SAP включає такі продукти, як SAP S/4HANA для управління підприємством, «SAP Ariba» для управління закупівлями та постачанням, «SAP SuccessFactors» для управління людськими ресурсами, «SAP Customer Experience» для управління відносинами з клієнтами та інші. SAP є ключовим гравцем на ринку корпоративного програмного забезпечення, забезпечуючи компаніям різних галузей можливість автоматизувати та оптимізувати бізнес-процеси, зменшити витрати та підвищити ефективність [3];

– Oracle CRM пропонує різні функції для підвищення ефективності бізнесу, такі як управління продажами, прогнозування продажів, маркетинговий аналіз та автоматизація маркетингу, управління контактами з клієнтами та інші. Система доступна як хмарне рішення, що дозволяє зменшити витрати на обслуговування та розгортання інфраструктури;

– Microsoft Dynamics CRM – це програмне забезпечення для управління взаємодією з клієнтами, що дозволяє збирати, зберігати та аналізувати дані про клієнтів та їх взаємовідносини з компанією. Воно забезпечує різноманітні функції, такі як управління продажами, маркетингом та обслуговуванням клієнтів, а також

забезпечує інтеграцію з іншими продуктами Microsoft, такими як Outlook та SharePoint. Microsoft Dynamics CRM є частиною пакету Microsoft Dynamics, який також включає в себе рішення для управління виробництвом, фінансами та іншими бізнес-процесами;

– Amdocs CRM – це програмне забезпечення для управління взаємодією з клієнтами, що розроблено компанією Amdocs. Ця CRM-система надає широкі можливості управління клієнтами, включаючи створення та керування замовленнями, управління контактами з клієнтами, підтримку процесу звітування, аналізу даних та багато іншого. Amdocs CRM знаходить застосування у численних компаніях з різних секторів, таких як: телекомунікації, фінанси, медіа та інші, для покращення взаємодії з клієнтами та підвищення ефективності бізнесу.

Перевагами цих систем є:

Microsoft Dynamics CRM:

– інтеграція з іншими продуктами Microsoft, такими як Outlook, Excel і SharePoint;

– налаштовування та розширення можливостей за допомогою конфігурування та програмування;

– Можливість розгортання як в хмарному сервісі, так і на місці.

Oracle CRM:

– велика кількість функцій та можливостей;

– інтеграція з іншими продуктами Oracle, такими як Oracle E-Business Suite та PeopleSoft;

– можливість розгортання як в хмарному сервісі, так і на місці.

SAP AG:

– велика кількість функцій та можливостей, особливо для великих підприємств;

– інтеграція з іншими продуктами SAP, такими як SAP ERP та SAP BW;

– можливість розгортання як в хмарному сервісі, так і на місці.

Amdocs:

– надає широкий спектр можливостей управління взаємодією з клієнтами, включаючи створення та керування замовленнями, управління контактами з клієнтами, підтримку процесу звітування та аналізу даних;

– дозволяє налаштовувати процеси роботи з клієнтами відповідно до потреб та бізнес-вимог компанії;

– забезпечує інтеграцію з іншими системами компанії, що дозволяє отримувати доступ до потрібної інформації та підвищує ефективність роботи.

Недоліками наведених систем є:

– високі вартості. Це програмне забезпечення відоме своїми високими вартостями. Що може стати перешкодою для менших компаній з обмеженим бюджетом;

– складність використання. З великою кількістю функцій та можливостей, ПЗ може бути складним у використанні. Для користувачів, які не мають досвіду з CRM, може знадобитися багато часу та зусиль, щоб оволодіти навичками;

– довгий термін впровадження. Впровадження цих систем може займати багато часу та зусиль, що може бути проблематичним для компаній з обмеженими ресурсами;

– залежність. Після впровадження підприємства можуть стати залежними, що призводить до ускладнення підтримки та їх розвитку, що в свою сергу обмежує їх гнучкість та варіативність. Наприклад, для повного використання всіх функцій Microsoft Dynamics CRM потрібно встановити та налаштувати інші продукти Microsoft, що може збільшити загальну вартість та складність використання системи;

– системні вимоги. програмне забезпечення вимагає великої кількості ресурсів, що може бути проблематичним для компаній зі слабкими комп’ютерними системами або обмеженим доступом до інтернету;

– складна інтеграція з деякими старішими системами та програмним забезпеченням, що може затримувати розгортання системи та збільшувати витрат.

Але найголовнішою проблемою цих CRM-систем є те, що вони потребують великої кількісті технічних і людських ресурсів для обслуговування, тому для вирішення поставленого завдання було обрано CRM-систему «Creatio» від компанії «Terrasoft». До того ж вона є комерційним Open Source, що дозволяє легко впроваджувати і розробляти нові допоміжні модулі. Також в цю систему досить легко інтегрувати допоміжні застосунки такі як, наприклад, провайдери для онлайн-перекладу.

«Creatio» – це хмарна CRM-система, яка надає комплексні інструменти для управління відносинами зі споживачами, продажами, маркетингом та сервісом підтримки. Система пропонує функціонал для автоматизації процесів продажу, маркетингу та обслуговування клієнтів, включаючи створення кампаній, управління продажами, аналіз даних та розподіл завдань між співробітниками. Creatio також має інтегрований інструментарій для аналізу та прогнозування діяльності компанії, який дозволяє вибудувати ефективну стратегію розвитку бізнесу.

1.2 Постановка завдання на кваліфікаційну роботу

Завданням на кваліфікаційну роботу бакалавра є розробка додаткового модуля для CRM системи «Creatio» з допомогою якого буде здійснюватися переклад даних в базі даних, з інтеграцією «Google Translate API».

Модуль буде реалізований як бізнес-процес з інтуїтивно-зрозумілим інтерфейсом користувача. Він повинен мати наступні можливості:

– автоматичний запис перекладених даних в розділ;

– можливість редагувати машинний переклад та затверджувати або не затверджувати його;

– автоматичне визначення мови з якої здійснюється переклад;

– запис вже наявних даних в допоміжну системну БД.

Для реалізації поставленого завдання було визначено наступні цілі:

– провести аналіз предметної області використання програмного продукту та виділити основні вимоги до його функціональності;

– на основі майбутньої функціональності додатку здійснити вибір програмних технологій та засобів реалізації;

– дослідити можливості інтеграції «Google Translate API» з CRM системою «Creatio»;

– виконати побудову структурної та функціональної схеми програмного модулю;

– розробити модуль для CRM системи «Creatio», керуючись попередньо побудованими структурною та функціональною схемами, який буде забезпечувати переклад даних в базі даних за допомогою «Google Translate API». Налаштувати модуль для використання різних мов;

– забезпечити безпеку та конфіденційність даних, які підлягають перекладу;

– протестувати розроблений модуль для виявлення та виправлення можливих помилок та недоліків;

– виконати опис розробленого програмного забезпечення, розробити інструкцію користувача з експлуатації. Підготувати звіт про результати роботи та представити його для захисту кваліфікаційної роботи бакалавра.

Висновки до розділу 1

Зважаючи на все вищеописане, можна зробити висновок, що розробка є актуальною, а також для виконання поставленого завдання було обрано найоптимальнішу систему управління відносинами з клієнтами та провайдер на основі яких буде здійснюватися розробка модуля-перекладача.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Після проведення аналізу предметної області та цілей для реалізації програмного продукту, було встановлено такі вимоги до функціональних характеристик: надійність проекту, параметри технічних засобів, інформаційна та програмна сумісність.

Ці вимоги повинні бути враховані під час використання та реалізації поставленого завдання.

Розроблений модуль має задовольняти основні вимоги, спрямовані на кінцевого користувача програми:

– зручна робота користувача з продуктом;

– функціональне керування.

Розроблений модуль повинен виконувати такі функції:

– коректне відображення розробленого розділу;

– стійке функціонування;

– зручний інтерфейс користувача;

– можливість вибору мови;

– масштабованість (можливість легко переносити його на між різними системами);

– безпека та конфіденційність даних.

Вимоги до надійності.

Програма має гарантувати безперебійне функціонування операційної системи, провайдера і CRM-системи «Creatio» без виклику збоїв. Програмний модуль повинен надійно працювати та забезпечувати стійке з’єднання з «Google Translate API». У випадку виникнення будь-яких помилок, програма має надавати коректні повідомлення з подальшими вказівками щодо їх усунення. Програмний

продукт повинен контролювати вхідні та вихідні дані, щоб вони відповідали вимогам.

Стабільне функціонування програмного засобу гарантується, якщо використовувати оновлене обладнання (ЕОМ) та CRM, а також точно дотримуватися рекомендацій.

Файли проекту мають бути обмежені в доступі. Видалення будь-якого файлу проекту може призвести до некоректної роботи програмного продукту та викликати можливі збої.

Умови експлуатації.

Умови роботи повинні відповідати технічним нормам експлуатації комп'ютера та нормам гігієни. Працювати з програмою можуть клієнти з мінімальним рівнем знань CRM систем. Для обслуговування програмного продукту достатньо однієї людини.

Вимоги до програмної документації.

Склад необхідної програмної документації:

– текст програми;

– інструкція користувача;

– опис програми – довідкова інформація про роботу системи і підказки користувачеві;

– пояснювальна записка – загальний опис та схема функціонування програми.

Порядок контролю і приймання.

Контроль над системою здійснюють кінцеві користувачі, які приєднуються під час тестування. Прийом комплексу відбувається після повної установки та налаштування для окремих користувачів, а також після проведення короткого навчального курсу.

Після завершення розробки системи необхідно провести наступні види тестувань: перевірку на захист від неправильного використання та перевірку на повноту обміну інформацією між різними системами.

2.2 Моделі елементів модуля перекладу у нотації UML

Для кращого розуміння функціональних вимог для розроблюваного модуля було створено діаграму прецедентів, її зображено на рисунку 2.1.

Діаграма прецедентів допомагає визначити вимоги до системи, а також взаємозв'язки між різними акторами (користувачами, іншими системами) та функціональними можливостями системи. Це дозволяє створювати більш детальний план розробки системи та визначати її архітектуру перед початком розробки. Основна мета діаграми прецедентів полягає у спрощенні розуміння функціонального призначення системи та її взаємодії з зовнішнім середовищем, включаючи користувачів, інші системи та обладнання [17].

На цій діаграмі чітко зображено, які маніпуляції здійснює кінцевий користувач, що виконується розробленим модулем машинного перекладу, а що реалізується провайдером.

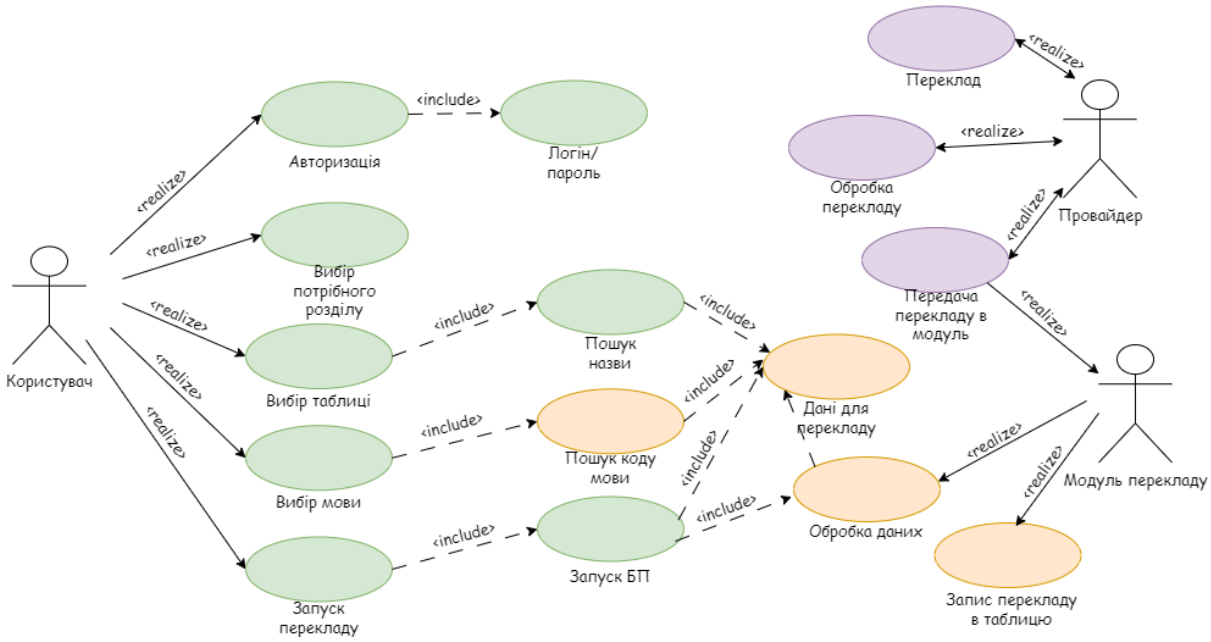

Рисунок 2.1 – Діаграма прецедентів розроблюваної системи

2.3 Вибір засобів, методів і алгоритмів для вирішення поставленого завдання

Як зазначалось в першому розділі розробка буде здійснюватися на базі CRM-системи «Creatio» та з інтеграцією провайдера перекладу «Google Translate API».

Цей API, розроблений компанією «Google» дозволяє розробникам інтегрувати машинний переклад в свої додатки. Він використовує високотехнологічну нейронну машину перекладу від «Google» для транслітерації тексту з однієї мови на іншу. Зараз існує досить багато різних онлайн-перекладачів, але саме цього провайдера було обрано для здійснення перекладу.

Він є найоптимальнішим варіантом для розробки модуля перекладу, тому що наведені нижче API, такі як, наприклад: «Microsoft Translator», «Amazon Translate», та «DeepL» мають досить багато недоліків.

Недоліки зазначених вище провайдерів:

«Microsoft Translator»:

– відносно менша точність для деяких менш поширених мов: «Microsoft Translator» може мати меншу точність для менш поширених або менш популярних мов, оскільки їх моделі перекладу можуть бути менш розробленими порівняно зі широко вживаними мовами;

– обмежені обсяги безкоштовного використання: Безкоштовний план «Microsoft Translator» має обмежені обсяги перекладу, що може бути недостатнім для великих проектів або комерційного використання.

«Amazon Translate»:

– відсутність підтримки деяких менш поширених мов: «Amazon Translate» може не підтримувати деякі менш поширені мови, що може бути обмеженням для проектів, які вимагають багатомовної підтримки;

– вартість використання: В порівнянні з деякими іншими сервісами, «Amazon Translate» може бути вартим уваги, оскільки він вимагає плати за

використання. Вартість може бути важливим фактором для проектів з обмеженим бюджетом.

«DeepL»:

– обмежена кількість підтримуваних мов: Незважаючи на високу якість перекладу, «DeepL» підтримує меншу кількість мов порівняно з іншими перекладачами, особливо для менш поширених мов;

– вартість використання API: «DeepL API» є платним сервісом, і вартість використання може бути вищою порівняно з безкоштовними або більш доступними альтернативами;

– обмежена можливість настройки: «DeepL» надає обмежені можливості настройки перекладу, що може бути обмеженням для деяких вимог або специфічних сценаріїв.

«Google Translate API» вирішує проблеми інших перекладачів. Він забезпечує:

– широкий спектр підтримуваних мов: «Google Cloud Translation API» підтримує більше 100 мов, включаючи популярні мови світу. Це робить його високо-гнучким для проектів з багатомовною підтримкою [15];

– висока якість перекладу: «Google» використовує потужні алгоритми машинного навчання та нейронні мережі для досягнення високої якості перекладів. API постійно оновлюється та вдосконалюється для надання найкращих результатів перекладу;

– розширені можливості: «Google Cloud Translation API» не обмежується лише перекладом тексту. Він також надає можливість перекладу аудіо, зображень, веб-сторінок і навіть відео. Це дозволяє використовувати API для різноманітних сценаріїв та типів проектів;

– масштабованість та надійність: «Google Cloud Translation API» побудовано на потужній хмарній інфраструктурі «Google», що дозволяє масштабувати та обробляти великі обсяги перекладів. API працює високоефективно та забезпечує стабільність та надійність сервісу;

– інтеграція з іншими сервісами «Google»: «Google Cloud Translation API» легко інтегрується з іншими сервісами та продуктами «Google», такими як «Google Cloud Platform», «Google Apps», «Chrome» та багатьма іншими [8]. Це дозволяє розширити можливості перекладу в рамках екосистеми «Google» та спростити взаємодію з іншими сервісами.

Ці переваги роблять «Google Cloud Translation API» привабливим вибором для розробників та підприємств, які шукають потужний та надійний інструмент перекладу.

Весь програмний код буде написаний в середовищі системи з допомогою її інструментів.

«Creatio» побудована на перевірених технологіях, які є широко розповсюдженими – це HTML 5, JavaScript, CSS, C# та Microsoft .NET Framework.

Весь вихідний код «Creatio» розроблено з використанням фреймворків та мов програмування, які є стандартами для створення серверних і клієнтських компонентів підприємницьких рішень. Це забезпечує не тільки активну підтримку та розвиток всіх складових додатку, але й доступність кваліфікованих фахівців з необхідним набором технологій

Microsoft .NET (або просто .NET) – це платформа розробки програмного забезпечення, розроблена компанією Microsoft. .NET забезпечує середовище виконання програм, компілятори, бібліотеки класів і інструменти для розробки різноманітних типів додатків, включаючи веб-додатки, настільні програми, мобільні додатки, ігри та інше [1].

Особливості фреймворку Microsoft .NET:

– мови програмування: .NET підтримує багато мов програмування, включаючи C#, Visual Basic.NET, F#, C++/CLI і інші. Це дозволяє розробникам вибирати мову залежно від їхніх вподобань і досвіду;

– кросплатформенність: За допомогою .NET Core, версії .NET, яка була випущена в 2016 році, .NET став кросплатформеним. Це означає, що ви можете

розробляти додатки на .NET і запускати їх на різних операційних системах, таких як Windows, macOS і Linux;

– багаторівнева архітектура: .NET має багаторівневу архітектуру, що охоплює середовище виконання (Common Language Runtime – CLR), бібліотеки класів, компілятори і інші компоненти. Це розділення дозволяє забезпечити високу переносимість та зручну розробку програм;

– безпека: .NET надає вбудовану підтримку для різних механізмів безпеки. Це включає в себе механізми аутентифікації, авторизації, криптографії, захисту від атак та інші. Використання цих механізмів допомагає забезпечити безпеку додатків та захистити конфіденційні дані;

– бібліотеки класів: .NET надає широкий набір стандартних бібліотек класів, які включають тисячі готових компонентів і функцій для розробки програм. Ці бібліотеки надають зручний доступ до різноманітних можливостей, таких як робота з мережевими протоколами, базами даних, графікою, обробкою XML, безпекою, шифруванням та багато іншого. Використання цих бібліотек дозволяє розробникам ефективно та швидко реалізовувати багато функціональностей в своїх програмах;

– інтегроване середовище розробки (IDE). Для розробки на платформі .NET доступні потужні інтегровані середовища розробки, такі як Visual Studio, які надають широкий спектр інструментів для написання, налагодження та тестування коду. Ці інтегровані середовища розробки (IDE) забезпечують зручне середовище для розробників, включаючи підтримку автодоповнення коду, аналіз коду, візуальні редактори форм і багато інших корисних функцій;

– розширення можливостей: .NET дозволяє створювати власні розширення і компоненти за допомогою механізму розширень (Extensions) і пакетів NuGet. Це дозволяє розробникам додавати нові функції і функціональність до своїх додатків, використовуючи вже існуючі компоненти або створюючи власні;

– веб-розробка: .NET має потужні фреймворки для веб-розробки, такі як ASP.NET і ASP.NET Core. Ці фреймворки надають засоби для створення веб-

додатків, веб-служб, API і динамічних веб-сторінок. Вони підтримують різні технології, такі як MVC (Model-View-Controller), SignalR, Web API і багато інших.

C# (C Sharp) – це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона була випущена в 2000 році як частина платформи .NET Framework і стала одним з основних інструментів для розробки додатків для Windows [2].

Особливості мови C#:

– синтаксис: C# має синтаксис, подібний до C і C++, що дозволяє легко переходити від цих мов. Він також запозичує синтаксис з Java, що робить його зрозумілим для розробників, які знайомі з цією мовою;

– об’єктно-орієнтована парадигма: C# є повністю об’єктно-орієнтованою мовою, що означає, що всі елементи програми, такі як класи, об’єкти, успадкування та поліморфізм, використовуються для створення модульних і простих в обслуговуванні програм;

– раннє зв’язування: C# використовує раннє зв’язування, що означає, що перевірка типу відбувається під час компіляції. Це допомагає виявити багато помилок на ранніх стадіях розробки та покращує продуктивність програми;

– між платформні можливості: C# нерозривно пов’язаний із платформою Windows і може використовуватися для розробки програм для різних платформ, таких як Windows, macOS, Linux, Android та iOS. Це дозволило представити .NET Core і .NET 5, які забезпечують кросплатформенну підтримку;

– керування пам’яттю: C# використовує систему автоматичного керування пам’яттю (Garbage Collection), яка звільняє розробника від необхідності вручну виділяти та звільняти пам’ять;

– багатопотоковість: C# надає потужні засоби для роботи з багатопотоковими програмами. Він підтримує створення, синхронізацію та керування потоками, що забезпечує ефективне виконання паралельних завдань і підвищує продуктивність програм;

– велика стандартна бібліотека: C# має велику стандартну бібліотеку класів (Class Library), яка містить багато готових компонентів для виконання різних завдань. Ця бібліотека надає доступ до таких функцій, як робота з дедлайнами, робота з файлами, мережева взаємодія та багато інших, що значно спрощує розробку програм [4];

– параметри розширення: C# дозволяє розширити функціональні можливості мови за допомогою розширених методів (методів розширення) і атрибутів. Це дозволяє створювати власні розширення для типів і додавати додаткові метадані до коду;

– інтеграція з іншими мовами: C# взаємодіє з іншими мовами, такими як C++, COM (Component Object Model), Python та іншими. Це дозволяє використовувати в одній програмі функції та компоненти, написані на різних мовах програмування;

– підтримка розробки інтерфейсу користувача: C# має різні інструменти та бібліотеки, такі як Windows Forms, WPF (Windows Presentation Foundation) і ASP.NET, для розробки графічного інтерфейсу користувача. Це дозволяє створювати різноманітні програми з привабливим та інтуїтивно зрозумілим інтерфейсом.

Ці особливості роблять мову C# потужним і гнучким інструментом для розробки різних типів додатків.

HTML 5 (HyperText Markup Language 5) є останньою версією стандарту HTML, який використовується для створення веб-сторінок і веб-додатків. Основні особливості HTML 5 включають:

– розмітка сторінок: HTML 5 надає розширені можливості для розмітки сторінок, включаючи нові елементи, що спрощує створення структурованих та семантично валідних сторінок [6];

– мультимедіа: HTML 5 включає в себе вбудовану підтримку відео та аудіо без необхідності використання сторонніх плагінів, таких як Adobe Flash;

– графіка і вектори: HTML 5 має підтримку для візуалізації графіки і векторних зображень. Це досягається за допомогою тегів дозволяє створювати динамічні графічні ефекти та інтерактивні анімації;

– форми і валідація: HTML 5 включає нові елементи та атрибути для створення потужних форм. Наприклад, тег <input> має різні типи, такі як e-mail, url, number, date, що спрощує валідацію даних без необхідності використання JavaScript.

HTML 5 пропонує багато нових можливостей для розробки веб-сторінок і веб-додатків, що сприяє покращенню їх взаємодії, графіки, мультимедіа та зручності використання.

CSS (Cascading Style Sheets) – це мова стилів, яка використовується для визначення зовнішнього вигляду і форматування веб-сторінок. Вона дозволяє розробникам контролювати стиль, розташування та оформлення елементів на веб-сторінці [5].

CSS використовує правила, які містять селектори та властивості, для надання стилю елементам веб-сторінки. Селектори вказують, на які елементи будуть застосовуватись стилі, а властивості визначають конкретні атрибути, що будуть застосовані до цих елементів. За допомогою CSS можна змінювати різні аспекти стилізації, такі як кольори, шрифти, розміри, відступи, рамки, фонові зображення та багато іншого.

CSS дозволяє створювати привабливий та організований зовнішній вигляд веб-сторінок, сприяючи покращенню їх естетики і користувацького досвіду.

JavaScript – це мова програмування, що використовується для створення динамічних та інтерактивних веб-сайтів. Вона працює на боці клієнта (веб-переглядача) і забезпечує можливість взаємодії користувача з веб-сторінкою, змінювати та маніпулювати вмістом сторінки в реальному часі.

JavaScript має широкий спектр функціональності, включаючи можливість обробки подій, зміна вмісту сторінки, валідація форм, створення анімації, взаємодія з сервером за допомогою AJAX і багато іншого. Вона також підтримує об'єктно-

орієнтоване програмування, що дозволяє створювати і використовувати об'єкти з методами і властивостями.

JavaScript є однією з найпоширеніших мов програмування, особливо у веб-розробці, вона надає можливості для створення динамічних, інтерактивних та багатофункціональних веб-додатків.

Інфраструктура системи «Creatio» включає відкрите програмне забезпечення з високою надійністю та продуктивністю. Інструменти, які використовуються в цій системі, є якісними та гнучкими, а їх активний розвиток дозволяє швидко розширювати функціональні можливості «Creatio». Використання компонентів з відкритим кодом дозволяє розробляти додатки відповідно до сучасних технічних вимог, вирішувати складні технологічні завдання та значно економити кошти на придбання ліцензійного програмного забезпечення, якщо «Creatio» буде розгорнуто на власних серверах.

Враховуючи постійно зростаючі вимоги до продуктивності, відмовостійкості та масштабованості корпоративних програм, які кидають виклик більшості рішень SaaS, підхід до розробки платформи «Creatio» базується на принципах мікросервісної архітектури (рис. 2.2).

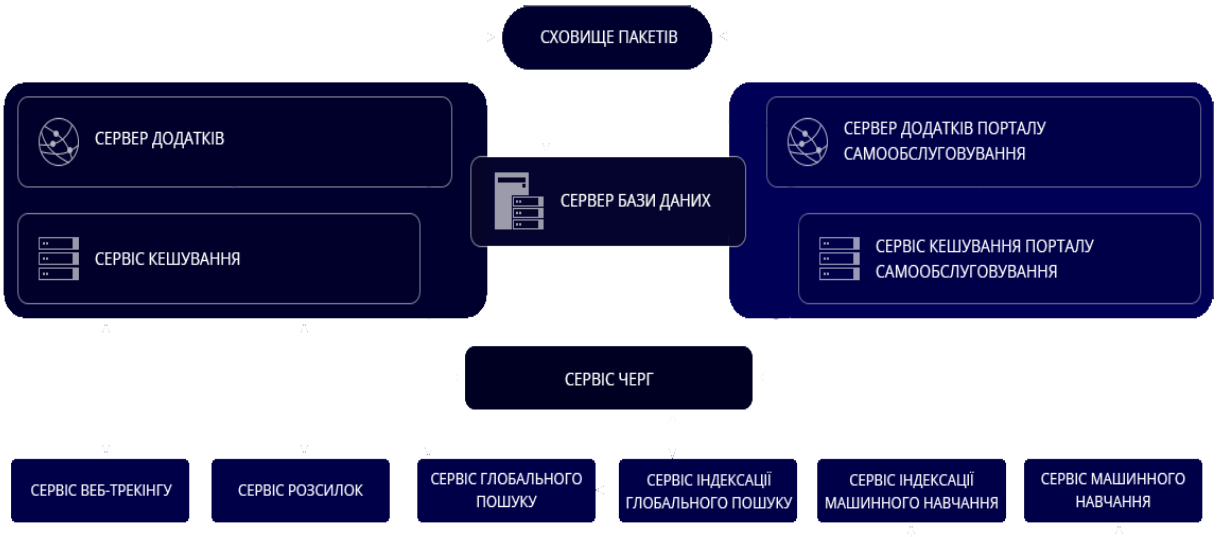

Рисунок 2.2 – Сервісна архітектура «Creatio»

Підтримка різних СУБД. «Creatio» – це незалежна від СУБД платформа на основі власної ORM. Це робить будь-яке рішення на основі «Creatio» автоматично сумісним з усіма реляційними базами даних, які підтримує «Creatio».

На даний момент «Creatio» сумісний з СУБД «Microsoft SQL Server», «Oracle» або «PostgreSQL».

Для виконання поставленого завдання буде використовуватись «PostgreSQL».

«PostgreSQL», часто згадувана як «Postgres», є потужною та розширеною системою управління базами даних (СУБД), яка використовуєся для зберігання, керування та отримання доступу до структурованих даних. Вона є однією з найпопулярніших відкритих джерел даних, що використовується у розмаїтих застосунках та проектах [13].

Основні особливості «PostgreSQL» включають:

– реляційна база даних: «PostgreSQL» підтримує реляційну модель даних, яка дозволяє створювати таблиці структурованих даних і створювати зв’язки між ними;

– розширені функції: він має широкий спектр функцій, включаючи підтримку складних запитів, індексування, транзакцій, збережених процедур, тригерів, реплікації та багато іншого;

– розширюваність: «PostgreSQL» можна розширити за допомогою розширень і спеціальних функцій, що дає можливість розширити функціональні можливості СУБД відповідно до потреб проекту;

– висока надійність і безпека: «PostgreSQL» надає механізми для забезпечення надійності даних, включаючи підтримку транзакцій і можливість відновлення після збоїв. Він також пропонує різні механізми безпеки, включаючи автентифікацію, авторизацію та шифрування;

– підтримка стандартів: «PostgreSQL» дотримується стандартів SQL і активно розвивається, включаючи підтримку нових функцій і вдосконалень.

Таким чином, «PostgreSQL» знаходить широке застосування в різних галузях, включаючи веб-розробку, геопросторовий аналіз, аналітику даних, фінанси, наукові

дослідження та інші. Його активна спільнота розробників сприяє постійному оновленню та підтримці цієї системи управління базами даних.

Крім того, «PostgreSQL» є безкоштовним та відкритим програмним забезпеченням, доступним під ліцензією «PostgreSQL». Це означає, що її можна використовувати «PostgreSQL» безкоштовно, змінювати його за своїми потребами та розповсюджувати його у своїх проектах.

Загалом, «PostgreSQL» є потужним та надійним рішенням для управління базами даних, яке надає розширені можливості, гнучкість, безпеку та високу продуктивність. Воно є важливим інструментом для багатьох розробників та організацій, які працюють зі структурованими даними [19].

Щоб візуалізувати структуру програмного продукту було створено діаграму класів (рис. 2.3).

Діаграма класів – це графічне зображення структури класів та їх залежності в системі або програмному продукті. Вона використовується для моделювання об’єктно-орієнтованих систем та програмних продуктів.

Класова діаграма включає класи, інтерфейси, абстрактні класи, зв'язки та залежності між ними. Кожен клас представляється у вигляді прямокутника з назвою класу, атрибутами та методами. Атрибути описують стан об'єктів класу, а методи визначають їх поведінку.

Класова діаграма дозволяє візуально представити структуру об’єктно-орієнтованої системи або програмного продукту, що полегшує розуміння коду та можливість його змін. Це важливий інструмент для проектування, розробки та документування програмних продуктів.

На рисунку 2.3 зображено діаграму класів «GoogleTranslateHelper» та «EntityTranslateHelper» пакетів «TranslateHelper» та «EntityTranslateHelper», які відповідатимуть за надсилання запитів в «Google Translate API» та оновлятимуть БД після отримання перекладу відповідно.

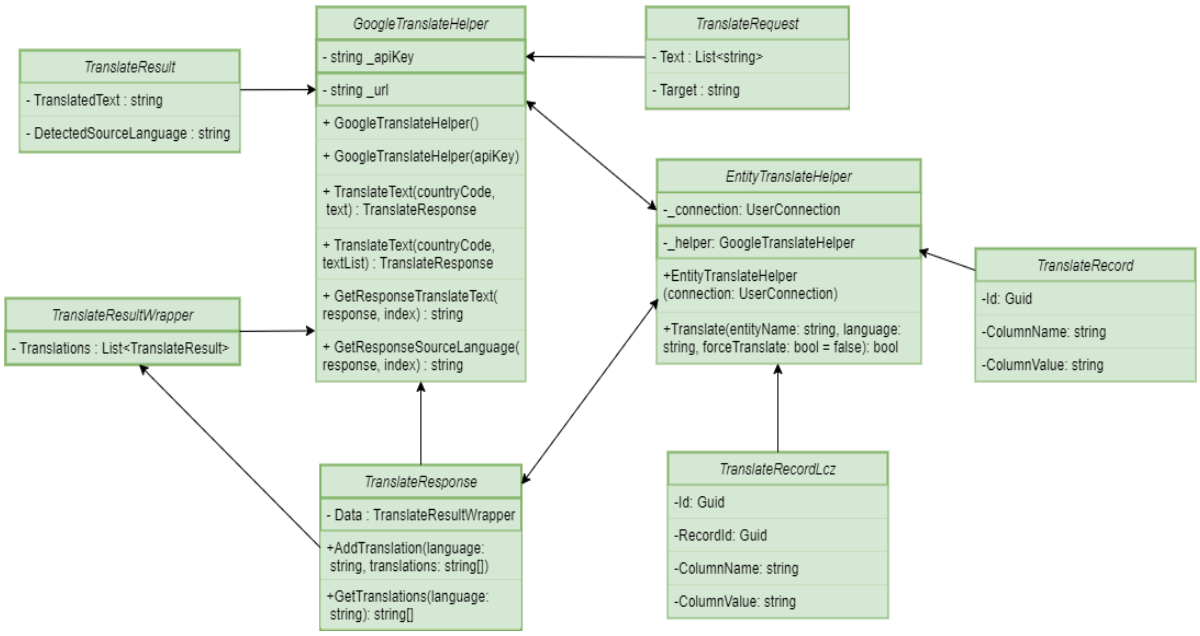

Рисунок 2.3 – Діаграма класів розроблюваної системи

Клас GoogleTranslateHelper – це клас, який містить методи для використання API Google Translate. Він має наступні методи:

GoogleTranslateHelper () – конструктор за замовчуванням.

GoogleTranslateHelper (string apiKey) – конструктор, який приймає ключ API як параметр.

TranslateText (string countryCode, string text) – метод, який приймає код країни та текст, який потрібно перекласти, та повертає відповідь перекладу.

TranslateText (string countryCode, List<string> textList) – метод, який приймає код країни та список текстів, які потрібно перекласти, та повертає відповідь перекладу.

GetResponseTranslateText (TranslateResponse response, int index) – метод, який приймає відповідь перекладу та індекс елемента, що потрібно повернути, та повертає перекладений текст.

GetResponseSourceLanguage (TranslateResponse response, int index) – метод, який приймає відповідь перекладу та індекс елемента, що потрібно повернути, та повертає мову джерела.

Клас TranslateRequest – це клас, який містить текст, який потрібно перекласти, та мову, на яку потрібно перекласти.

Клас TranslateResponse – це клас, який містить обгортку результату перекладу.

Клас TranslateResultWrapper – це клас, який містить список результатів перекладу.

Клас TranslateResult – це клас, який містить текст перекладу та мову джерела.

Клас EntityTranslateHelper – це клас, який містить методи для автоматичного перекладу значень полів для заданої таблиці. Він має наступні методи:

EntityTranslateHelper (UserConnection connection) – конструктор, який приймає підключення користувача.

Translate (string entityName, string language, bool forceTranslate = false) – метод, який приймає назву таблиці, мову, на яку потрібно перекласти, та параметр, який вказує, чи потрібно перекладати всі поля заново (forceTranslate). Метод повертає кількість перекладених полів.

Висновки до розділу 2

Отже, розробка програмного продукту буде здійснюватися з допомогою вищеописаних технологій, а розроблена специфікація окреслює функціональні та нефункціональні вимоги до розроблюваного модуля, описує методи та прийоми, що будуть задіяні в процесі розробки.

РОЗДІЛ 3
РОЗРОБКА МОДУЛЯ ПЕРЕКЛАДУ З ІНТЕГРАЦІЄЮ «GOOGLE TRANSLATE API»

3.1 Практична реалізація об’єкта проєктування

На першому етапі розробки було створено новий пакет під назвою «Translator» в конфігураторі «Creatio». Це спрощує навігацію між різними сервісами та дозволяє перенести проект на інші системи.

Пакет «Creatio» представляє собою набір конфігураційних елементів, які реалізують певний функціонал. Фізично пакет є папкою, яка містить вкладені папки та файли [12].

Для забезпечення доступу до базових бібліотек CRM-системи було використано успадкування від базових пакетів.

Розробка програми «Creatio» ґрунтується на основних принципах проектування програмного забезпечення, зокрема принципу "не повторюйся" (DRY).

У архітектурі «Creatio» цей принцип реалізується за допомогою пакетів та їх залежностей. Кожен пакет містить певний функціонал програми, який може повторюватися в інших пакетах. Щоб використовувати такий функціонал в іншому пакеті, необхідно додати залежність від пакета, в якому він реалізований [10].

Види залежностей:

– щоб поточний пакет успадкував всю функціональність програми, як батьківський пакет необхідно вибрати пакет, який в ієрархії знаходиться наступним після пакета «Custom»;

– щоб поточний пакет успадкував функціональність пакета, як батьківський пакет необхідно вибрати пакет, функціональність якого необхідно успадковувати.

На рисунку 3.1 зображено ієрархію успадкування пакетів та пакет з розроблюваною системою.

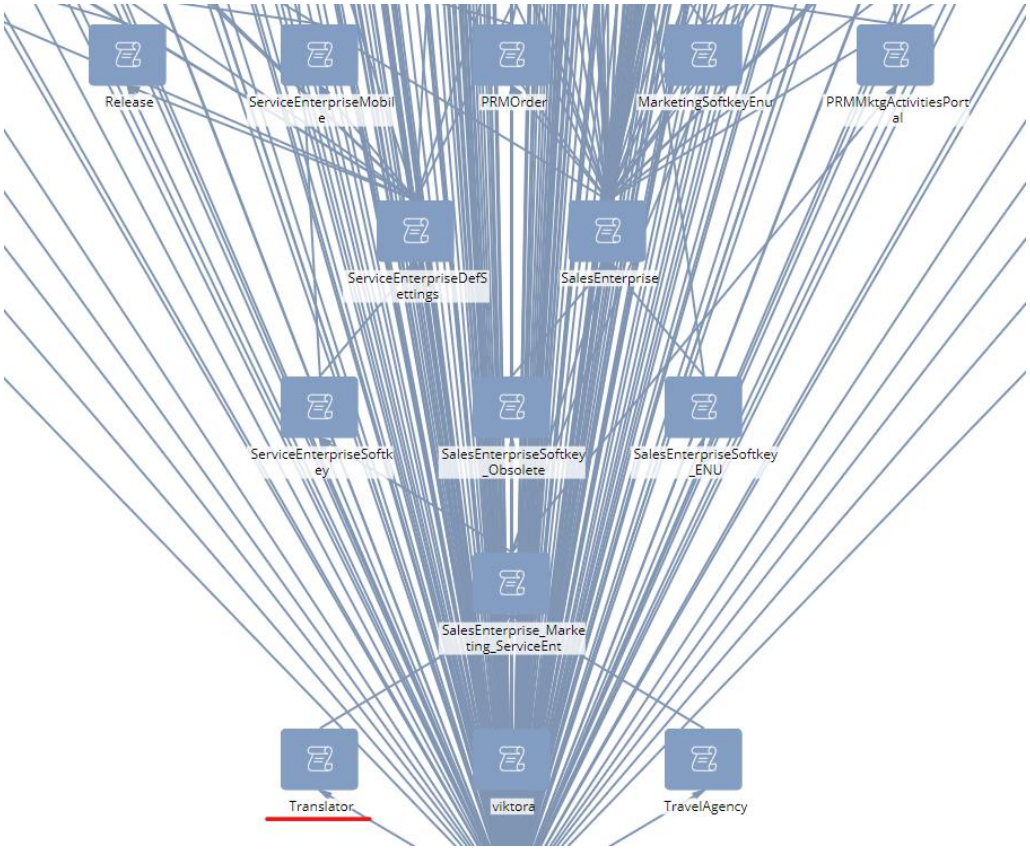

Рисунок 3.1 – Діаграма успадкування пакетів

Нижче наведено лістинг метаданих пакету. Властивість «DependsOn» – масив об’єктів, в яких вказується ім’я пакета від якого здійснюється успадкування, а також його версія та унікальний ідентифікатор, за яким можна визначити пакет у базі даних програми.

Лістинг 3.1 – Метадані пакету «Translator»

```
"Descriptor": {
    "UId": "51b3ed42-678c-4da3-bd16-8596b95c0546",
    "PackageVersion": "7.18.0",
    "Name": "Translator",
    "ModifiedOnUtc": "\/Date(1522653150000)\/",
    "Maintainer": "Customer",
    "DependsOn": [
        {
            "UId": "e14dcfb1-e53c-4439-a876-af7f97083ed9",
            "PackageVersion": "7.18.0",
            "Name": "SalesEnterprise"
        }
    ]
```

```
}
```

Кінець лістингу 3.1

Далі було створено бізнес-процес, який допоможе користувачу спростити використання перекладача (рис. 3.2). Бізнес-процес – це певна послідовність кроків, що веде до мети. З допомогою БП можна створювати та підтримувати високо структуровані послідовності дій, що виконуються як автоматично системою, так і користувачами [11].

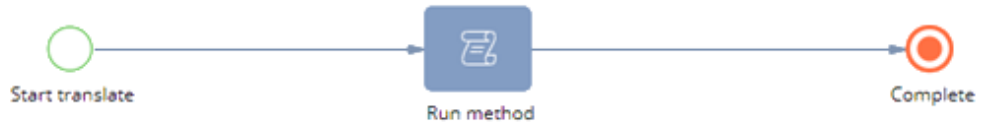

Рисунок 3.2 – Структура БП

В скрипті бізнес-процесу (див. Лістинг 3.2) прописано функцію «Main» з якої відбувається виклик сервісу «EntityTranslateHelper» та в цей сервіс передаються такі параметри:

– «langs» – масив код мов на які потрібно перекласти;
– «entityNames» – масив назв таблиць, які перекладаються.

Лістинг 3.2 – Функція «Main»

```
public void Main() {
    var helper = new EntityTranslateHelper(UserConnection);
    string[] langs = { "en", "uk", "pl" };
    string[] entityNames = { Get<string>("EntityName") };
    foreach(var entity in entityNames)
    foreach(var lang in langs) {
        helper.Translate(entity, lang);
    }
}
```

Кінець лістингу 3.2

Клас «EntityTranslateHelper» (див. Додаток А) використовується для допомоги в перекладі таблиць, включаючи взаємодію з сервісом «TranslateHelper», який викликає машинний переклад з «Google Translate API» та роботу з базою даних. Нижче подано короткий опис цього класу.

Поля «_connection» об’єкт типу «UserConnection» для підключення до бази даних та «_helper» об’єкт типу «GoogleTranslateHelper» для отримання перекладів за допомогою сервісу «Google Translate».

Константа «_enLang» рядок, що містить код англійської мови.

Конструктор «EntityTranslateHelper». Приймає об’єкт «UserConnection» та ініціалізує поля «_connection» і «_helper».

Метод «Translate» виконує переклад елементів зазначеної сутності на вказану мову. Приймає імена сутності (entityName), мови (language) і метод «forceTranslate», який вказує, чи потрібно перекласти навіть ті тексти, які вже мають переклад. Повертає true, якщо переклад успішно виконаний.

Метод «GetCultureIdByCode» повертає ідентифікатор культури за кодом мови. Приймає код мови (code) і повертає «Guid».

Метод «GetSysRecordLczId» повертає запис з таблиці «Sys{entityName}Lcz» для зазначеної сутності, культури і даних запису перекладу. Приймає імена сутності (entityName), ідентифікатор культури (cultureId) і об’єкт «TranslateRecord» (recordData). Повертає об’єкт «TranslateRecordLcz».

Також було створено новий розділ «Machine translating» в якому відбувається перевірка та затвердження машинного перекладу. За автоматичне створення записів в цьому розділі відповідає метод «CreateMachineTranslateEntity» сервісу «EntityTranslateHelper» (лістинг 3.3). Цей метод створює запис у таблиці «MachineTranslating» з вказаною назвою сутності (entityName), ідентифікатором культури (cultureId), ідентифікатором запису (recordId), перекладом (translate), ідентифікатором провайдера (providerId) та ідентифікатором стану (StateId).

Лістинг 3.3 – Метод «CreateMachineTranslateEntity»

```
    private void CreateMachineTranslateEntity(string entityName, Guid
cultureId, Guid recordId, string translate, Guid providerId, DBExecutor
dbExecutor) {
        var ins = new Insert(_connection)
        .Into("MachineTranslating")
            .Set("EntityName", Column.Parameter(entityName)
        .Set("CultureId", Column.Const(cultureId))
            .Set("RecordId", Column.Const(recordId))
            .Set("Translation", Column.Parameter(translate))
            .Set("ProviderId", Column.Const(providerId))
            .Set("StateId", Column.Const(new Guid("40a465c5-b8b5-4f5c-
8602-1ffbbdefa391")))
                    as Insert;
                    ins.Execute(dbExecutor);
    }
```

Кінець лістингу 3.3

Вищеподаний лістинг викликається кожного разу, коли здійснюється виконання запиту оновлення чи запису даних «dbExecutor». Його використання зумовлене тим, що він забезпечує керування транзакціями, що дозволяє здійснювати групу операцій як єдину транзакцію, забезпечуючи цілісність даних та можливість відкату у разі необхідності.

Сервіс «TranslateHelper» (див. Додаток Б) дозволяє зручно взаємодіяти з сервісом перекладу Google Translate, надаючи методи для виконання запитів на переклад і отримання результатів. В ньому міститься клас Клас «GoogleTranslateHelper», який використовується для спрощеного взаємодії з сервісом перекладу «Google Translate». Він містить наступні елементи:

Внутрішній рядковий ресурс «_apiKey», який містить ключ API для доступу до сервісу «Google Translate».

Внутрішній рядковий ресурс «_url», який містить URL-адресу API «Google Translate».

Конструктор без параметрів «GoogleTranslateHelper», який ініціалізує об'єкт зі значеннями за замовчуванням для «_apiKe»у і «_url».

Конструктор з одним параметром «GoogleTranslateHelper», який дозволяє задати значення «_apiKey» при створенні об’єкта.

Метод «TranslateText», який приймає код країни «countryCode» і текст для перекладу «text» і повертає об’єкт «TranslateResponse», який містить результат перекладу.

Метод «TranslateText», який приймає код країни «countryCode» і список текстів для перекладу «textList» і повертає об’єкт «TranslateResponse», який містить результати перекладу для кожного тексту.

Метод «GetResponseTranslateText», який приймає об’єкт «TranslateResponse» і індекс перекладу і повертає перекладений текст зазначеного індексу.

Метод «GetResponseSourceLanguage», який приймає об’єкт «TranslateResponse» і індекс перекладу і повертає визначену мову джерела тексту зазначеного індексу.

Також у коді визначені наступні сутності (класи):

«TranslateRequest». Об’єкт даних для відправки запиту на переклад. Містить список текстів для перекладу (Text) і цільову мову перекладу (Target).

«TranslateResponse». Об’єкт даних, який отримується відповідь від сервісу перекладу. Містить обгортку результатів перекладу (Data).

«TranslateResultWrapper». Обгортка результатів перекладу. Містить список об’єктів «TranslateResult», які містять перекладений текст і визначену мову джерела тексту.

«TranslateResult». Результат перекладу. Містить перекладений текст (TranslatedText) і визначену мову джерела тексту (DetectedSourceLanguage).

Додатково було створено бізнес-процес «Approve Translate», який використовується в розділі «Machine translating» для затвердження правильності перекладу. На рисунку 3.3 зображено схему цього БП.

Рисунок 3.3 – Схема бізнес-процесу «Machine translating»

В ньому прописано метод, який змінює поле «State» з допомогою Update (лістинг 3.4)

Лістинг 3.4 – Метод «Run»

```
private void Run() {
    var upd = new Update(UserConnection, "MachineTranslating")
        .Set("StateId", Column.Const(new Guid("GuID")))
        .Where("Id").IsEqual(Column.Const(Get<Guid>("RecordId")))
    as Update;
    upd.Execute();
}
```

Кінець лістингу 3.4

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Для тестування відправки запитів в «Google Translate API» було використано «Postman». Це популярний інструмент для розробки та тестування API. Він надає зручне середовище для створення, відправлення та отримання HTTP-запитів до різних веб-сервісів.

Основні можливості «Postman» включають:

– створення запитів: «Postman» дозволяє легко створювати HTTP-запити, вибираючи тип запиту (GET, POST, PUT, DELETE тощо), встановлюючи заголовки, передавати параметри запиту та тіло запиту [9];

– відправлення запитів: Після створення запиту ви можете відправити його на сервер і отримати відповідь. «Postman» відображає статус-код, заголовки та вміст відповіді;

– тестування API: Ви можете написати тести для перевірки правильності відповідей сервера. «Postman» надає можливості для автоматичного виконання тестів та перевірки відповідності очікуваним результатам;

– колекції та середовища: «Postman» дозволяє організовувати ваші запити в колекції для легшого управління. Ви також можете використовувати середовища для збереження значень змінних, які можна використовувати в запитах;

– підтримка різних типів запитів: «Postman» підтримує роботу з різними типами запитів, такими як REST, SOAP, GraphQL тощо [14].

«Postman» є дуже корисним інструментом для розробників, які працюють з API, дозволяючи ефективно тестувати та взаємодіяти з веб-сервісами.

Для тестування сервісу «EntityTranslate» було створено запит типу POST. Він є одним із методів HTTP-протоколу, який використовується для відправки даних на сервер з метою створення або оновлення ресурсу. У контексті веб-розробки, POST-запити використовуються для відправки даних з клієнта на сервер для подальшої обробки.

В параметрах запиту було вказано унікальний ключ за який відбувається звернення до сервісу «Google». А в тілі запиту задано простий масив тексту призначеного до перекладу (лістинг 3.5).

Лістинг 3.5 – Тіло запиту

```
{
    "q": ["привіт", "я пишу", "диплом"],
    "target": "en"
}
```

Кінець лістингу 3.5

Далі після відправки у відповідь отримуємо масив з перекладеними даними у вигляді JSON-об’єкту (лістинг 3.6). Він містить вкладений об’єкт «data», який

містить масив «translations». Кожен елемент цього масиву представляє один переклад тексту.

У кожному об’єкті перекладу ми маємо наступні поля:

– «translatedText» це текст, який є перекладом оригінального тексту. Наприклад, перший елемент масиву має значення «Hello», що є перекладом оригінального тексту;

– «detectedSourceLanguage» це код мови, який вказує на мову джерела оригінального тексту. Наприклад, у першому елементі масиву це значення «uk», що вказує на українську мову.

Лістинг 3.6 – Масив з перекладом

```
{
    "data": {
        "translations": [
            {
                "translatedText": "Hello",
                "detectedSourceLanguage": "uk"
            },
            {
                "translatedText": "I write",
                "detectedSourceLanguage": "uk"
            },
            {
                "translatedText": "diploma",
                "detectedSourceLanguage": "uk"
            }
        ]
    }
}
```

Кінець лістингу 3.6

Отже, в даному прикладі ми отримуємо тест запиту перекладів тексту для різних мов. Кожен елемент масиву «translations» містить перекладний текст і вказує на мову джерела оригінального тексту. Це дозволяє зрозуміти, що написаний сервіс «EntityTranslate» працює правильно.

Мінімальні вимоги до технічних засобів для нормального функціонування програмного продукту:

– процесор (мінімальні вимоги: 3.0 GHz Intel Pentium 4);

– оперативна пам’ять (мінімальні вимоги: 2GB RAM);

– відеокарта (мінімальні вимоги: відеопам’ять – NVIDIA GeForce 7300 series, ATI Radeon™ X1600 Video Card with 256MB RAM);

Вимоги до програмної та інформаційної сумісності.

Програмний продукт функціонує у системі Windows 10 та вище. Та на всіх версіях CRM-системи «Creatio».

3.3 Розробка документації для програмного продукту

Великий плюс CRM-системи «Creatio» в тому, що вона надає вже вбудовані інструменти для легкого переносу розроблених модулів на інші версії системи, що полегшує роботу з нею. Нижче наведено інструкцію користувача з встановлення та використання розробленого програмного продукту.

Вся розробка проводилась в окремому пакеті конфігуратора «Creatio» (рис. 3.1).

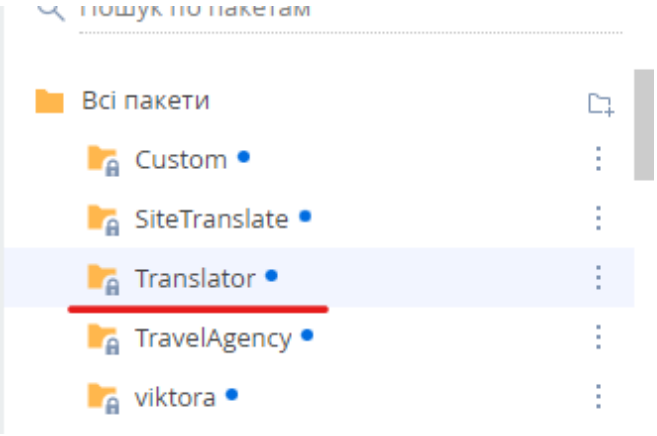

Рисунок 3.1 – Пакет з розробленим модулем

Наступним кроком, потрібно вивантажити пакет з програмним продуктом (рис. 3.2).

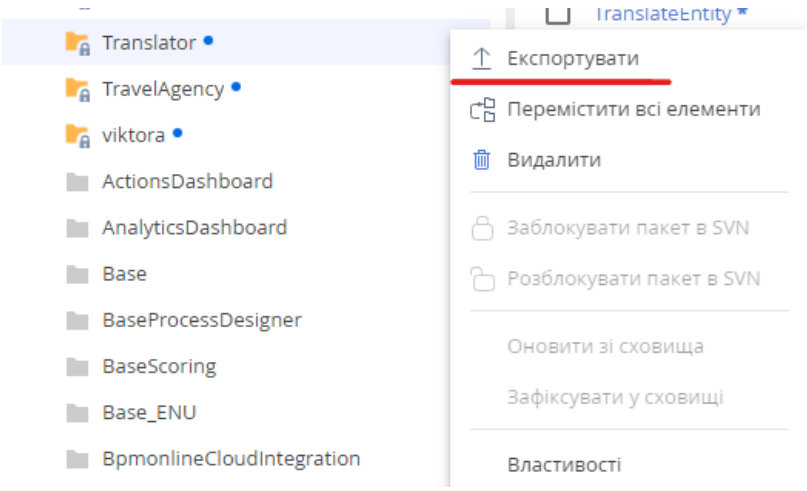

Рисунок 3.2 – Експорт пакету

Після цього створюється інсталяційний архів з ПП. Для його встановлення потрібно перейти в Дизайнер системи та обрати системний розділ «Установка та видалення додатків» (рис. 3.3).

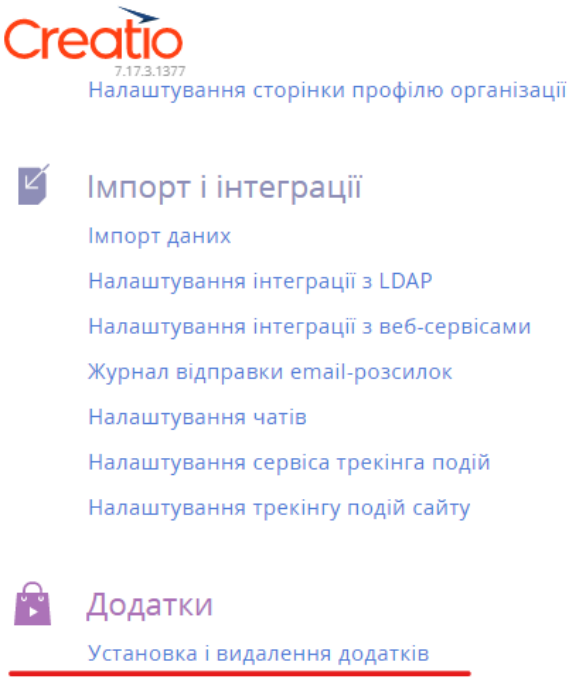

Рисунок 3.3 – Розділ «Додатки»

Переходимо в цей розділ та натискаємо на кнопку «Додати додаток» вибираємо варіант «Встановити з файлу» (рис. 3.4).

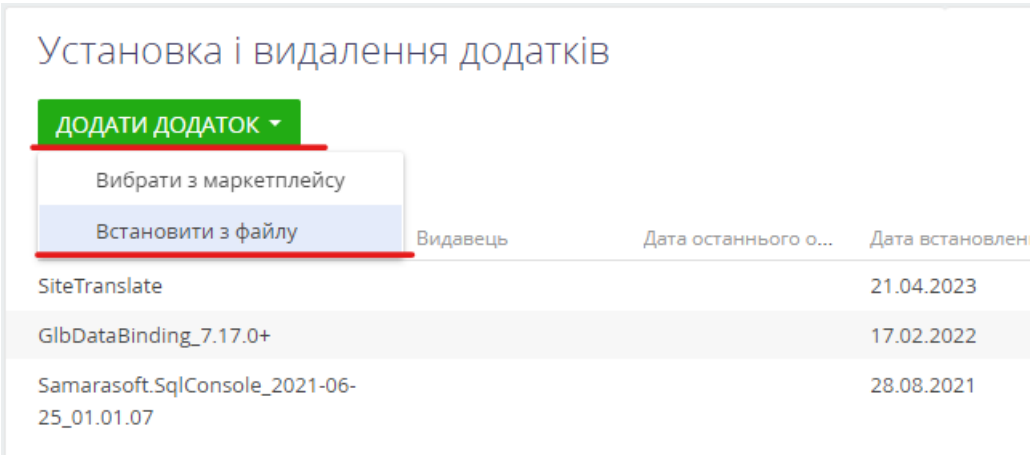

Рисунок 3.4 – Кнопка «Додати додаток»

Потім обираємо завантажувальний файл, який ми створили раніше (рис. 3.5).

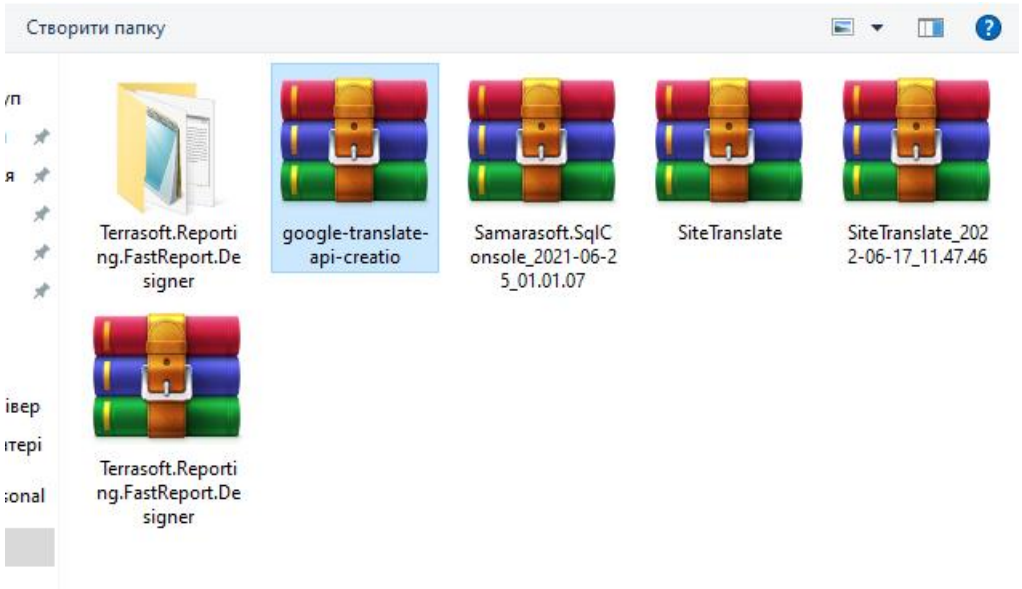

Рисунок 3.5 – Вибір файлу

Після цього відбудеться встановлення нашого пакету на нову систему, якщо станеться помилка система автоматично відкотить всі зроблені зміни та створить лог-файл з описом помилки.

Заключним кроком є компіляція конфігурації, після її завершення, з’явиться новий розділ з назвою «Machine Translating». Встановлення системи завершено, вона повністю готова до використання.

Для того, щоб здійснити переклад будь-якої таблиці з нашої бази даних потрібно виконати наступні кроки:

Переходимо в «System Designer» та обираємо пункт «Бібліотека процесів» (рис. 3.6).

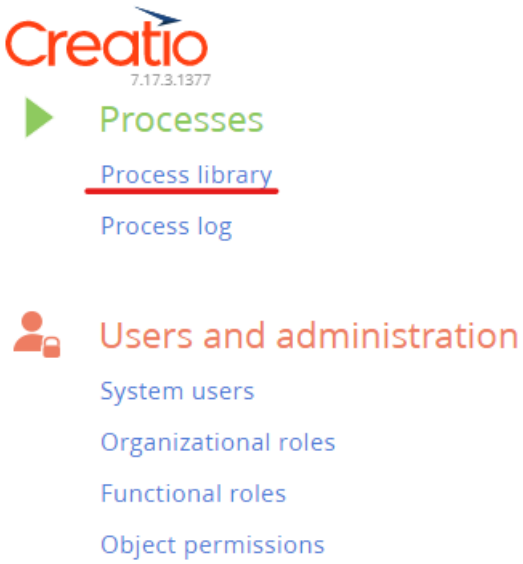

Рисунок 3.6 – Бібліотека процесів

Вибираємо бізнес-процес під назвою «Translate Entity» (рис. 3.7).

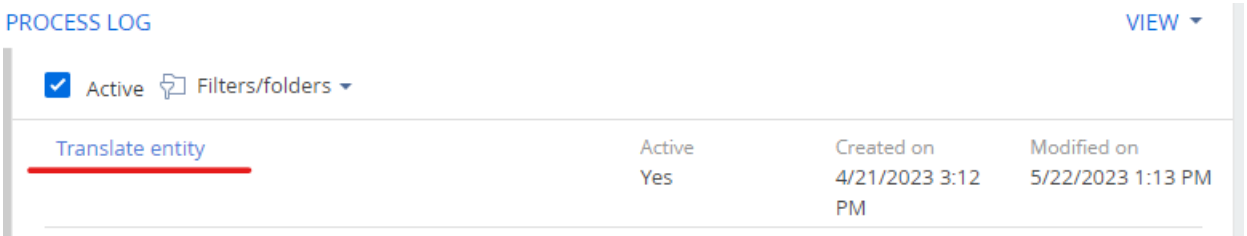

Рисунок 3.7 – Бізнес процес перекладу

На рисунку 3.8 зображено бізнес-процес. В пункті «Parameters» є параметр з назвою «EntityName» сюди потрібно вписати назву таблиці для якої потрібно зробити машинний переклад, зараз там вже прописано назву тестової таблиці.

Рисунок 3.8 – Вигляд процесу

В пункті «Methods» можна змінити код мов на які буде здійснено переклад таблиці, зараз там вказано три мови: англійська, українська та польська (рис. 3.9).

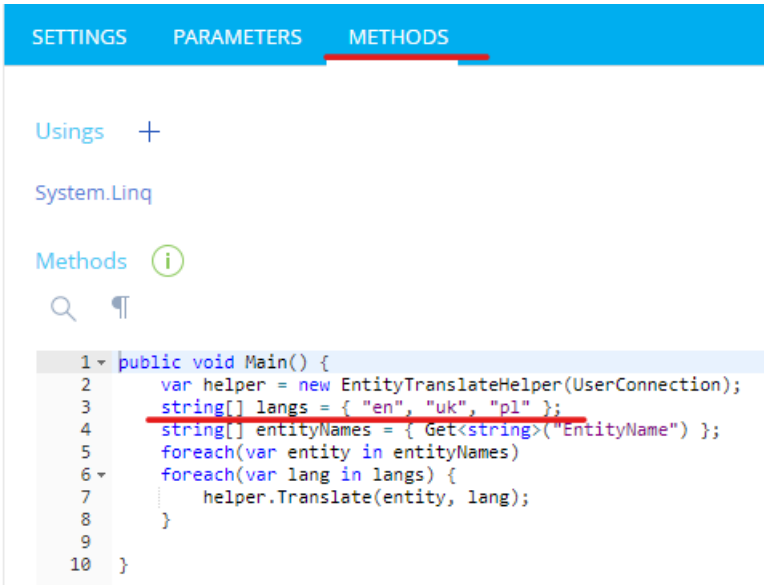

Рисунок 3.9 – Пункт «Methods»

Після внесених змін потрібно зберегти та здійснити компіляцію БП, натискаємо на кнопку «Save» (рис. 3.10).

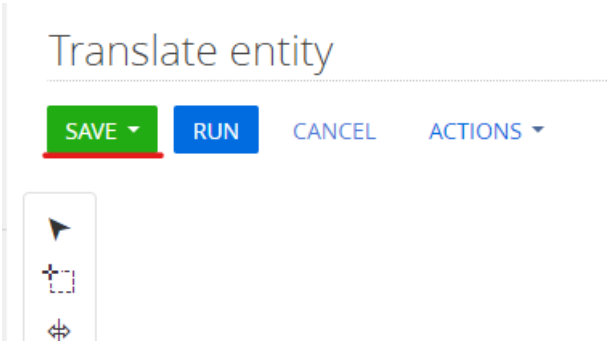

Рисунок 3.10 – Кнопка «Save»

З’явиться діалогове вікно з повідомленням, що процес успішно збережено і, що його потрібно опублікувати (компілювати) перед запуском БП, натискаємо кнопку «Publish» (рис. 3.11).

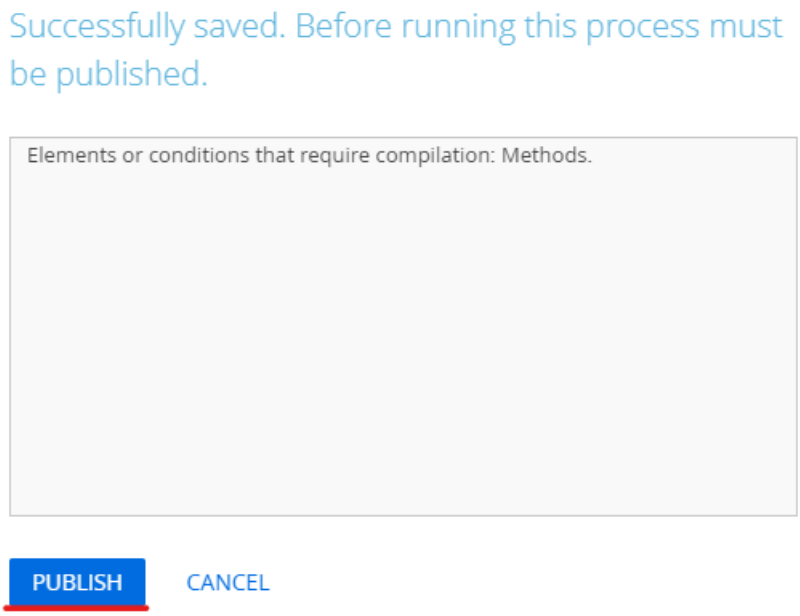

Рисунок 3.11 – Кнопка «Publish»

Після успішного завершення компіляції можна запускати бізнес-процес. Для цього натискаємо кнопку «Run» (рис. 3.12).

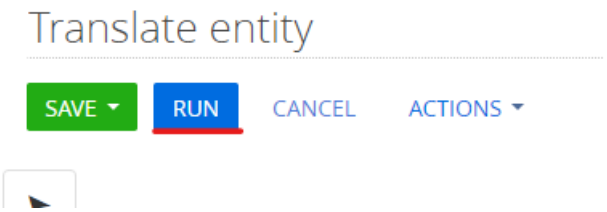

Рисунок 3.12 – Кнопка «Run»

Після цього відбудеться запуск бізнес-процесу та з’явиться повідомлення з написом «Successfully started» (рис. 3.13).

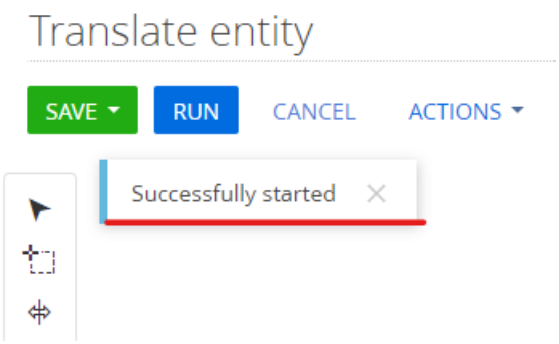

Рисунок 3.13 – Повідомлення про успішний запуск БП

Переходимо в розділ «Machine Translating» тут відображаються записи для яких було зроблено переклад, також ми бачимо мови на яку було зроблено переклад та те що переклад ще не затверджено (рис. 3.14).

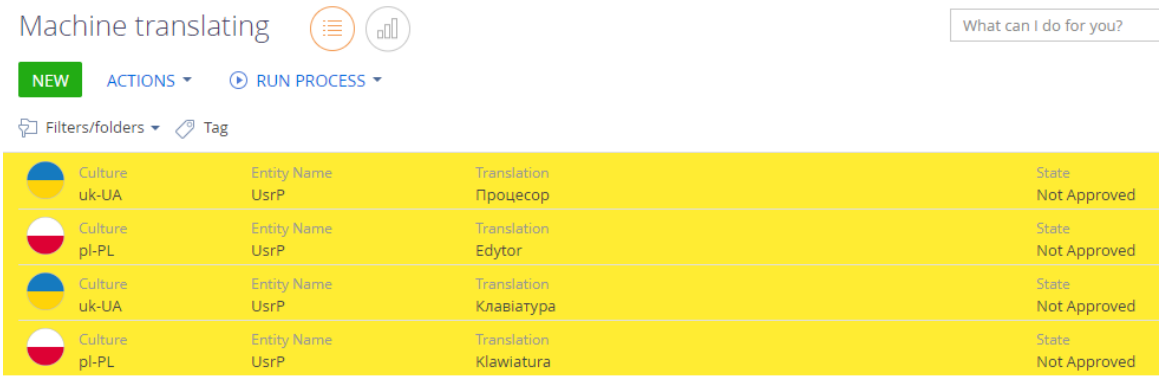

Рисунок 3.14 – Розділ «Machine Translating»

Тепер, якщо переклад нас влаштовує його потрібно затвердити, для цього виділяємо записи та натискаємо кнопку «Run Process» та «Approve Translate» (рис. 3.15).

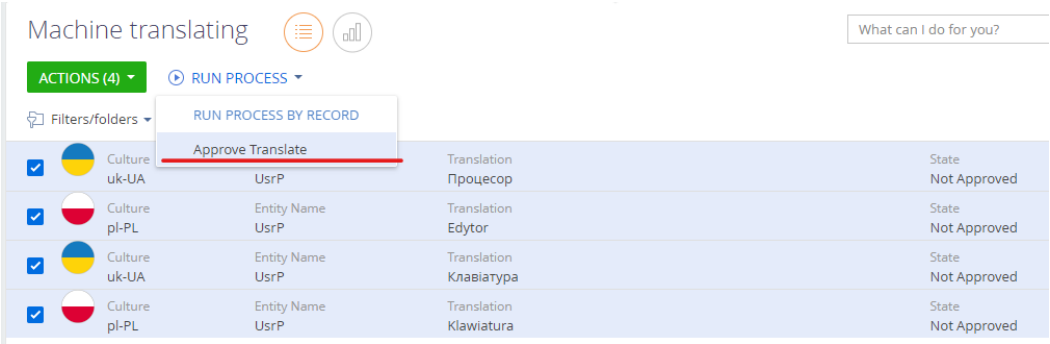

Рисунок 3.15 – Затвердження перекладу

Після, перезавантажуємо сторінку і бачимо, що переклад затверджено

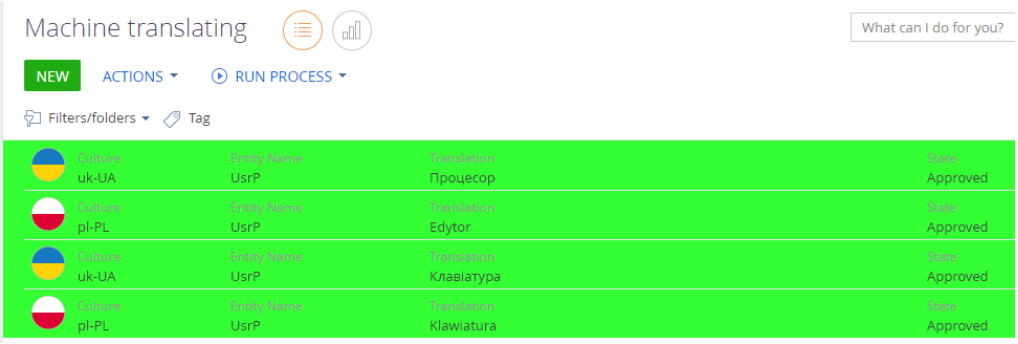

Рисунок 3.16 – Переклад затверджено

Надалі при зміні мови користувача дані в перекладеній таблиці будуть відображатись мовою користувача.

Висновки до розділу 3

В цьому розділі було описано методи та засоби з допомогою яких було досягнено цілей поставлених для виконання завдання та розроблено інструкцію з використання системи.

ВИСНОВКИ

Отже, розробка додаткового модуля для CRM «Creatio» на базі «Google Translate API» для перекладу бази даних є потужним інструментом, який може значно полегшити комунікацію та роботу компаній з клієнтами по всьому світу. Цей модуль надає можливість автоматичного перекладу текстових даних, таких як клієнтські замовлення, контактна інформація та інші дані, які зберігаються в системі CRM.

Переваги такого модуля очевидні. Він знижує витрати часу та зусиль на переклад, забезпечує високу точність перекладу і дозволяє компаніям ефективніше працювати з клієнтами з різних країн і культур. Інтеграція з «Google Translate API» забезпечує надійний та безпечний механізм перекладу, що важливо для збереження конфіденційності даних.

Під час розробки було виконано поставлені завдання**:**

– автоматичний запис перекладених даних в розділ;

– реалізовано можливість редагувати машинний переклад та затверджувати або не затверджувати його;

– автоматичне визначення мови з якої здійснюється переклад;

– запис вже наявних даних в допоміжну системну БД.

В повній мірі досягнено поставлених цілей. В ході виконання поставлених завдань було проведено аналіз предметної області використання програмного продукту і виокремлено основні вимоги до його функціональності. Для реалізації майбутньої функціональності були обрані відповідні програмні технології та засоби.

В ході написання пояснювальної записки було доведено актуальність розробки

Досліджено можливості інтеграції «Google Translate API» з CRM системою «Creatio» та досліджено інші провайдери перекладачі, такі як: «DeepL», «Amazon Translate» та інші. Була побудована структурна та функціональна схеми

програмного модулю, які послужили основою для розробки модуля. Цей модуль повністю забезпечує переклад даних в базі даних за допомогою «Google Translate API» і був налаштований для використання різних мов.

Під час розробки модуля було приділено особливу увагу безпеці та конфіденційності даних, які підлягають перекладу. Розроблений модуль був протестований з допомогою «Posmtan» для виявлення та виправлення можливих помилок та недоліків адже «Creatio» забезпечує інтеграцію з ним.

Також був підготовлений опис розробленого програмного забезпечення та інструкція користувача з експлуатації. Результати роботи були документовані у звіті, який був представлений для захисту кваліфікаційної роботи бакалавра.

В результаті виконаних завдань було успішно розроблено модуль для CRM системи «Creatio», який забезпечує переклад даних за допомогою «Google Translate API». Завдяки цьому модулю користувачі зможуть ефективно працювати з даними у різних мовах, що сприятиме покращенню продуктивності та ефективності роботи в системі «Creatio».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. .NET microservices application architecture guidance. Microsoft. URL: https://dotnet.microsoft.com/en-us/learn/aspnet/microservices-architecture (date of access: 10.03.2023).

2. C# documentation: навчальний посібник. 2023. 3374 p. URL: https://learn.microsoft.com/pdf?url=https://learn.microsoft.com/enus/dotnet/csharp/toc.json (date of access: 17.03.2023).

3. Carter M. C. Mastering SAP: A comprehensive guide to todays SAP Software : навчальний посібник. Independently published, 2023. 88 p.

4. Create custom attributes. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/csharp/advanced-topics/reflection-and-attributes/creating-custom-attributes (date of access: 23.04.2023).

5. DevDocs – CSS documentation. DevDocs API Documentation. URL: https://devdocs.io/css/ (date of access: 28.03.2023).

6. Front-end debugging. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/development_tools/debugging_tools/front_end_debugging/overview (date of access: 07.04.2023).

7. Helgeson L. CRM for dummies : навчальний посібник. For Dummies, 2018. 368 p.

8. Integration options. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/integrations_and_api/integration_options/overview (date of access: 10.04.2023).

9. JIT Optimization and Debugging - Visual Studio (Windows). Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/visualstudio/debugger/jit-optimization-and-debugging?view=vs-2022 (date of access: 12.04.2023).

10. Packages basics. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/development_tools/packages/packages_basics/overview (date of access: 20.04.2023).

11. Process Designer basics. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/user/bpm_tools/business_process_setup/business_process_setup/process_designer_basics (date of access: 21.04.2023).

12. Service that runs business processes. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/integrations_and_api/business_process_service/overview (date of access: 25.04.2023).

13. Special features of working with PostgreSQL. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/development_tools/instrumens_for_database/postgresql_features/overview (date of access: 27.04.2023).

14. Using Postman. Creatio Academy | Improve your skills with Creatio Training Courses & Certification. URL: https://academy.creatio.com/docs/developer/integrations_and_api/using_postman/overview (date of access: 29.04.2023).

15. What is Cloud Translation? | Google Cloud. Google Cloud. URL: https://cloud.google.com/translate/docs/overview (date of access: 01.05.2023).

16. Жирій К. Час для роботи: як відновлювався та адаптувався український бізнес за рік війни. Новини України - останні новини України сьогодні – УНІАН. URL: https://www.unian.ua/economics/finance/chas-dlya-roboti-yak-vidnovlyuvavsya-ta-adaptuvavsya-ukrajinskiy-biznes-za-rik-viyni-12154170.html (дата звернення: 02.05.2023).

17. Марголін О. Діаграми UML для моделювання процесів і архітектури проекту. Evergreen – web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 03.05.2023).

18. Методичні вказівки до виконання кваліфікаційної роботи бакалавра : метод. рек. / уклад.: Н. М. Ліщина, А. А. Ящук, О. М. Суринович. Луцьк, 2022. 56 с.

19. Опис технологій які використовуються в платформі Creatio. One platform to automate workflows and CRM with no-code | Creatio. URL: https://www.creatio.com/ua/our technologies/architecture and principles?activity =adwords_brand_ua (дата звернення: 01.05.2023).

20. Що таке CRM-система? Як обрати й працювати з CRM? Що важливо замірювати? Дія.Бізнес – Головна сторінка. URL: https://business.diia.gov.ua/handbook/prodazi/so-take-crm-sistema-ak-obrati-jpracuvati-z-crm-so-vazlivo-zamiruvati (дата звернення: 02.04.2023).

ДОДАТКИ

ДОДАТОК А

EntityTranslateHelper

```
namespace Terrasoft.Common {
    using Newtonsoft.Json;
    using Newtonsoft.Json.Linq;
    using System;
    using System.Collections.Generic;
    using System.Collections.ObjectModel;
    using System.Linq;
    using System.Text;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.Configuration;
    using Terrasoft.Core.DB;
    using Terrasoft.Core.Entities;
    using Terrasoft.Core.Process;
    using Terrasoft.Core.Process.Configuration;

    // lang codes -https://www.labnol.org/code/19899-google-
translate-languages
    public class EntityTranslateHelper {
        internal UserConnection _connection;
        internal GoogleTranslateHelper _helper;

        internal const string _enLang = "en";

        public EntityTranslateHelper(UserConnection connection) {
            _connection = connection;
            _helper = new GoogleTranslateHelper();
        }

        public bool Translate(string entityName, string language,
bool forceTranslate = false) {
            Guid mainCultureId = GetCultureIdByCode(language);
            if(mainCultureId == Guid.Empty) {
                throw new Exception($"cultureId for lang: {language}
not found");
            }
            bool isEng = language == _enLang;
            string lczEntityName = $"Sys{entityName}Lcz";

            var schema =
_connection.EntitySchemaManager.GetInstanceByName(entityName);
            var columns = schema.Columns;
            var locColumns = columns.Where(e =>
e.IsLocalizable).ToList();
```

```
                var select = new Select(_connection).Column("Id") as
Select;
                foreach(EntitySchemaColumn column in locColumns) {
                    select.Column(column.Name);
                }
                select.From(entityName);
                var recordDataList = new List<TranslateRecord>();
                using(var dbExecutor = _connection.EnsureDBConnection())
                using(var reader = select.ExecuteReader(dbExecutor))
                while(reader.Read()) {
                    Guid id = reader.GetColumnValue<Guid>("Id");
                    foreach(EntitySchemaColumn column in locColumns) {
                        string text =
reader.GetColumnValue<string>(column.Name);
                        if(string.IsNullOrEmpty(text)) {
                            continue;
                        }
                        recordDataList.Add(new TranslateRecord() {
                            Id = id,
                            ColumnName = column.Name,
                            ColumnValue = text
                        });
                    }
                }
                TranslateResponse response =
_helper.TranslateText(language, recordDataList.Select(e =>
e.ColumnValue).ToList<string>());
                Insert insertMain = null;
                Update updateMain = null;
                Guid lastMainRecordId = Guid.Empty;
                Guid lastLczMainRecordId = Guid.Empty;
                int recordsProceed = 0;
                int recordsPerTransaction = 50;
                using(var dbExecutor = _connection.EnsureDBConnection())
{
                    dbExecutor.StartTransaction();
                    for(int i = 0; i < recordDataList.Count; i++) {
                        try {
                            // get params
                            string translatedText =
_helper.GetResponseTranslateText(response, i);
                            string sourceLanguage =
_helper.GetResponseSourceLanguage(response, i);
                            TranslateRecord recordData =
recordDataList[i];
                            if(sourceLanguage == language) {
                                // todo: if first translate not en,
update eng loc
                                continue;
                            }
```

```
                            // check if another record
                            if(lastMainRecordId != recordData.Id &&
lastMainRecordId != Guid.Empty) {
                                if(insertMain != null) {
                                    insertMain.Set("SysCultureId",
Column.Const(mainCultureId));
                                    insertMain.Set("RecordId",
Column.Const(lastMainRecordId));
                                    insertMain.Execute(dbExecutor);
                                    insertMain = null;
                                }
                                if(updateMain != null) {
                                    updateMain.Where("Id").IsEqual(Column
.Const(lastLczMainRecordId));
                                    updateMain.Execute(dbExecutor);
                                    updateMain = null;
                                }
                                lastLczMainRecordId = Guid.Empty;
                                recordsProceed++;
                            }
                            lastMainRecordId = recordData.Id;
                            // main translate
                            if(isEng) {
                                if(sourceLanguage == _enLang) {
                                    continue;
                                }
                                (new Update(_connection, entityName)
                                        .Set(recordData.ColumnName,
Column.Parameter(translatedText))
                                        .Where("Id").IsEqual(Column.Const
(recordData.Id)) as Update).Execute(dbExecutor);
                            } else {
                                if(sourceLanguage != _enLang) {
                                    var engResponse =
_helper.TranslateText(_enLang, new List<string>() { recordData.ColumnValue
});
                                    string engTranslated =
_helper.GetResponseTranslateText(engResponse, 0);
                                    if(string.IsNullOrEmpty(engTranslated
)) {
                                        throw new Exception($"cant
translate to english when translate from {sourceLanguage} to {language};
id: {recordData.Id}");
                                    }
                                    (new Update(_connection, entityName)
                                        .Set(recordData.ColumnName,
Column.Parameter(engTranslated))
                                        .Where("Id").IsEqual(Column.Const
(recordData.Id)) as Update).Execute(dbExecutor);
                                }
```

```
                                    TranslateRecordLcz lczRecord =
GetSysRecordLczId(entityName, mainCultureId, recordData);
                                    if(lczRecord != null) {
                                        if(!lczRecord.IsExistColumnValue ||
(lczRecord.IsExistColumnValue && forceTranslate)) {
                                            lastLczMainRecordId =
lczRecord.Id;
                                            if(updateMain == null) {
                                                updateMain = new
Update(_connection, lczEntityName) as Update;
                                            }
                                            updateMain.Set(recordData.ColumnN
ame, Column.Parameter(translatedText));
                                        }
                                    } else {
                                        if(lastMainRecordId == Guid.Empty) {
                                            lastMainRecordId =
Guid.NewGuid();
                                        }
                                        if(insertMain == null) {
                                            insertMain = new
Insert(_connection).Into(lczEntityName) as Insert;
                                        }
                                        insertMain.Set(recordData.ColumnName,
Column.Parameter(translatedText));
                                    }
                                }
                                // tail
                                Guid tailCultureId =
GetCultureIdByCode(sourceLanguage);
                                if(tailCultureId == Guid.Empty) {
                                    continue;
                                }
                                if(sourceLanguage != _enLang) {
                                    TranslateRecordLcz lczRecord =
GetSysRecordLczId(entityName, tailCultureId, recordData);
                                    if(lczRecord != null) {
                                        if(!lczRecord.IsExistColumnValue ||
(lczRecord.IsExistColumnValue && forceTranslate)) {
                                            (new Update(_connection,
lczEntityName)
                                                .Set(recordData.ColumnName,
Column.Parameter(recordData.ColumnValue))
                                                .Where("Id").IsEqual(Column.C
onst(lczRecord.Id)) as Update).Execute(dbExecutor);
                                        }
                                    } else {
                                        (new
Insert(_connection).Into(lczEntityName)
```

```
                                        .Set(recordData.ColumnName,
Column.Parameter(recordData.ColumnValue))
                                        .Set("RecordId",
Column.Const(recordData.Id))
                                        .Set("SysCultureId",
Column.Const(tailCultureId)) as Insert).Execute(dbExecutor);
                                    }
                                }
                            } catch {
                                dbExecutor.RollbackTransaction();
                                throw;
                            }
                            if(recordsProceed % recordsPerTransaction == 0) {
                                dbExecutor.CommitTransaction();
                                dbExecutor.StartTransaction();
                            }
                        }
                        // todo: move to method
                        if(insertMain != null) {
                            insertMain.Set("SysCultureId",
Column.Const(mainCultureId));
                            insertMain.Set("RecordId",
Column.Const(lastMainRecordId));
                            insertMain.Execute(dbExecutor);
                            insertMain = null;
                        }
                        if(updateMain != null) {
                            updateMain.Where("Id").IsEqual(Column.Const(lastL
czMainRecordId));
                            updateMain.Execute(dbExecutor);
                            updateMain = null;
                        }
                        dbExecutor.CommitTransaction();
                    }
                    return true;
                }
                public Guid GetCultureIdByCode(string code) {
                    string result = "";
                    switch(code) {
                        case("en"):
                            result = "a5420246-0a8e-e111-84a3-00155d054c03";
                            break;
                        case("uk"):
                            result = "f0ea9715-8757-474c-aee9-743d672e48c9";
                            break;
                        case("pl"):
                            result = "479b0043-e504-41ca-b247-b2ffa51933ba";
                            break;
                        case("ru"):
                            result = "1a778e3f-0a8e-e111-84a3-00155d054c03";
```

```
                        break;
                    }
                    if(result == "") {
                        return Guid.Empty;
                    }
                    return new Guid(result);
                }
                public TranslateRecordLcz GetSysRecordLczId(string
entityName, Guid cultureId, TranslateRecord recordData) {
                    if(recordData == null) {
                        return null;
                    }
                    var select = new Select(_connection)
                            .Column("Id")
                            .Column(recordData.ColumnName)
                        .From($"Sys{entityName}Lcz")
                        .Where("RecordId").IsEqual(Column.Const(recordData.Id
))
                            .And("SysCultureId").IsEqual(Column.Const(culture
Id)) as Select;

                    using(var dbExecutor = _connection.EnsureDBConnection())
                    using(var reader = select.ExecuteReader(dbExecutor))
                    while(reader.Read()) {
                        return new TranslateRecordLcz() {
                            Id = reader.GetColumnValue<Guid>("Id"),
                            ColumnName = recordData.ColumnName,
                            IsExistColumnValue =
string.IsNullOrEmpty(reader.GetColumnValue<string>(recordData.ColumnName))
                        };
                    }
                    return null;
                }
            }
            public class TranslateRecord {
                public Guid Id { get; set; }
                public string ColumnName { get; set; }
                public string ColumnValue { get; set; }
            }
            public class TranslateRecordLcz {
                public Guid Id { get; set; }
                public string ColumnName { get; set; }
                public bool IsExistColumnValue = false;
            }
        }
```

ДОДАТОК Б

GoogleTranslateHelper

```
namespace Terrasoft.Common {
    using System;
    using System.Linq;
    using System.Text;
    using System.Collections.Generic;
    using System.Net.Http;
    using System.Runtime.Serialization;
    using Terrasoft.Core;
    using Terrasoft.Common;
    using Terrasoft.Core.DB;
    using Terrasoft.Core.Store;
    using Terrasoft.Core.Entities;
    using Terrasoft.Core.Process;
    using Terrasoft.Core.Configuration;
    using Newtonsoft.Json;
    using Newtonsoft.Json.Linq;
    public class GoogleTranslateHelper {
        internal readonly string _apiKey =
"AIzaSyCYBAXrIg_mzhnAO_XpJ_1hrKYt0UKWs7c";
        internal readonly string _url =
"https://translation.googleapis.com/language/translate/v2";
         public GoogleTranslateHelper() {}

        public GoogleTranslateHelper(string apiKey) {
            _apiKey = apiKey;
        }
        public TranslateResponse TranslateText(string countryCode,
string text) {
            var list = new List<string>();
            list.Add(text);
            return TranslateText(countryCode, list);
        }
        public TranslateResponse TranslateText(string countryCode,
List<string> textList) {
            var data = new TranslateRequest() {
                Text = textList,
                Target = countryCode
            };
            var json = JsonConvert.SerializeObject(data);
            var postData = new StringContent(json, Encoding.UTF8,
"application/json");
            var url = $"{_url}?key={_apiKey}";
            using(var client = new HttpClient()) {
                string resultJson = client.PostAsync(url,
postData).Result.Content.ReadAsStringAsync().Result.ToString();
```

```
                    var result =
JsonConvert.DeserializeObject<TranslateResponse>(resultJson);
                    return result;
                }
                return null;
            }
            public string GetResponseTranslateText(TranslateResponse
response, int index) {
                if(response == null || index < 0 || index >
response.Data.Translations.Count) {
                    return "";
                }
                return response.Data.Translations[index].TranslatedText;
            }
            public string GetResponseSourceLanguage(TranslateResponse
response, int index) {
                if(response == null || index < 0 || index >
response.Data.Translations.Count) {
                    return "";
                }
                return
response.Data.Translations[index].DetectedSourceLanguage;
            }
        }
        [DataContract]
        public class TranslateRequest {
            [DataMember(Name = "q")]
            public List<string> Text = new List<string>();
            [DataMember(Name = "target")]
            public string Target = "";
        }
        [DataContract]
        public class TranslateResponse {
            [DataMember(Name = "data")]
            public TranslateResultWrapper Data;
        }
        [DataContract]
        public class TranslateResultWrapper {
            [DataMember(Name = "translations")]
            public List<TranslateResult> Translations;
        }
        [DataContract]
        public class TranslateResult {
            [DataMember(Name = "translatedText")]
            public string TranslatedText;
            [DataMember(Name = "detectedSourceLanguage")]
            public string DetectedSourceLanguage;
        }
    }
```